International Journal on Advances in Security

2019 vol. 12 nr. 1&2

The *International Journal on Advances in Security* is published by IARIA.
ISSN: 1942-2636
journals site: http://www.iariajournals.org
contact: petre@iaria.org

International Journal on Advances in Security
Volume 12, Number 1 & 2, 2019

International Journal on Advances in Security
Volume 12, Numbers 1 & 2, 2019

CONTENTS

Towards a Comprehensive Automotive Cybersecurity Reference Architecture

Christoph Schmittner, Martin Latzenhofer,
Shaaban Abdelkader Magdy, Arndt Bonitz, Markus Hofer
Center for Digital Safety & Security
Austrian Institute of Technology
Vienna, Austria
Email: {christoph.schmittner | martin.latzenhofer |
abdelkader.shaaban | arndt.bonitz | markus.hofer}@ait.ac.at

Abstract— **While interconnectivity, complexity, and software-dependency are prerequisites for automated driving, they also increase cybersecurity risks for the whole transportation system. Information and communication technology infrastructure is becoming a second layer for critical transportation infrastructure. In the ongoing Austrian research project CySiVuS, we identified stakeholders which are involved, the services offered and consumed, as well as the risks to a cooperative intelligent transport system (C-ITS) in a structured manner. We collected and categorized different use cases and developed a specific service matrix for C-ITS. Based on an adapted security risk management process we conducted an exemplary security risk management process, using threat modelling. This serves as a fundamental preliminary step towards a comprehensive automotive cybersecurity reference architecture, which is the main objective of the CySiVuS project. Only if all components in the information and communication technology (ICT) infrastructure provide their services in a sufficient quality in accordance with the required security and safety demands, society can rely on an interconnected automotive system.**

Keywords- automotive cybersecurity; cooperative intelligent transport system; service matrix; reference architecture; risk management.

I. INTRODUCTION

In complex and multi-modal environments, smart urban mobility in form of automated driving requires new approaches, which interconnect vehicles with other road users and the road infrastructure. The paper contribution is extended from the authors' previous work [1], which refers to the Austrian national security research project “Cybersecurity for Traffic infrastructure and road operators” (CySiVuS), which aims to tackle cybersecurity and privacy as the key challenges for cooperative traffic infrastructures and automated driving of interconnected cars. The project shifts the perspective from OEMs to traffic infrastructure providers and road service operators. The existing and future road traffic system, together with the associated digital infrastructure is analyzed, and different automatic driving scenarios are collected. Various attack vectors based on different aspects of the whole automotive system require enhanced and further matured cybersecurity standards specific for the automotive domain. Based on these outcomes, the objective is to work out a comprehensive automotive cyber security reference architecture. As a step towards this goal, we identified use cases, their involved services and structured the services based on stakeholder offering and consuming services. Here, all interdisciplinary interests and objectives of stakeholders have been addressed and existing technologies and new technological innovations have been integrated. This article provides an overview of the project’s approach and highlights the urgent need for a complete reference architecture for a (cyber) secure automotive traffic infrastructure.

Main benefits of connected vehicles are a reduction of accidents by communicating road conditions, hazards, and critical situations, as well as increasing traffic efficiency through techniques like platooning or real-time traffic monitoring and control [4]. Reliable connectivity is the mandatory prerequisite for processing various states of the automated vehicle and accelerating further development. Positioning and localization, the creation of complete situational awareness, the reduction of accidents and the increase of comfort and efficiency depend on cooperative and automated driving. Current approaches towards stand-alone vehicles are sufficient for driving on highway or country roads, but these vehicles are not yet ready for urban environments. In our position paper, the idea of a comprehensive automotive cybersecurity reference architecture was postulated [1]. In this paper we included a more detailed consideration of security aspects and additional uses cases, collected from different sources. This leads to a more substantial understanding of how a cooperative intelligent transport system (C-ITS), its components and its respective stakeholders interact with each other. A preliminary step to develop the reference architecture was to establish a service matrix showing the interdependencies of services. Especially in urban environments, it is necessary to integrate automated driving vehicles into a holistic, intelligent transportation system to take advantage of all the potential benefit [2]. Therefore, this paper will specify the infrastructure and connectivity related aspects of automated driving.

Recent projects on a European level [3] identified cybersecurity as a key challenge and risk for future transportation systems. Like physical security and protection for transportation infrastructure, cybersecurity of ICT infrastructure for connected and automated vehicles cannot be left exclusively to the private sector, as their interests and objectives differ, as well as their scopes is restricted to their specific domain. Extensive mobility needs the cooperation of all stakeholders, i.e., automotive original equipment manufacturers (OEMs), infrastructure providers and road service operators, transport facilitators, end users, physical and ICT infrastructure providers, and authorities. All these actors with their different perspectives, as well as all the components together with their interrelationships are considered as one comprehensive infrastructure system. This system relies on extensive and reliable communication between these elements on different tiers. The communication should not be eavesdropped, compromised or manipulated. This makes cybersecurity a critical requirement for a connected automated transportation system, which is vital for the physical transportation infrastructure and a modern society.

This paper is divided into seven sections. After this introduction in Section II, we first give a brief overview of the state of the art of a road transport system. Here we argue that there is no sustainable structured reference architecture that supports a broad perspective on automotive cybersecurity. Risks should be identified, assessed and addressed through an extensive risk management approach. In order to establish a clear reference architecture, we suggest a tailored risk management process as discussed in Section III. Additionally, concrete use cases provide information about the implicit structure of a C-ITS. Consequently, we discuss typical use case scenarios and form use case categories with the potential affecting the security of these automotive services from the infrastructure perspective in Section IV. Based on these specific use cases, we define the structure of a proposed C-ITS service matrix in Section V. We introduce the core aspects and high-level guidelines in Section VI. The final Section VII provides conclusions and outlooks for the near future.

II. State of the Art

For automated vehicles, the Society of Automotive Engineers (SAE) J3016 [5] defines five levels, which give a framework to classify automated vehicles. Currently available mass-market systems reach up to level three. Examples of level three are highway automation and parking assistance systems. The best-known example is Tesla's autopilot and parking assistance system [6]. Higher levels, moving towards high driving automation or even complete automation, are already in a real-world test stage [7], but not yet publicly available. While systems up to level three can rely on in-vehicle sensors and generate the world model on-demand based on local sensor data, higher levels of automation need a pre-mapping to create a world model in which the vehicle is placed via sensor data [8]. This implies that such vehicles require external input to have the latest information and react on permanent or temporary modification of the road infrastructure. This is especially important in urban environments where other localization approaches, relying on Global Navigation Satellite System (GNSS) or road infrastructure (road markings or roadway detection) are more challenging [8].

In the United States, the National Highway Traffic Safety Administration (NHTSA) [9] currently prepares regulations, which require connectivity for active safety features in all new vehicles sold in the US starting from 2020. Such features commonly referred as cooperative active safety, require a high level of trust on outside information and communication. Safety reasons were the primary motivation for OEMs to establish information communication initiated by the vehicle. Security issues – which are following a different paradigm than safety-related ones – are a rather new challenge, currently addressed by the different stakeholders from different viewpoints and with different maturities. Recent hacks show that the majority of their systems lack sufficient security protection [10], [11]. Naturally, OEMs and manufacturers tend to restrict their security focus on the vehicle itself and do not follow a holistic approach, analyzing the whole infrastructure system in which their cars are only elements. Despite first approaches, like the H.R.701 – Security and Privacy in Your (SPY) Car Study Act of 2017 [12], cybersecurity issues of the vehicle are still primarily handled by the vehicle manufacturer, not considering other stakeholders and their security measures. Especially when moving towards connected, intelligent and automated transportation systems, the road traffic infrastructure needs to be looked at in its entirety. As for the legal situation briefly summarized, new regulations are being developed, but they are not timely enough and significantly fragmented. In an automated driving scenario, ICT infrastructure becomes a second layer of critical transportation infrastructure. Hence, the European "Directive on Security of Network and Information Systems", which is also known as the NIS Directive [13] and is enforced since the end of May 2018, applies to the road authorities responsible for traffic control and the operators of intelligent transport systems (ITS). The consequences for the OEMs are not yet clear, even while the car and its communication system are a key component in the superior ITS. There is also an ongoing effort to develop a regulation for considering cybersecurity and cybersecurity processes in modern vehicles through the type approval process [14]. The European directive seeks to ensure a high level of network and information security by improving the common security level of the provider of critical services and digital contents. We expect that the transport sector will become such a critical infrastructure due to the increasing interoperability, connectivity aspects, communication requirements, ICT in general, and privacy issues. Hence, there is an urgent need for full categorization and structured development.

Autonomous and automated vehicles require detailed data about the environment to generate a situational awareness in real time and to ensure their safe movement. It is further evident that automated driving scenarios are not restricted to the vehicles as a stand-alone system. Instead, the vehicles must interact in real-time with the other vehicles and with the infrastructure to assess the current situation. Thus, interoperability is the first key requisite for efficient traffic management, co-operative functions and coordinative autonomy [15]. Furthermore, this implies that the integrity of all data is a prerequisite for autonomous inter-connected driving.

Connectivity between vehicles and other traffic elements is currently still under development, even while standards such as 802.11p already exist [16]. While almost all new premium cars offer connectivity via Global System for Mobile Communication (GSM) or newer standards like long-term evolution (LTE) to a backend system of the manufacturer [17], the main motivation is to reduce costly recalls due to software adaptions and updates [18], as well as to be compliant with the European eCall initiative. Since April 2018, all new vehicles sold in Europe are obliged to be able to automatically call the nearest emergency center in the case of a crash and submit position and crash-related information [3]. Applications like intelligent coordination are already tested and evaluated in real-world scenarios [19]. In such scenarios, vehicles and infrastructure need to communicate within a defined time frame and exchange information like traffic status, travel times, road conditions and road works warnings. There are higher requirements on the connectivity for the next level of cooperation and connectivity. Although there are ITS architecture and connectivity scenarios defined by European Telecommunications Standards Institute (ETSI) [20] available, it is unclear whether vehicles will possess multiple communication systems for each service provider or if the communication will be handled via a central data hub [21]. Different approaches to the future communication infrastructure are presented and discussed in a report of the C-ITS platform [6]. One conclusion is that, in order to support interoperability, stay cost-efficient, reduce the number of attack surfaces and support future applications, the connectivity should follow some sort of coordinated model, considering not only the vehicle, but the complete infrastructure and service value chain [1].

Especially in the field of cybersecurity, multiple indicators show that the current state of the art cannot adequately protect the new and vital role ICT will play in transportation. Automotive cybersecurity is slowly rising to this aspect [22] triggered by research and governmental pressure [17], [23], [24], [25]. Technical developments and industrial awareness of new challenges are followed by the development of first guidelines for tackling the issues [26]. On a higher level, the ITS infrastructure security is also a known issue which is addressed [27]. There is still ongoing discussion who will control and provide the communication infrastructure [3]. Since all mobility and the complete road transportation sector will depend on the ICT system, it is of utmost importance to clarify responsibilities and to achieve a dependable balance between private and public control.

As an additional security property, the protection of personally identifiable information is also an important aspect. A recent survey of the German consumer organization "Stiftung Warentest" showed that almost all connectivity solutions offered by automotive OEMs have weaknesses in protecting privacy [28]. Personal information is exchanged without encryption, and the excessive amounts of information is collected and transmitted, partially without informing the user and without explicit consent. One important discussion is here not only the protection, but also consent to data collection. There are first efforts to develop processes for addressing these issues.

III. Risk Managment

There is currently no domain-specific risk management framework available for the automotive domain [1]. First approaches [26] are promising, but initial evaluations show certain challenges in the application [29]. A guidebook [26] was published at the beginning of 2012, and after being available for half a year again set to "work in progress" status. The International Organization for Standardization (ISO) and SAE founded a common working group developing a standard for the cybersecurity engineering of road vehicles [30], but the publication is currently envisioned for 2020. There is an ongoing effort of the UNECE WP29 - UN Task Force on Cyber security and OTA issues (CS/OTA) to define a minimum required cybersecurity management system (CSMS), which includes a risk-based approach [14]. Due to the focus of the UNECE on type approval, there is a missing consideration of dynamic effects. Recent work focused on dynamic risk assessments in the IoT domain [49] and showed that traditional methods are often challenged by the dynamic nature regarding change times and system boundaries, focus to much on assets and not on the overall system context and did not consider assets as potential attack vectors. In the absence of applicable domain-specific frameworks, we propose to tailor ISO 31000 [31] for the application in the automotive domain. To set up the context, define the stakeholder and the application environment, an appropriate management framework has to be established first. A second main part of the risk management standard proposes the steps depicted in Figure 1.

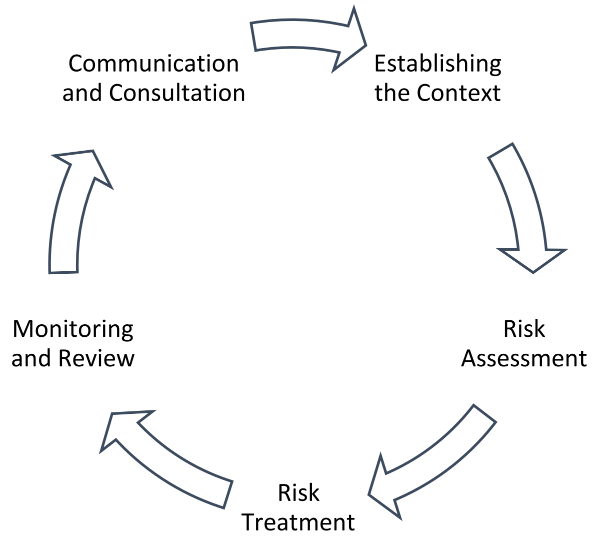


Figure 1: Risk management standard activities

We start by presenting the framework with suggestions on how it can be tailored towards the area of application. The proposed tailoring will be partially carried out on a higher level.

A. *Establishing the Context*

The previously given state of the art overview shows that currently there is no specific regulatory or legal framework for road traffic. Discussions are currently underway for a legal and regulatory framework for road transport, but no clear consensus has yet emerged. The automotive and transportation domain is an important part of ensuring and enabling our modern lifestyle, and therefore it should be avoided that a cybersecurity attack entails the consequences as described in Table I. We consider the following objectives necessary:

TABLE I. CONSEQUENCES OF CYBERSECURITY ATTACKS

Safety	Causes immediate damage to environment or human lives.
Privacy	Causes the loss of control over personal information.
Finance	Causes financial damage.
Operation	Negatively impacts the operation and traffic flow.

We propose two restrictions to these statements. First, we restrict the risk management to direct and immediate consequences. It means that we do not consider second-level consequences, e.g., an operational impact would also impact emergency services and could therefore cause damage to human lives. Our focus lies on the direct consequences. Second, we assess the impact rating on users and society higher than the impact on the organization. That means that safety impacts and financial impacts for users or society are prioritized compared with risks for individual organizations. Society needs to trust and rely on the transportation system, which is supported by ensuring their needs and protection first.

B. *Risk Assessment*

Risk assessment includes identification, analysis and evaluation of risks. While [32] presents examples of risk assessment techniques, none of them are tailored for cybersecurity in the road traffic domain. Multiple proposals exist to extend established safety risk assessment methods towards cybersecurity [33], [34] or to tailor cybersecurity methods for the automotive domain [35], [36]. It should be remarked that there is no general risk assessment implementation, each selected methodology needs to be justified. Depending on the abstraction level, different methods are favored. We propose threat modelling [37] for the analysis of risks. For risk evaluation purposes, we choose four impact levels, divided into four categories, as shown in Table II. This covers most forms of potential impact of attacks. This is an abstraction of the categories proposed by SAE J3061 [26] and EVITA [38]. Both use similar categories.

TABLE II. IMPACT LEVELS

	User / Society	**Service provider / organization**
Safety	1	-
Operational	3	4
Privacy	2	3
Financial	3	4

A critical factor for risk evaluation in cybersecurity is the consideration of likelihood. For example, the railway domain is discussing to consider the potential impact as only input for risk evaluation [39]. This can lead to unlikely risks being given higher priority. Details of the likelihood assessment we are using are presented in [29], but in short, we propose to evaluate the likelihood based on the following four parameters:

- Assumed attacker capabilities
- Ease of gaining information about the systems
- Reachability and accessibility of the system
- Required equipment for an attack

C. *Risk Treatment*

Risk treatment is based on an assessment whether the risk is tolerable for a specific stakeholder. CySiVuS focuses on the society as the most relevant stakeholder, which means that benefits of connected and automated road traffic scenarios should outweigh the risks, especially to human lives. Unless this is the case, we need to either modify the risk by implementing specific technical or organizational measures or avoid the risk altogether by deciding not to implement the scenario. Each risk treatment needs to be followed by an assessment of the effectiveness of the treatment, e.g., if the remaining risk is tolerable and can be accepted. Risk treatment assessment also includes the evaluation if the chosen measures influence other risks or scenarios.

D. *Monitoring and Review*

There are currently no clear responsibilities defined for monitoring and reviewing of risks. This is impeded by the hierarchical silo structure which currently dominates the automotive domain. OEMs only have a restricted system view and are only able to identify risks on their level. Suppliers are responsible for the implementation of risk treatment activities, in fact mitigation measures, for their specific components and identification of change requirements. There is no unambiguous allocation of risk monitoring responsibilities. Established approaches in the automotive domain mainly follow an incident based approach, i.e., reactive behaviour. For cybersecurity challenges, active monitoring and reaction are necessary. We propose to assign a reporting responsibility and develop a cyber incident response plan. In addition to that, risk treatments need to be coordinated between all stakeholders.

E. *Communication and Consultation*

As a continuous and parallel step along the risk assessment, treatment and monitoring, the complete management process needs to be recorded, documented and communicated to the stakeholders. This includes capturing the decisions, results and most importantly the justification for decisions and actions. Only this step makes risk management transparent and comprehensible. It should be remarked that such records are sensitive and could be potentially misused by attackers.

IV. USE CASES

In the CySiVuS project, we identified and collected various use cases for typical situations that a C-ITS has to cope with. The scoping of the use cases supports the identification of stakeholder, roles, components, communication types and data flows, interfaces, as well as all critical services. Thus, they form a starting point for further structuring the system and prepare a preliminary step to develop a comprehensive reference architecture. In the following we introduce the use case collection.

A. *C-ITS Day 1 Use Cases*

The first collection of use cases is based on the Cooperative Intelligent Transport System C-ITS Day 1 Use Case [40]. Day 1 refers to the first set of uses cases implemented and evaluated in the European Corridor – Austrian Testbed for Cooperative Systems (Eco-AT) project. One typical use case is the Road Works Warning (RWW) use case. This use case describes an interaction between vehicles and cooperative roadside elements, which provide information about short time modifications in the road infrastructure to optimize traffic flow and driving strategy. In Eco-AT the transmitted data will only be used as information for the vehicle driver. We will consider the next step and assume that in the future vehicles will automatically act based on the received information. In addition, we will also set up a third Vehicle to Vehicle (V2V) use case, e.g., a vehicle is broadcasting information about position and speed to enable other vehicles, which cannot obtain the information by their built-in sensors to adapt their trajectories according to the current situation.

Figure 2 depicts the introduced use cases. The Road Side Unit (RSU) sends information to all vehicles about a temporal change in the road shape. Vehicles A and B coordinate how B, which is not visible to A, enters the main road and all vehicles receive information from the traffic light system.

B. *CySiVuS Use Cases*

Examples of additional typical use cases in an ITS are listed in the following subsections. We categorized them in five different categories.

1) *Normal cases:* The term normal refers to the most frequently encountered applications of an ITS or an automated vehicle. The typical generic forms in a transport system by using the road infrastructure and its technical equipments are:

- *Transport of people and goods* from place A to place B. Some concrete use cases for transport of people are trips from home to work, to places for leisure activities, to fulfill daily needs, of service providers and time-critical blue-light emergency drives. Examples for transport of goods are home deliveries of daily goods, special transport of chemicals, heavy- or money-transports and time-critical transports like blood and organs. Another type of normal transport use cases are maintenance drives for snow removal, road cleaning or driving school trips.

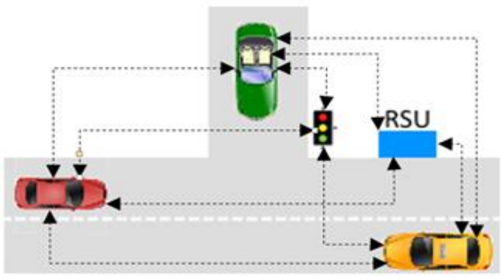


Figure 2: Use cases

- *Departing predefined routes* (e.g., sightseeing) in a specific order or driving as a leisure activity for fun.

2) *Emergency cases:* Emergency cases are cases in which the ITS needs to react on some unforseen event in order to esnure safety of human life and avaialability of the transportation service.

- The *function of vehicle components is suddenly no longer available* (e.g., steering, brake, EMP). It requires reporting to other vehicles, roadside units, original equipment manufacturer (OEM) backend, etc. It is necessary to transfer the vehicle to a safe location or condition and to call for support, e.g., the ambulance or service personnel.
- *The driver is not able to interact* (e.g., impaired, unconscious, or dead);
- *Occurrence of an unexpected event* (e.g., mudslide, avalanche) requires a report to the RSU. It is necessary to transfer the vehicle to a safe location or condition and to call for support, e.g., the ambulance or service personnel.
- *Unauthorized active or passive intervention of third parties* (e.g., hacking, targeted scattering of misleading information) leads to a broadcast information to other vehicles, the RSUs, OEM backend disabling all the network functionalities of the concerned vehicle.
- *Authorized active or passive interference of third parties* (e.g., targeted manipulation of the control unit from outside) requires verifying the authorization, broadcasting information to other vehicles, the RSUs, OEM backend when appropriate, applying the action needed, transmitting the location data and to transfer the vehicle to a safe location or condition.
- *Automatic acquisition of civil vehicles for emergency transports* requires to transfer the vehicle to a safe

location or condition, to verify the authorization and the execution of the action.

3) Comfort cases: There are different perspectives of comfort features. These features are not really necessary for the fundamental task to transport persons or goods from A to B, but they make the process more convenient. This could be beneficial for the driver or passengers, for the OEM or manufacturer, the road or infrastructure operator or for other stakeholders. Some examples are driver assistance systems, intelligent route planning, entertainment routes, multimodal transport services, additional bookable driving performance or features, etc.

4) Road safety and other special cases: These cases concern the road operator and aim to ensure the usability of road sections. Here cases like the distribution of information about dangerous zones or special situations, e.g., due to weather conditions, road works, atypical lane guidance, unusual behavior of other road users, outage of infrastructure elements, maintenance works, and traffic controls are collected.

5) Traffic management cases: There are some situations for the road operator to interfere in the traffic flow to enforce speed limits, release service lanes as additional lanes, service announcements with alternative routes, telematic systems for road work areas.

V. SERVICE MATRIX

Based on the proposed risk management process described in Section III and the collection of use cases discussed in the previous Section IV, we set up a service matrix, where we clarify the inherent dependencies between the services provided by different stakeholders. Firstly, we introduce the different stakeholders, assign the specific components to their responsibility to finally document which stakeholder provides essential services within a C-ITS for the other stakeholders.

A. C-ITS Stakeholder

The various stakeholders, shown in Figure 3, were deducted from the use cases discussed in Section IV. Each stakeholder has their own view on the C-ITS, with different requirements, usage patterns or interests.

- Vehicle users are direct users of the transportation system, and usually those who are transported in the vehicle itself.
- Vehicle manufacturers (i.e., OEMs) and maintenance providers.
- Infrastructure and road operators are typically responsible for the construction and maintenance of roads, road networks, bridges and tunnels and other infrastructure elements.
- Authorities, for example, police, the ministry of transport or delegated organizations, are responsible for ensuring the proper functioning of the transport system.
- Third-party service providers summarize all entities that provide services, for example, fuel stations, mobility clubs, insurance companies or telecommunication providers.
- Society comprises all persons living, working and residing in a given area.

B. C-ITS Infrastructure Components

This section identifies the components and their structural connections. The following formulates the road transport system, seen in Figure 4 below.

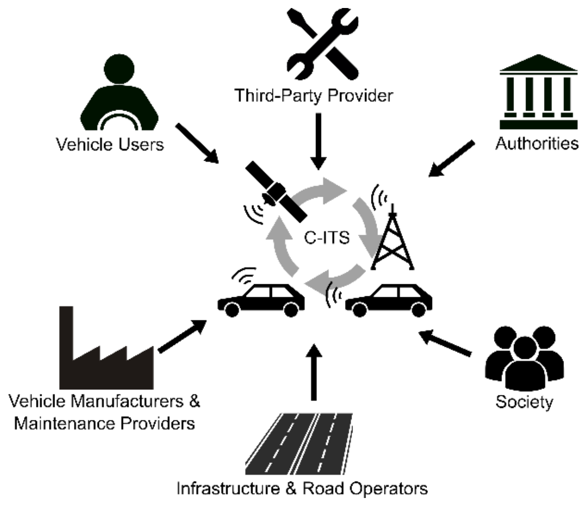

Third-Party Provider
Vehicle Users
Authorities
C-ITS
Vehicle Manufacturers & Maintenance Providers
Society
Infrastructure & Road Operators

Figure 3: Stakeholder of a C-ITS

1) Communication in the Vehicle. There are various communication requirements in the vehicle between Electronic Control Units (ECU), sensors, and actuators. Different communication busses help to structure the data flow.

- Classically, the Controller Area Network bus (CAN) is used for communication between control units [46]. The CAN protocol was defined in 1986. In 1991 the first vehicle with CAN was available on the market [47]. The CAN bus was developed as a standard vehicle protocol that minimizes cabling effort and enables prioritization of communication. The big amounts of data such as generated by sensors or camera systems can be a big challenge for a CAN network.
- A Local Interconnect Network (LIN) [48] was developed to a network which an increasing number of sensors in the vehicle with the control units. Most of the sensors have requirements with fewer capabilities than the CAN network. Therefore, the CAN network can be useful for simple networks.

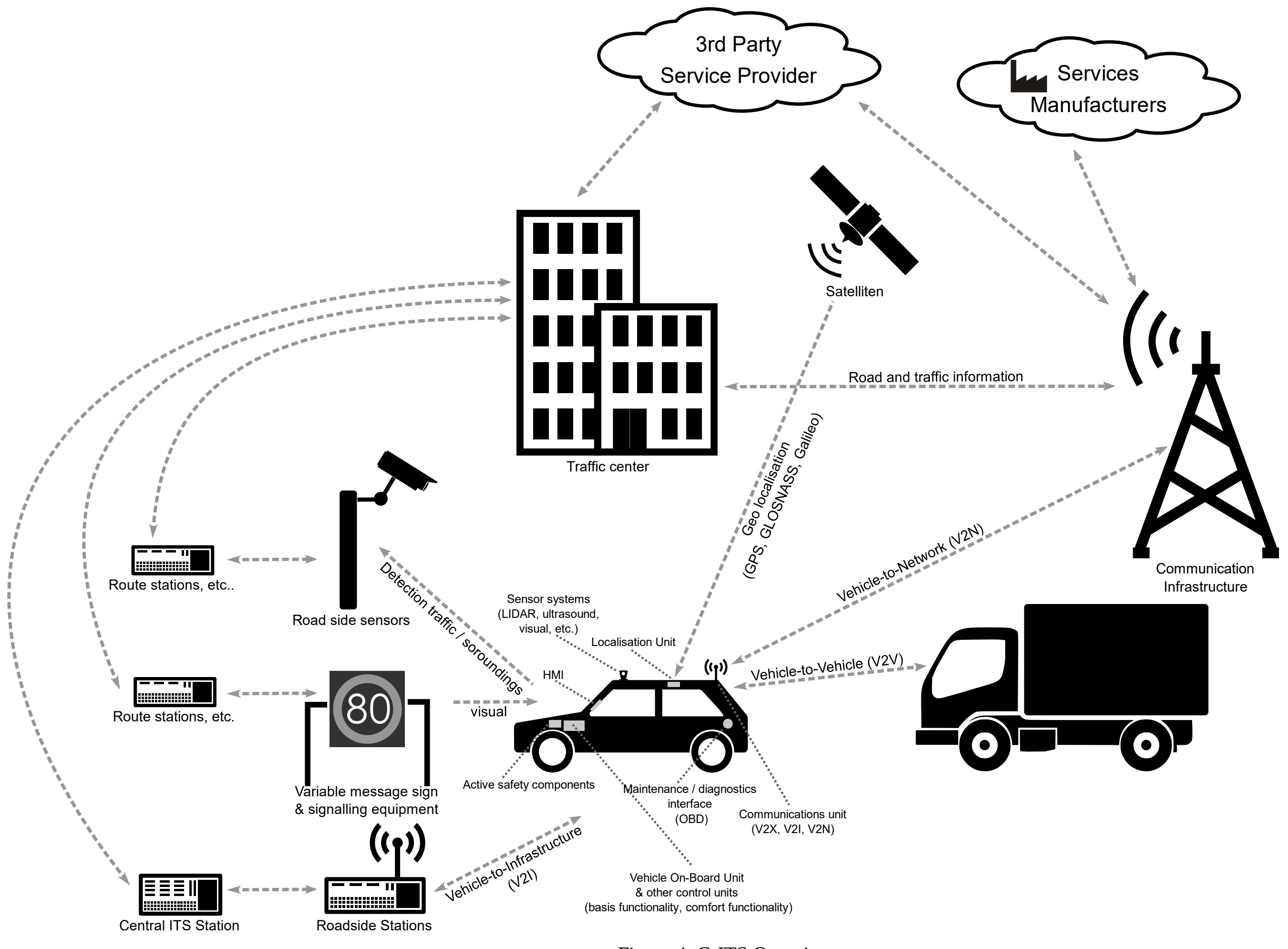

3rd Party Service Provider
Services Manufacturers
Traffic center
Satelliten
Road and traffic information
Geo localisation (GPS, GLOSNASS, Galileo)
Communication Infrastructure
Vehicle-to-Network (V2N)
Route stations, etc..
Road side sensors
Detection traffic / soroundings
Sensor systems (LIDAR, ultrasound, visual, etc.)
Localisation Unit
HMI
Vehicle-to-Vehicle (V2V)
Route stations, etc.
80
visual
Variable message sign & signalling equipment
Active safety components
Maintenance / diagnostics interface (OBD)
Communications unit (V2X, V2I, V2N)
Vehicle-to-Infrastructure (V2I)
Vehicle On-Board Unit & other control units (basis functionality, comfort functionality)
Central ITS Station
Roadside Stations

Figure 4: C-ITS Overview

- The Media Oriented Systems Transport (MOST) bus was developed to enable the communication of media content in the vehicle. It can be used, to transmit video content to monitors in the vehicle.

2) Vehicle Interfaces: There are different types of interfaces, which allow the transfer of data to and from the vehicle:

- On-board diagnostics (OBD) and increasingly USB interfaces require a physical connection. ODB is designed for Machine-to-Machine (M2M) communication between the diagnostic system and the vehicle system, while USB is a way for the user to exchange data with the vehicle.
- WLAN, Bluetooth, and WLAN-based V2X protocols provide medium-range wireless communication. Normally, no additional infrastructure is required, and the devices communicate directly with each other. The interfaces are intended for communication between systems.
- The last group consists of interfaces for long-distance communication, such as radio receivers or mobile (cellular) radio transceivers. The radio receiver is the only unidirectional interface.

3) Road Infrastructure Interfaces: The road infrastructure has the following general communication interfaces in order to provide data exchange with RSUs.

- Interfaces to internal sensors and actuators, e.g., to traffic detectors or traffic control technology such as variable-message signs
- Interfaces to vehicles moving on the road infrastructure such as radio interfaces from roadside units to vehicles
- Interfaces to third-party providers, e.g., DATEX II Web Service

4) Backend infrastructure and services. This category compromises all non-road specific background services, i.e., the backend systems of the car manufacturer or navigation service providers. Depending on the service, the vehicle ECU might either be directly connected via a cellular radio modem to a background service (for example for accessing manufacturer updates) or use a modem integrated to a dedicated device, i.e., a navigation unit.

C. C-ITS Service Matrix

The types of services that can be implemented for automated driving applications are diverse. To gain a better overview and understanding, the identified services of the entire road traffic system should be visualized so that not only the actual basic components, but also their inherent connections can be easily grasped. The service matrix introduced in this section is one possibility which helps to get a good overview of the complexity by showing which services are offered by which stakeholders and which stakeholders in turn used them. This means, the possible services should be considered from different stakeholder perspectives, i.e., vehicle (user), infrastructure and road operators, vehicle manufacturers (OEM) and maintenance providers, third party service providers, authorities, and the society as a whole.

The following service matrix in Figure 5 shows the evaluation being carried out by the project team as a type of expert evaluation. It shows how the importance of services provided by a certain entity for the user of the service, i.e., the extent to which the service user is impaired by the discontinuation of the service in the safe execution of his tasks and his responsibilities.

The first step of generating this matrix was to find and define exemplary services for each combination of two of the stakeholders. For example, services being provided by infrastructure and road operators to vehicle users are numerous, including but not limited to traffic flow information, construction site warnings, traffic status information, and route information. Less strong, for example, is the connection between infrastructure and road operators and third-party service providers, the only relevant services here are traffic radio, and in some cases information for the modification of infrastructure.

The final service matrix was derived from this intermediate matrix with all services found and then rated for criticality by the experts in the project consortium. The scale used for rating goes from 0 (irrelevant), over 1 (little impact) and 2 (impairment) up to 3 (critical). As an example, the services provided by an infrastructure and road operator is crucial to a vehicle user, but not to OEMs or maintenance providers. A general observation revealed by the service matrix is that most of the stakeholder rely on services provided by others. This means that a C-ITS is a highly interdependent overall system and requires a well-structured reference architecture to be understandable by key players forming a C-ITS like authorities, decision makers as well as the industry.

Service Customer ➔ / Service Providers 🡻	Vehicle user	Infrastructure and Road Ops	OEM, Maintenance Provider	Third Party Service Provider	Society	Authorities
Vehicle user	2	2	2	1	0	2
Infrastructure and Road Ops	3	1	0	1	1	2
OEM, Maintenance Provider	3	1	0	2	0	2
Third Party Service Provider	0	1	1	2	1	1
Society	2	1	2	1	2	1
Authorities	3	2	2	1	2	2

Figure 5. Service provider/service user matrix

D. Security analysis example

We identify security threats based on the data flow between vehicle A, B, and roadside units in Figure 2. We use the threat analysis tool [36] developed by Austrian Institute of Technology. The threat tool uses several source materials to ensure a range of threats is considered. The following source documents were used to develop the threats database:

- Threat Modeling for Automotive Security Analysis [36]
- Connected cars Threats, vulnerabilities and their impact [41]
- The ENISA Threat Landscape 2015, Top Threats [42].

Figure 6 depicts the data flow between vehicle A, B, and RSUs. Based on the given input to the tool, and without any security mitigation measures, 55 threats were identified.

The threat tool classifies the detected potential threats into six main classes according to the STRIDE model [43], i.e., Spoofing, Tampering, Repudiation, Information disclosure, Denial of Service (DoS), and Elevation of privilege. Table III summarizes the numbers of the detected threats regarding the STRIDE model.

TABLE III. DETECTED THREATS ACCORDING TO STRIDE CLASSIFICATION

Type	Numbers
Denial of Service	7
Elevation of Privilege	7
Information Disclosure	15
Repudiation	5
Spoofing	14
Tampering	7

The tool performs a risk assessment process to classify the risk of the identified threats as an extreme, high, medium, or low risks. Figure 7 shows statistical percentages of risks in the

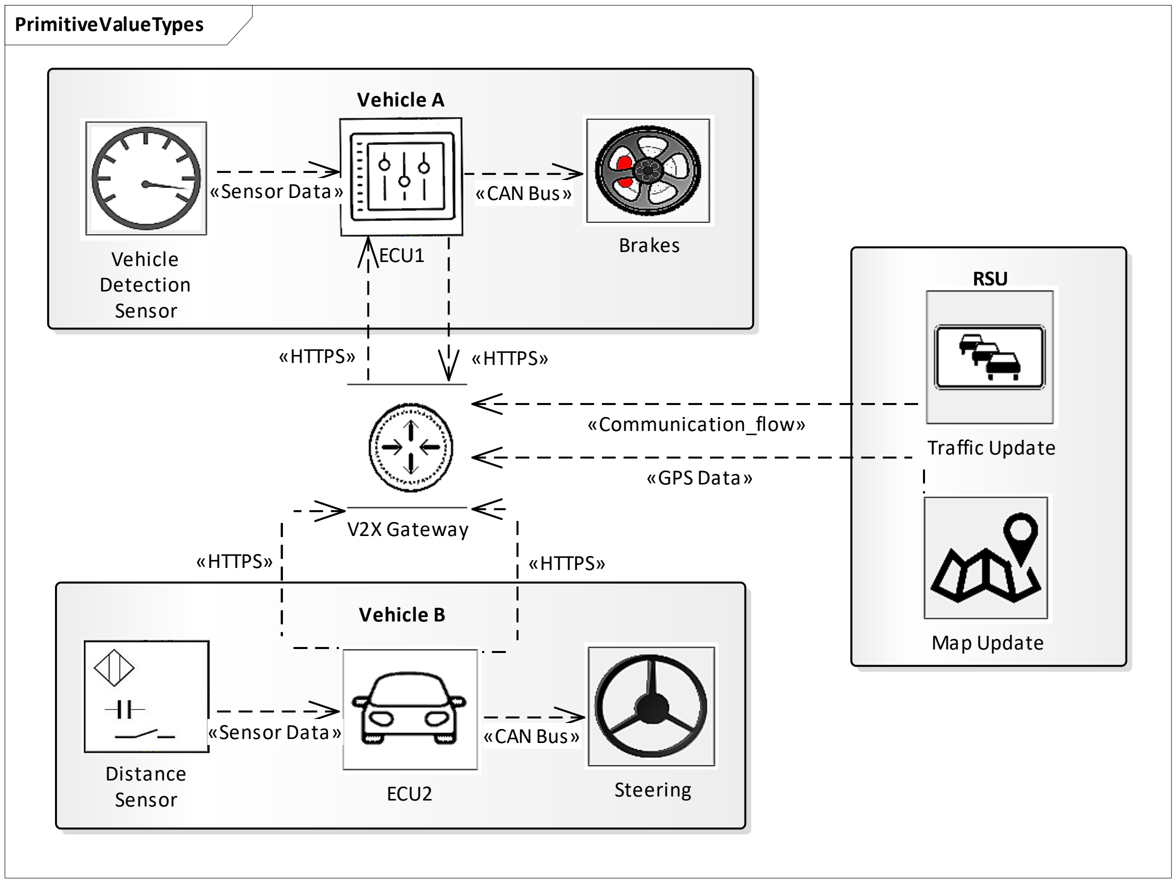


Figure 6: Data flow model for threat assessment

identified threats. From the observed risk statistic, we see that the highest number of risks are medium risks.

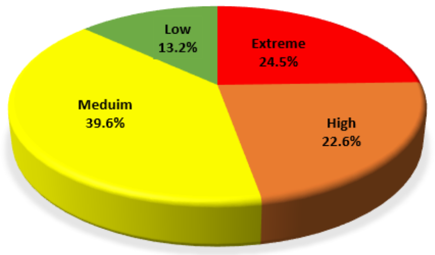


Figure 7: Statistical percentage of risks

We focus in the following on the interaction type and the corresponding threat, seen in Table IV.

TABLE IV. DELIVER MALICIOUS UPDATES TO VEHICLE B [PRIORITY: HIGH]

Category	**Spoofing**
Description	Deliver Malicious Updates to Vehicle B
Justification	<no mitigation provided>
Attack method	Spoofing vehicle A in order to send malicious updates.

For connected automotive vehicles and their corresponding brakes control and steering algorithms, the correct and especially secure reception of safety and kinematic related messages is of utmost importance. A manipulated sending unit from some distance away could communicate status information, e.g., nonexistent barriers, road works or vehicle positions ahead leading to slow down or even stop of the traffic culminating to accidents. To prevent such a threat, we propose distance-bounding protocols that allow a safe decision if the communication partner is within a certain radius, defined as bubble [44], [45].

The adapted Table V summarizes the considerations detailed above.

TABLE V. DELIVER MALICIOUS UPDATES TO VEHICLE B [PRIORITY: LOW]

Category	**Spoofing**
Description	Deliver Malicious Updates to vehicle B
Justification	~~<no mitigation provided>~~ *Distance bounding avoids remote attacks and requires physical access to the environment in order to conduct the attack*
Attack method	Spoofing vehicle A in order to send malicious updates.

This capability requires the introduction of a bidirectional communication link between Verifier (V) and Proofer (P) and a fast processing of the challenge sent from V to P. This reduces the evaluated attack likelihood by enforcing physical access to conduct such attacks and reduces the risk to a tolerable level.

VI. Reference Architecture

An automotive reference architecture for security analysis was presented in [11]. While it includes the elements of communication between backend and vehicle, it does not consider all relevant scenarios for C-ITS like V2V communication. Furthermore, it only defines the technical elements and does not differentiate between environments, stakeholder, objects in the architecture a division. However, this pure technical approach is not sufficient and to apply the reference architecture in practice, this separation is vital.

As a first approach, we divide the ITS into five clusters of elements as shown in Figure 8. On the physical side (blue, left side), we have vehicles, infrastructure and personal devices. The provider's side (green, right side) contains elements which are maintained and operated by infrastructure operators and road service providers offering mobility services (grey, lower side) available to the users (yellow, upper side). All elements are interconnected by a communication system (orange, in the middle). It should be highlighted that these blocks can overlap, e.g., infrastructure providers can also provide services; and blocks can contain multiple diverse sub-blocks, e.g., communication collects a multitude of techniques like wireless networking (WLAN) or GSM, which can be applied for V2V or Vehicle to Infrastructure (V2I) communication.

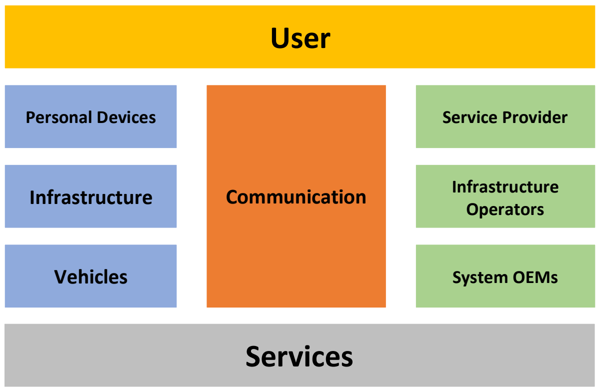


Figure 9: Clustering of elements in the transportation system

Moreover, the approach described above offers a relatively high-level view on the system, which is, to a certain degree, architecture independent. As it is discussed in [6] and [21], it is still in discussion how the connectivity architecture will finally look like, but all discussed architectural variants fit in the presented structural model. Such a structural model helps to identify the involved parties, allows assigning risk mitigations to technical elements and assigns the responsibility of implementing and maintaining these risk mitigations to involved parties. To be practically applicable, the identified risk mitigation measure is implemented in infrastructure and vehicle, conducted by system OEM and infrastructure providers, which is shown in Figure 9.

A possible solution approach is a structured multi-tiered reference architecture. However, a consistent risk management methodology is a critical success factor for developing a unified architecture across all perspectives. Our approach is to take the widely accepted risk management standard ISO 31000 [31] as a basis and tailor it to the automotive requirements.

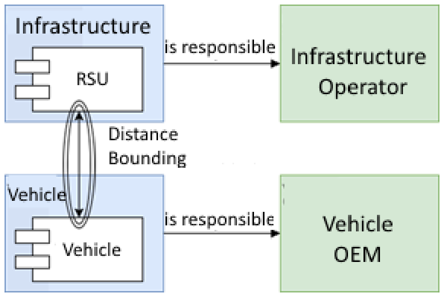


Figure 8: Application of the structural model

We discuss the five main steps of the risk management process when we apply it to a road traffic system. It is crucial to restrict the proposed approach to direct risks only and to weight the impacts differently depending on the consequences. The risk management analysis steps are

essential to finding an appropriate mixture of applicable methods to form a reliable methodology for the assessment. Additionally, the evaluation of the likelihood and the handling of uncertainty needs to be solved. Risk treatment in a complex and interconnected environment must consider different actors.

VII. Conclusion and Outlook

In this paper we analyze the technological and legal state of the art of automated driving for smart urban mobility. We conclude that the current state of the art is not yet sufficient with the complex requirements of such an environment. We identified four current challenges to a comprehensive traffic road system: The interoperability of the components among the vehicles as well as the infrastructure elements, connectivity and communication tasks especially for interacting and cooperation of the different components, ICT in general and cybersecurity issues to address security threats, and privacy finally aspects which subsume protection requirements of personal data of the vehicle drivers. There are efforts to form a compliant legal and technological framework, but all these considerations are not yet completed. By considering a tailored risk management process and collecting and categorizing different use cases, we initially identified stakeholders and components. Based on this, we developed a C-ITS service matrix to visualize the service usage between the five stakeholder groups and to reveal their interdependencies among each other. This is a potential starting point for future cyber security investigations.

Considering the challenges for cybersecurity risk management in dynamic environments like Internet of Things we already consider some of the challenges [49], [50]. By following a model-based approach we are able to automatically re-asses the system and our risk evaluation already considers assets as potential attack vectors. The model-based approach is challenged by systems with unclear boundaries or composition, e.g., new risks due to a change in system composition are difficult to measure.

The primary task of the CySiVuS research project is to develop a wide-ranging model on all necessary perspective levels, which the rough approach introduced in this article could be a starting point. By conducting the risk management process and developing the reference architecture, we show the multidimensional nature of a road traffic system. The main upcoming challenges are a concrete in-depth-analysis of risk assessment by applying adapted risk management methods. The next step is to develop a comprehensive automotive reference architecture based on the considerations introduced in the previous section. The main objective is to determine an appropriate layered visualization taking the different stakeholders, components, services into account.

Acknowledgment

The research project "Cybersicherheit für Verkehrsinfrastruktur- und Straßenbetreiber" (CySiVuS, in English: „Cyber security for transport infrastructure and road operators") (Project-Nr. 865081) is supported and partially funded by the Austrian National Security Research Program KIRAS (Federal Ministry for Transport, Innovation and Technology (BMVIT) and Austrian Research Promotion Agency (FFG) 2017).

References

[1] C. Schmittner, M. Latzenhofer, S. Abdelkader, and M. Hofer, "A Proposal for a Comprehensive Automotive Cybersecurity Reference Architecture," in *VEHICULAR 2018, The Seventh International Conference on Advances in Vehicular Systems, Technologies and Applications*, Venice, 2018, pp. 30–36

[2] Q. Xu, K. Hedrick, R. Sengupta, and J. VanderWerf, "Effects of vehicle-vehicle/roadside-vehicle communication on adaptive cruise controlled highway systems," in *Proceedings IEEE 56th Vehicular Technology Conference*, Vancouver, BC, Canada, 2002, vol. 2, pp. 1249–1253

[3] C-ITS Platform, "Working Group 6 Access to in-vehicle resources and data," Dec. 2015.

[4] European Telecommunications Standards Institute (ETSI), "ETSI TR 102 638 V1.1.1; Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Definitions." Jun-2009 [Online]. Available: https://www.etsi.org/deliver/etsi_tr/102600_102699/102638/01.01.01_60/tr_102638v010101p.pdf [Accessed: 28-05-2019]

[5] SAE, "J3016 Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles," *2016*.

[6] M. Dikmen and C. M. Burns, "Autonomous Driving in the Real World: Experiences with Tesla Autopilot and Summon," in Proceedings of the 8th International Conference on Automotive User Interfaces and Interactive Vehicular Applications, 2016, pp. 225–228

[7] M. Aeberhard *et al.*, "Experience, Results and Lessons Learned from Automated Driving on Germany's Highways," *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 1, pp. 42–57, 2015.

[8] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous Localization and Mapping: A Survey of Current Trends in Autonomous Driving," *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 3, pp. 194–220, Sep. 2017.

[9] NHTSA, "NHTSA | National Highway Traffic Safety Administration." [Online]. Available: https://www.nhtsa.gov/ [28-05-2019]

[10] A. Greenberg, "The Jeep Hackers Are Back to Prove Car Hacking Can Get Much Worse.", Wired, 08.01.16 [Online]. Available: https://www.wired.com/2016/08/jeep-hackers-return-high-speed-steering-acceleration-hacks/ [Accessed: 28-05-2019]

[11] J. Brückmann, T. Madl, and H. J. Hof, "An Analysis of Automotive Security Based on a Reference Model for Automotive Cyber Systems," *SECURWARE 2017 : The Eleventh International Conference on Emerging Security Information, Systems and Technologies* [Online]. Available: https://www.researchgate.net/publication/319932479_An_Analysis_of_Automotive_Security_Based_on_a_Reference_Model_for_Automotive_Cyber_Systems [Accessed: 28-05-2019]

[12] Library Congress, "H.R.701 - 115th Congress (2017-2018): SPY Car Study Act of 2017." [Online]. Available: https://www.congress.gov/bill/115th-congress/house-bill/701/text [Accessed: 28-05-2019]

[13] European Union, *Directive (EU) 2016/1148 of the European Parliament and of the Council of 6 July 2016 concerning measures for a high common level of security of network and information systems across the Union*, vol. L194. 2016 [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=OJ:L:2016:194:FULL&from=EN [Accessed: 28-05-2019]

[14] Secretary of the UN Task Force on Cyber Security and Over-the-Air issues, "Draft Recommendation on Cyber Security of the Task Force on CyberSecurity and Over-the-air issues of UNECE WP.29 GRVA (Informal Document)." 20-Sep-2018 [Online]. Available: https://www.unece.org/fileadmin/DAM/trans/doc/2018/wp29grva/GRVA-01-17.pdf [Accessed: 28-05-2019]

[15] NHTSA, "Vehicle-to-Vehicle Communications: Readiness of V2V Technology for Application," National Highway Traffic Safety Administration, US Department of Transportation.

[16] European Telecommunications Standards Institute (ETSI), "ETSI ES 202 663 V1.1.0; Intelligent Transport Systems (ITS); European profile standard for the physical and medium access control layer of Intelligent Transport Systems operatingin the 5 GHz frequency band." Nov-2009 [Online]. Available: https://www.etsi.org/deliver/etsi_es/202600_202699/202663/01.01.00_50/es_202663v010100m.pdf [Accessed: 28-05-2019]

[17] C. Valasek and C. Miller, "A Survey of Remote Automotive Attack Surfaces," IOActive [Online]. Available: https://ioactive.com/wp-content/uploads/2018/05/IOActive_Remote_Attack_Surfaces.pdf [Accessed: 28-05-2019]

[18] H. A. Odat and S. Ganesan, "Firmware over the air for automotive, Fotamotive," in *IEEE International Conference on Electro/Information Technology*, 2014, pp. 130–139.

[19] EcoAT, “Der österreichische Beitrag zum Kooperativen ITS Korridor“ [Online]. Available: http://eco-at.info/ [Accessed: 28-05-2019]

[20] ETSI, “Automotive Intelligent Transport Systems,” *European Telecommunications Standards Institute*. [Online]. Available: https://www.etsi.org/technologies-clusters/technologies/automotive-intelligent-transport [Accessed: 28-05-2019]

[21] B. Datler, “A Road Operator’s View on Cloud-based ITS – Requirements and Cooperation Models,” *23rd ITS World Congress*, 2016

[22] E. Khayari, “SECURE AUTOMOTIVE ON-BOARD ELECTRONICS NETWORK ARCHITECTURE,” p. 9.

[23] D. Spaar, “Car, open yourself! Vulnerabilities in BMW’s ConnectedDrive,” pp. 86–90, 2015.

[24] J. Petit and S. E. Shladover, “Potential Cyberattacks on Automated Vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 546–556, Apr. 2015.

[25] D. C. Miller and C. Valasek, “Remote Exploitation of an Unaltered Passenger Vehicle,” p. 91.

[26] *Cybersecurity Guidebook for Cyber-Physical Vehicle Systems*. SAE, 2016

[27] ECo-AT, “ECo-AT SWP3.4 Security,” ECo-AT, 2016 [Online]. Available: http://www.eco-at.info/system-spezifikationen.html [Accessed: 28-05-2019]

[28] “Connected Cars: Die Apps der Auto-hersteller sind Daten-schnüffler.” 26-Sep-2017 [Online]. Available: https://www.test.de/ Connected-Cars-Die-Apps-der-Autohersteller-sind-Datenschnueffler-5231839-5231843/ [Accessed: 28-05-2019]

[29] C. Schmittner, Z. Ma, C. Reyes, O. Dillinger, and P. Puschner, “Using SAE J3061 for Automotive Security Requirement Engineering,” in *Computer Safety, Reliability, and Security*, 2016, pp. 157–170.

[30] International Organization for Standardization, Ed., *ISO/SAE CD 21434 Road Vehicles - Cybersecurity engineering*. ISO, Geneva, Switzerland [Online]. Available: https://www.iso.org/standard/70918.html [Accessed: 28-05-2019]

[31] “ISO 31000:2009 Risk management -- Principles and guidelines,” ISO, Feb. 2018 [Online]. Available: https://www.iso.org/standard/65694.html [Accessed: 28-05-2019]

[32] International Organization for Standardization, Ed., *ISO 31010:2009 Risk management - Risk assessment techniques*. ISO, Geneva, Switzerland, 2009.

[33] G. Macher, H. Sporer, R. Berlach, E. Armengaud, and C. Kreiner, “SAHARA: A security-aware hazard and risk analysis method,” in *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2015, pp. 621–624.

[34] C. Schmittner, T. Gruber, P. Puschner, and E. Schoitsch, “Security Application of Failure Mode and Effect Analysis (FMEA),” in *Computer Safety, Reliability, and Security*, 2014, pp. 310–325.

[35] “A case study of FMVEA and CHASSIS as safety and security co-analysis method for automotive cyber-physical systems,” *ResearchGate*. [Online]. Available: https://www.researchgate.net/publication/282792587_A_case_study_of_FMVEA_and_CHASSIS_as_safety_and_security_co-analysis_method_for_automotive_cyber-physical_systems [Accessed: 28-05-2019]

[36] M. Zhendong, and C. Schmittner. "Threat modeling for automotive security analysis." Advanced Science and Technology Letters 139 (2016): 333-339.

[37] F. Swiderski and W. Snyder, *Threat Modeling (Microsoft Professional)*. 2004.

[38] “E-safety vehicle intrusion protected applications,” EVITA, 2008 [Online]. Available: https://www.evita-project.org/Publications/EVITAD0.pdf [Accessed: 28-05-2019]

[39] J. Braband, “Towards an IT Security Framework for Railway Automation,” presented at the Embedded Real Time Software and Systems,” *Toulouse*, Feb. 2014.

[40] “C-ITS Strategy Austria [C-ITS Strategy Austria],” Network Drivers, Promote Efficiency and Safety in Transport., Jun. 2016 [Online]. Available: https://www.bmvit.gv.at/en/service/publications/transport/downloads/citsstategy.pdf [Accessed: 28-05-2019]

[41] S. Strobl, D. Hofbauer, C. Schmittner, S. Maksuti, M. Tauber, and J. Delsing, “Connected cars — Threats, vulnerabilities and their impact,” in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, 2018, pp. 375–380.

[42] “ENISA Threat Landscape 2015 — ENISA.” [Online]. Available: https://www.enisa.europa.eu/publications/etl2015 [Accessed: 28-05-2019]

[43] “Uncover Security Design Flaws using The STRIDE Approach,” *MSDN Magazine*, Nov. 2006 [Online]. Available: https://adam.shostack.org/uncover.html [Accessed: 28-05-2019]

[44] K. B. Rasmussen, S. Capkun “Realization of RF Distance Bounding”. InUSENIX Security Symposium 11-08-2010 (pp. 389-402).

[45] G. P. Hancke and M. G. Kuhn, “Attacks on Time-of-flight Distance Bounding Channels,” in *Proceedings of the First ACM Conference on Wireless Network Security*, New York, NY, USA, 2008, pp. 194–202 [Online]. Available: http://doi.acm.org/10.1145/1352533.1352566 [Accessed: 28-05-2019]

[46] “History of CAN technology.” [Online]. Available: https://www.can-cia.org/can-knowledge/can/can-history/ [Accessed: 28-05-2019]

[47] “Mercedes W140: First car with CAN.” [Online]. Available: https://can-newsletter.org/engineering/applications/160322_25th-anniversary-mercedes-w140-first-car-with-can [Accessed: 28-05-2019]

[48] International Organization for Standardization, Ed., *ISO 17987-1:2016 Road vehicles; Local Interconnect Network (LIN); Part 1: General information and use case definition*. ISO, Geneva, Switzerland, 2016.

[49] J. R. C. Nurse, S. Creese, and D. De Roure, “Security Risk Assessment in Internet of Things Systems,” IT Prof., vol. 19, no. 5, pp. 20–26, 2017.

[50] J. R. C. Nurse, P. Radanliev, S. Creese, and D. De Roure, “If you can’t understand it, you can’t properly assess it! The reality of assessing security risks in Internet of Things systems,” Living in the Internet of Things: Cybersecurity of the IoT - 2018,

MEADcast: Explicit Multicast with Privacy Aspects

Vitalian Danciu*, Cuong Ngoc Tran*

* Ludwig-Maximilians-Universität München
Oettingenstr. 67, 80538 München, Germany
Email: {danciu, cuongtran}@mnm-team.org

***Abstract*—We find that existing multicast protocols require either the participation of hosts in group management or partial address lists of the group members to be sent to end-points (hosts), thus creating a privacy issue. Many services suitable for multicast are transmitted via massive unicast for technical or management reasons outside the sphere of influence of the sender. The large amount of identical payload transmitted constitutes a significant waste of network resources. We address these issues by presenting *MEADcast*, a multicast protocol intended to support a smooth transition from massive unicast to sender-centric multicast over the Internet. Senders perform all group management while receivers do not require explicit support for the protocol. The protocol copes with varying degrees of support by routers in the network and avoids the disclosure of end-point addresses to other end-points. Performance evaluation shows a decrease of the total traffic volume in the network of up to 1:5 as compared to unicast, suggesting suitability for applications, such as Internet Protocol Television (IP-TV), video conferences, online auctions and others.**

Keywords*—*Privacy-Preserving Multicast; Agnostic Destination; Explicit Multicast; Sender-Centric Multicast.

I. Introduction

Applications replacing traditional broadcast services (IP-TV, IP-Radio), phone and video conferencing, and also technical services for software update or large-scale configuration may profit from an *n:m*, multicast, distribution scheme. Today, these applications still rely mostly on unicast transmission despite multicast having been available for a long time. In this text, we propose MEADcast (see also [1]), a multicast protocol intended to allow the optional and gradual introduction of 1:n communication between a sender and end-points into networks with initially unknown support for our protocol.

A. Challenges to Multicast Adoption

Many applications have evolved into their present form based on the technologies of the World Wide Web, including a connection-oriented, TCP-based communication layer and using the tacit design assumption that each service instance induces a one-to-one relationship between a service provider and a service user. The difficulty to roll out and maintain specialised client software for a service, compounded with the broad availability of a general-purpose, multimedia-capable client – the web browser – and the lack of pressure on service providers to conserve transmission capacity have led to the unicast design of applications that clearly lend themselves to multicast.

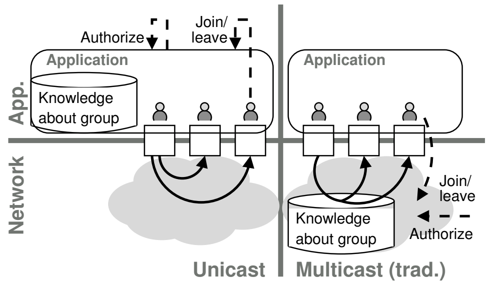


Figure 1. Knowledge and management actions in unicast and multicast.

B. Technical background

Typical multicast schemes are based on managed groups (e.g., [2], [3], [4]). End-points may join a multicast group and the network forwards messages addressed to that group to all its members, i.e., to all end-points that joined it. As a rule, a multicast group is symmetric in allowing any participant to address a message to all others. Unfortunately, it requires the network manager to effect configuration reflecting that a given application uses a different kind of network function, while the user is responsible for configuring the application to use multicast. The setup for services being provided across networks and thus across administrative domains always requires the cooperation of each participant domain's network managers.

Another important reason for the lack of multicast adoption seems to lie in the difference of scope of the application and the network function: the local scope of traditional multicast is inherent in the need to apply network configuration (e.g., IGMP) to the access network routers, while the applications are being provided from outside the local scope (i.e., the Autonomous System) over the global Internet.

As illustrated in Figure 1, by requiring an end-point to join and leave the multicast group that supports the desired application, the use of multicast

1) requires network management to authorize a service session and possibly setup (multicast routers),
2) requires the user to execute a network management action,
3) requires transfer of knowledge on group membership at the application level to a multicast group managed within the network layer and
4) introduces state to the otherwise state-less (from the view of the end-point) IP communication.

A number of additional properties exacerbate the perceived drawbacks to multicast use:

5) If the network-level setup of multicast fails, there is no automatic fall-back to unicast; instead, the application must detect and handle the failure.
6) All participants in a service must have multicast support.
7) Re-configuration of the application group requires re-configuration of the network.
8) Knowledge about the identities of the participants in a service session is present in the network, possibly in several administrative domains.

Applications seem therefore to prefer unicast even at the expense of the higher transmission volume, or Application-Layer Multicast (ALM) (e.g., [5], [6]) in spite of it being application specific and requiring a network function within the application's code.

In essence, ALM reduces the *n:m* multicast pattern to the asymmetric case of *1:n* communication, where a single sender addresses a group of receivers. In this case, it is sufficient for the sender to hold knowledge about the group. Since the sender necessarily implements the application layer of the service being provided, group management may be transacted at the application level. Such communication is easily implemented over unicast transmissions. However, it requires receiver-side configuration and does not profit from network support.

1) Non-proliferation of multicast: Although technical means to address inter-domain multicast continue to be developed (e.g., [7], [8], [9]), the adoption of multicast requires an incentive to both the service provider and to the access network operator. It requires the establishment of a cooperation between them by setting up routers supporting the inter-domain multicast scheme. What is more, it requires a mapping of services onto the (limited) multicast address space, implying a-priori knowledge about the services being used. Outside of applications provisioned centrally in the domain of an access network operator, this knowledge is generally not available: the reception of broadcast services (e.g., IP-TV, Internet radio) as well as the use of multi-party audio/video conferencing with participants outside of the network operator's domain is typically initiated by end users. Even when a certain service is provided locally, end users might still opt for an alternative service provided from outside the domain, if the local one requires a specific request or setup while the "foreign" one does not.

C. Contribution

We propose to combine the benefits of multicast to agnostic receivers with those of optional network support.

We introduce a protocol named Multicast to Explicit Agnostic Destinations (MEADcast) to allow sender-based multicast of IPv6 over the Internet. The novelty of MEADcast is that it protects receivers' anonymity and allows a gradual, pro-active and selective transition between multiple unicast and network-supported multicast. As the protocol favours conservative decisions, we present studies of the transmission cost in the network performed by simulating randomized as well as designed situations.

D. Technical overview

MEADcast implements a sender-centric multicast in that all knowledge about the receiver group, the network topology and the availability of MEADcast-capable routers (called *MEADcast routers* in this text) is gathered at and decided upon by the sender. Given an initial list of receivers, the sender commences to send data in unicast to each receiver while simultaneously probing the network for the presence of MEADcast routers and hence for the option to consolidate some of the unicast streams into multicast. Multicast packet headers reflect the MEADcast router responsible for translating the multicast packets into (multiple) unicast packets. Receiving end-points (called *receivers* in this text) always receive true unicast packets either directly from the sender or generated by a MEADcast router based on a multicast packet. Only unicast addresses are used in the protocol.

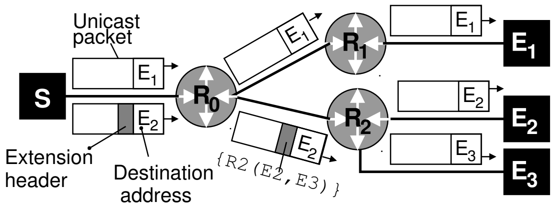


Figure 2. Multicast to agnostic receivers.

Multicast packets begin with a standard IPv6 header addressed to one of the multicast receivers on a path, followed by a Hop-by-Hop Routing Header with Router Alert. The addresses of all multicast receivers on a path as well as the MEADcast router(s) responsible for translation are encoded into a multicast header. It is typically followed by a UDP header. The addressing pattern is similar to the one in Internet email, where one receiver is addressed directly (To:) while all receivers are included in the carbon copy (CC:) list. The protocol is designed to minimize packet duplication, and receiver list re-writing in transit routers is eliminated.

Figure 2 shows an example where a sender S transmits to three receivers E_i with the aid of three MEADcast routers R_j. Note that the sender transmits unicast directly to E_1, as it is the only end-point on its subtree. It transmits one multicast packet to E_2 and E_3, to be transformed into unicast by router R_2.

None of the end-points can discern the identity of the others, thus preserving privacy, or if the data has been multicasted.

E. Synopsis

In the following Section II, we analyse the challenges to multicast adoption in dependency of the application being used. After describing a typical application scenario, we characterise applications with respect to their eligibility for multicast and discuss two example applications. Section III reviews different approaches to multicast, particularly Xcast [10], which is technically most related to our work, and juxtaposes their properties to those of our approach.

We continue by expounding the technical properties of MEADcast in Section IV by describing the behaviour of protocol entities, the structure of its protocol data unit and the API offered to applications. The description is complemented by a detailed example of the use of MEADcast. The study of the protocol's behaviour and performance, presented in Section V, indicates that the reduction in total volume may well be worth the effort for its introduction.

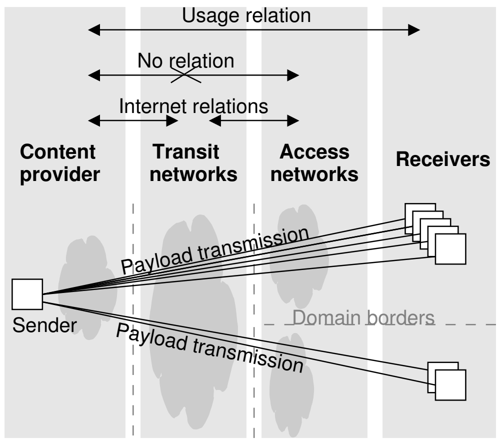


Figure 3. Relationships within the scenario.

In Section VI, we discuss the properties of MEADcast that address the challenges formulated in the Problem Analysis (Section II). In addition, the discussion addresses the protocol's overhead and limitations, security considerations and opportunities for optimization. Section VII summarizes our ideas and findings and points out further directions of research.

II. Problem Analysis

We analyse the problem of multicast adoption by characterising the applications that would benefit from it. The challenges faced by content providers are illustrated by means of a scenario. The analysis of applications' properties that influence their use of multicast are summarised in a problem space, which is exemplified by two different applications, one of them being the one from the scenario. The review in Section VI of the challenges identified in this section indicates how they are addressed by the MEADcast protocol proposed in this work.

A. *Scenario*

We illustrate the challenges to large-scale multicast use by means of the following scenario.

A media directory operates a website that lists freely accessible IP-radio and IP-TV offerings from different content providers, i.e., from Internet radio and TV "stations". The content streams are transmitted from the content providers to users in remote domains. Users that become increasingly aware of privacy issues are reluctant to use the service if other users become knowledgeable of their content consumption behaviour without their knowledge and consent. The privacy concept within our context is extensively discussed in the work "privacy terminology" [11].

Some content providers are willing to employ techniques like multicast in order to reduce the volume of data transmitted to the users. At the same time, some of the operators of networks hosting many receivers of the media streams are also interested in reducing the load within their own core and access network.

Both the media directory and the content providers are interested in reaching as many users as possible. Therefore, they are reluctant to create any barrier to the users' access of their content. The current and potential users are distributed globally over many domains, they have the skills to operate only basic, common software (e.g., web browsers) running on a diverse spectrum of types and versions of operating systems, thus obviating the effective introduction of a specialised client for receiving media streams.

Figure 3 illustrates the relationship between the content provider that hosts the sender, the transit networks and the networks hosting receivers. While there is a relationship of service delivery and service usage between the content providers and the users (brokered by the directory), this relationship is created ad-hoc based on interaction at the application layer. It does not extend to the network operators that host the users: the operators lack knowledge about the service being provided beyond what they may infer from observations of the network traffic entering their network. Content providers and network operators are associated with other networks by the relationships fundamental to the Internet, i.e., by peering and transit agreements. Such relationships are not necessarily between the two roles, and they are not transitive. Hence, in the general case, content provider and access network operator roles lack a direct relationship.

In consequence, it is difficult to establish a consensus for the use of multicast both technically due to the inter-domain aspect, and formally due to the lack of relationship between the content providers and the network operator hosting a user. There remains no other choice that is "safe" from preventing users' access to the offerings, than for the directory to broker unicast streams between the content providers and each of their users. This choice implies that a significant amount of the transmitted traffic will consist of identical payload.

B. *Problem space*

The scenario is exemplary of application scenarios that would benefit from multicast but do not use it. Such scenarios can be categorised by the properties of the applications arising from their purpose and their technical implementation:

- their communication pattern can be symmetric or asymmetric (i.e., m:n or 1:n communication),
- they can be operated within an administrative domain or across domains,
- their usage sessions can be of short or long duration,
- they can have a fixed or variable set of participants,
- they can have a small or large number of participants,
- they can have high or low demands on network resources,
- their capacity utilisation can be symmetric or asymmetric between participants, etc.

To be effective and efficient, the properties of the multicast scheme should be determined by the application category it serves.

The properties of the settings in which applications are deployed also characterise the scenario:

- network administration may be aware or unaware of the application,
- end-point users may or may not be allowed to self-configure group membership in a multicast group,

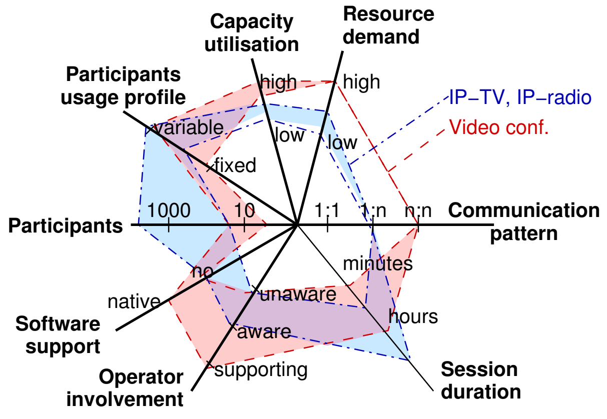


Figure 4. Application characterisation within a requirements space.

- the software package providing the application on the end-point side may or may not support group-based multicast.

C. Application examples

The resulting space is depicted in Figure 4, including two application scenarios described in the following, their usage requirements being shown as areas denoting their usage subspaces.

a) IP-radio and IP-TV: are part of a class of "broadcast" applications in that they replace traditional, e.g., terrestrial broadcast services. In analogy to the services it replaces, the application class exhibits a 1:n communication pattern and sessions that can reach durations of many hours, offered to a large, variable set of participants distributed throughout the Internet. The opportunity to employ buffering renders this application class insensitive to latency, jitter and drop rate and requires in principle only a sufficient end-to-end throughput according to the media transmitted. Network capacity is utilised in a highly asymmetric manner with a single sender producing all traffic once a session has been established.

b) Video conferencing: is an application with an n:n communication pattern of usage sessions in the range of a few minutes to a few hours between a small, variable set of participants that are grouped within a small number of domains. It has rather high requirements on quality of the network service due to being sensitive to latency, jitter, throughput and to some extent drop rate. The capacity utilised is symmetric, due to each participant sending his audio and video streams to all others. Hence, an application session creates a full-mesh network with traffic volume growing with the square of the number of participants. This may be the motivation for some operators to explicitly support multicast transport of conferences in order to reduce the peak throughput requirements on the network. Conferencing software typically supports the use of multicast addresses, however, group management, i.e., the creation of a group for a conference and the joining and leaving of multicast groups by participants, requires management information with respect to the mapping of conferences to multicast addresses. The participation of users outside the domain of the network operator cannot easily be supported from within the domain.

D. Challenges and opportunities

The challenges and opportunities in scenarios suitable for multicast can be summarised as follows.

- Both 1:n and m:n communication patterns offer a high degree of potential load reduction.
- The rather high volume requirements of pseudo-multicasted streams of media (e.g., audio, video) increase the potential reduction.
- The number of receivers of a particular content varies strongly, and receivers must be expected to enter and exit the group at any time.
- The duration of a session ranges between a few minutes and a few hours. If transmission is continuous, the potential savings in transmitted volume are significant.
- Inter-domain transmission lacks a direct relationship between content provider and access network domains on which a consensus for technology choice can be founded.
- Users cannot be expected to interact with non-trivial network functions, such as group management functions, unless explicit support by their local administration (i.e., the network operator) is provided. Likewise, the introduction of special software packages to support multicast on the users' side acts as a deterrent from using the service.
- The awareness to privacy issues with services provided in the Internet needs to be addressed such that the members of a communication group (e.g., a group simply receiving content in a 1:n communication pattern) may retain anonymity within the group.

Our construction of the MEADcast protocol, described in Section IV, aims to exploit these opportunities and to address the challenges in the scenario.

III. Related work

The idea of multicast was introduced decades ago and has drawn research efforts broadly. A variety of solutions have been proposed and a selection is presented here.

Traditional group-managed multicast [2], [3], [4], [12] specifies the transmission of an IP datagram to a *host group*, a set of zero or more hosts identified by a single IP destination address. It requires network support (multicast-capable routers) and the receivers to proactively join the host group. The routers and end-points use the Internet Group Management Protocol (for IPv4) or Multicast Listener Discovery (for IPv6) to maintain the multicast group. The deployment of IP multicast in the Internet is still far behind expectations due to a number of long-standing issues [13]. Amongst those are the management complexity put on the end-point and the requirement of global updates to routers.

An interesting approach on routing multicast traffic through a "multicast domain", namely Bit Index Explicit Replication (BIER) is presented in [7]. A multicast packet entering the domain is encapsulated in a BIER header by the ingress router, who then determines the set of egress routers to send the packets to them. Egress routers are represented by a bitstring in the BIER header. BIER simplifies the traditional multicast by eliminating the per-flow state and the explicit tree-building protocols with the trade-off of additional management layers introduced in BIER-compliant routers including the routing underlay, the BIER layer and the multicast flow overlay. A router has to maintain the routing information with other routers in a BIER domain besides the ordinary unicast one and be able to determine a set of egress routers for an incoming multicast packet, this indicates an extra state to be maintained. In comparison, a MEADcast router is much simpler by virtue of being totally stateless. Moreover, BIER limits its scope in routing multicast traffic within a BIER domain, leaving the inherent management issues of multicast untouched. In contrast, MEADcast concentrates all management initiative with the sender, allowing middle-boxes, non-MEADcast routers and receivers to remain agnostic of the use of multicast.

Common practice to overcome the traditional multicast management burden but still achieve better performance than ordinary unicast is to employ Application Layer Multicast (ALM), which implements multicasting functionality at the application layer instead of at the network layer by using the unicasting capability of the network. In contrast to the slow deployment of IP multicast, ALM gains practical success thanks to the ease of deployment. A survey of ALM over the period 1995-2005 was given in [14]. ALM's common approach is to establish an overlay topology of unicast links between the multicast participants where multicast trees can be constructed. The drawback of ALM is that the privacy of receivers is not ensured, which means the identity of one end-point might become known to the other; furthermore, the data delivery of ALM depends on the end-point capability, which could not guarantee the stability and reliability. MEADcast overcomes the above issues by leaving receivers agnostic of the underlying technology being used. ALM also requires, by principle, the deployment of ALM-capable software on at least a subset of the hosts participating in a multicast group.

Xcast [10] is a multicast scheme with explicit encoding of the destinations list in the data packets, instead of using a multicast group address. Xcast is able to support a very large number of small multicast sessions, making up the complementary scaling property to traditional IP multicast, as the latter has a scalability issue for a very large number of distinct multicast groups. Xcast transmits data along optimal route without traffic redundancy in the sufficient presence of Xcast-capable routers; otherwise, some special mechanisms have to be employed, e.g., tunneling or end-point upgrade, which introduce new management tasks for Xcast routers (exchanging and maintaining Xcast routing information) or end-points (performing network functions of an Xcast-router). In case an end-point assumes the Xcast router's functions due to the lack of network support, the identities of all receivers in a session are exposed to this one, raising the privacy issue and possibly hindering Xcast adoption by some applications. By design, an Xcast-router suffers from complex header processing: it has to perform a routing table lookup for each destination in the Xcast header's address list. Besides, Xcast is intended for multicast sessions of small number of participants (i.e., small group size), confining its suitable applications. Xcast6 Treemap [15], [16], a derivative of Xcast, while providing some enhancements to improve the traffic transmission latency in a use case of Xcast gradual deployment, still shares the same aforementioned limitations. However, Xcast does introduce many efficient features: no maintenance of multicast state by routers, routing based on ordinary unicast and automatic reaction to unicast reroutes, easy security and accounting, flexibility, among others. MEADcast takes advantage of some valuable concepts from Xcast while off-loading the management burden to the sender and simplifying the router functions.

IV. Protocol design

MEADcast is implemented by senders and routers. We describe the functions relevant for sender and router elements and message types and procedures necessary for the realization of these functions. A simple multicast scenario described in full illustrates the behaviour of the protocol.

The information needed to describe the protocol is

- the sender S,
- the set of end-points E_i to which S transmits data,
- the set of MEADcast routers R_j in the network,
- the MEADcast distance d (or MEADcast hop-count) from a MEADcast router to the sender in hops between MEADcast routers.

Association is indicated by superscript, i.e., a router responsible for a sub-set E_k of the end-points is R^k and an end-point served by a router R_j is E^j.

A. Functions

In MEADcast, we need to distinguish two groups of functions for the sender and the router.

Sender functions include transmission of *unicast* and *multicast* messages, *initiation of discovery* of MEADcast routers on paths to end-points and *discovery response evaluation*.

Router functions include normal *forwarding*, *decomposition* of multicast packets to unicast packets and multicast packets and *reaction to discovery requests* from a sender.

1) Discovery-related functions: Both the sender and the MEADcast router are involved in the discovery process. The goal of discovery is for the sender to determine the sequence of routers $(R_1^i, R_2^i, \cdots)$ on the path to each end-point E_i. Discovery requests and responses can be written as *req(E, d)* and *resp(E, d, R)*, respectively.

To initiate discovery, the sender addresses a MEADcast discovery request *req(E, 0)* to an end-point E. When receiving the request, every router R on the path to E increments d and forwards the discovery request *req(E, d+1)* to the next hop; at the same time, R sends a discovery response *resp(E, d+1, R)* to S. Thus, the first router R_1 on the path to E will send $(E, 1, R_1)$, the second $(E, 2, R_2)$ and so on.

S can compile the sequence $\{(E_i, d_1, R_1^i, d_2, R_2^i, \cdots), \cdots\}$ and can compute the group of end-points to be handled by a given router with a specific distance $(R_j, d_j, E_1^j, E_2^j, \cdots)$.

2) Decomposition: Decomposition, which is specific to MEADcast router, means the transformation of a multicast packet addressed to a set of target end-points into multiple unicast and multicast packets with the same payload.

The target addresses $R_j, E_1^j, E_2^j, \cdots, R_k, E_1^k, E_2^k, \cdots$ within a multicast message are structured to denote that a router R_i is responsible for end-points E^i. During decomposition a router will send unicast packets to each of

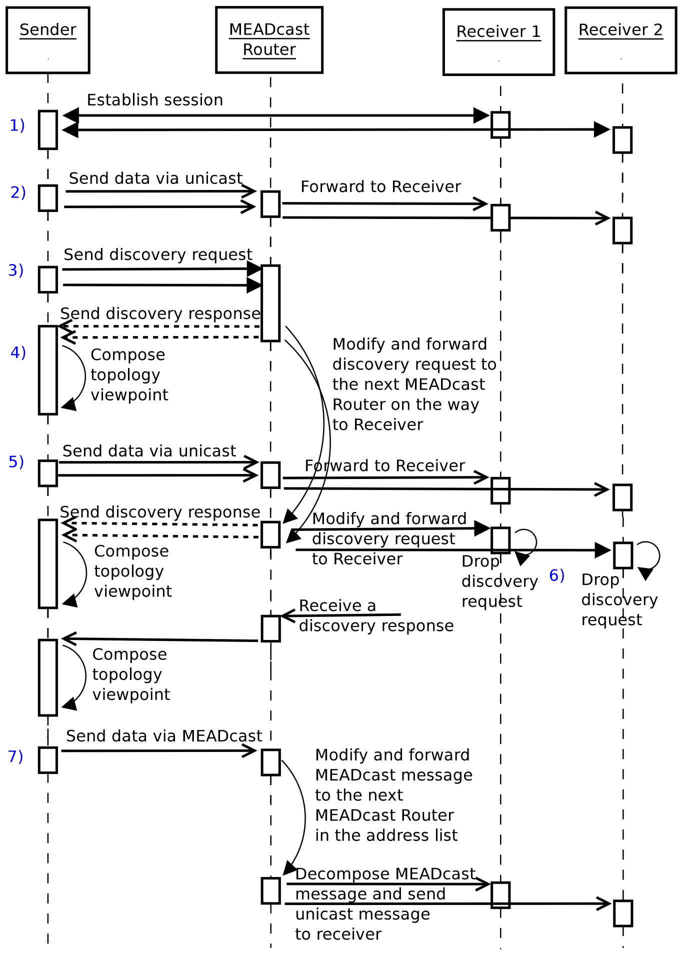

Sender
MEADcast Router
Receiver 1
Receiver 2
1)
Establish session
2)
Send data via unicast
Forward to Receiver
3)
Send discovery request
Send discovery response
4)
Compose topology viewpoint
Modify and forward discovery request to the next MEADcast Router on the way to Receiver
5)
Send data via unicast
Forward to Receiver
Send discovery response
Modify and forward discovery request to Receiver
Compose topology viewpoint
Drop discovery request
6)
Drop discovery request
Receive a discovery response
Compose topology viewpoint
7)
Send data via MEADcast
Modify and forward MEADcast message to the next MEADcast Router in the address list
Decompose MEADcast message and send unicast message to receiver

Figure 5. Interaction between MEADcast's entities.

the end-points it is responsible for and send multicast packets to the routers responsible for the remaining target end-points.

When a multicast packet is created, the targets already served either by unicast or by other multicast messages are removed from the list of targets of the packet being created. The removal process can be implemented efficiently by marking removal in a bitmap and thus eliminating the need to compose a new list of targets.

B. Interaction between MEADcast's entities

Figure 5 shows the interaction between the sender, the MEADcast router and the receivers, which involves the following steps.

1) First, the session is established between the sender and the receivers. In essence, this could be the request on data sent from each receiver to the sender or a data push from sender to some pre-defined targets. The double-headed arrow is used in this step to indicate both possibilities of a session initiator (sender or receivers).
2) The sender starts transmitting data to the receivers via unicast. Each router on the path takes part in the data transmission process as usual.
3) In the meantime, the sender also performs the MEADcast discovery by sending *MEADcast discovery request* to each receiver. Upon receiving a request message, the MEADcast router sends a *MEADcast discovery response* back to the sender and a *MEADcast discovery request* towards the receiver. The *MEADcast hop-count* in each message is modified indicating the *MEADcast distance* from the current router to the sender.
4) On receiving a *MEADcast discovery response*, the sender will compose its topology view, which is an overlay network connecting the sender, the MEADcast routers and the receivers.
5) The data transmission by unicast is still carried out during the MEADcast discovery phase.
6) The receivers drop *MEADcast discovery request* messages since they do not understand them.
7) After a pre-defined timeout, the sender stops transmitting data via unicast and starts the *MEADcast data sending* phase. MEADcast data messages are built based on the current topology viewpoint of the sender and are then transmitted. Each MEADcast router on the path processes the message according to the encoded MEADcast information. It may forward the message intact or decompose the message into multiple ones and forward them to the other MEADcast routers, or build and send unicast messages to the receivers.

C. Sender behaviour

The sender behaviour involves two phases: *MEADcast discovery* and *MEADcast data sending*.

The sender sends *MEADcast discovery requests req*$(E_i, 0)$ to all receivers and updates the network topology in the form of $(R_j, d_j, E_1^j, E_2^j, \cdots)$ whenever it receives a *MEADcast discovery response*. In the mean time, the sender also transmits unicast data to these end-points.

The *MEADcast data sending* phase starts when the discovery phase is complete (i.e., after a pre-defined timeout). Based on its network topology view, the sender constructs and transmits *MEADcast data messages* containing the target addresses $(R_j, E_1^j, E_2^j, \cdots, R_k, E_1^k, E_2^k, \cdots)$ for those receivers that can be served by MEADcast routers and stops unicast data to them. If the set of receivers of a multicast is larger than the maximum number of addresses encodable in the MEADcast header, the sender will divide the receivers into suitable subgroups, each served by its own MEADcast packet. If discovery reveals that a given MEADcast router would only be responsible for a single receiver, that receiver is served unicast in order to conserve header space in MEADcast transmission.

It is obvious that if there is no MEADcast router responsible for any receivers, the data sending phase of MEADcast operates exactly as unicast.

The discovery phase is carried out periodically so that the sender can maintain an updated view of the topology.

D. Router behaviour

A MEADcast router is an IP router with added control plane behaviour. It is capable to fulfil two tasks: participate in the discovery mechanism and process MEADcast packets that transport payload. The router does not hold state with respect to the topology, the presence or the location of other routers supporting MEADcast; nor does it hold information about the multicast groups.

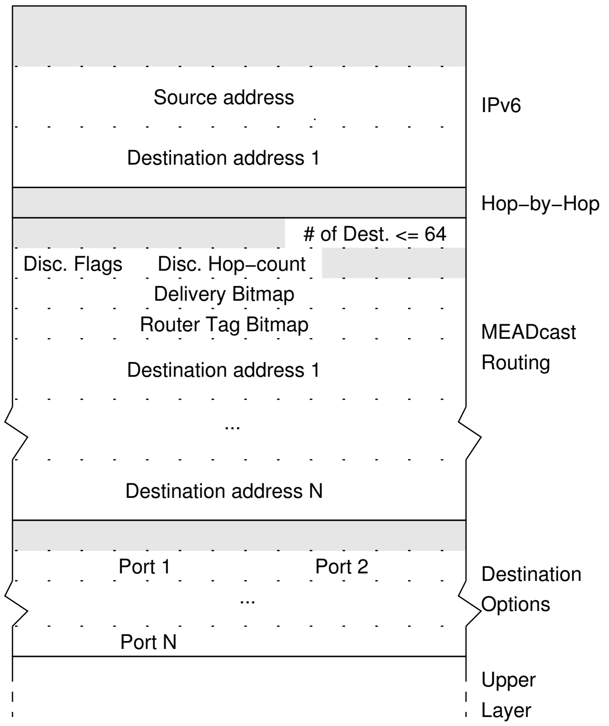
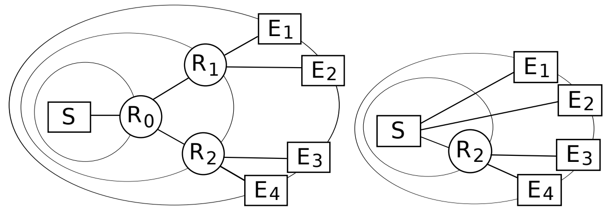


Figure 6. MEADcast header sequence with relevant fields.

1) The discovery mechanism: Discovery requests pertain to a path to one receiver. They are therefore addressed to that receiver and have the source address of the sender performing discovery. Upon receiving such a request, the router increments the MEADcast hop-count field of the request packet before processing (and forwarding) it as any other IP packet. In addition, the router transmits a discovery response to the sender, transmitting its own address, the hop-count of the discovery request and the address of the receiver that the discovery is for. In consequence, the sender is capable of discerning the sequence of MEADcast-capable routers on the path to a given receiver, as it is able to order the routers by the MEADcast hop-count of their responses.

2) The processing of multicast packets: relies on the address information transported in the MEADcast header. This information consists of a list containing both the addresses of receivers and those of the MEADcast routers expected to process the packet. A static *router tag bitmap* within the header differentiates between the addresses belonging to receivers and those belonging to routers. An additional bitmap, the *delivery bitmap*, is employed to mark the addresses already accounted for within the multicast tree.

On receiving a MEADcast packet, the router determines with the help of the router bitmap the addresses that it is responsible for. For the addresses of receivers (e.g., those in directly connected networks) it creates unicast packets and introduces them into its output queues. For the addresses of routers that are expected to perform additional distribution, it duplicates the received packet. In the duplicates, it marks all the addresses that it has accounted for in the delivery bitmap. Thus, the next router in the chain can determine that it should neither send unicast to the receivers at those destinations nor create multicast packets for the routers among them.



(a) Full network support. (b) Partial network support.

Figure 7. Network topology from sender viewpoint.

E. Protocol headers

The tasks of sender and router are supported by a number of data structures beyond those of the IP header. MEADcast for IPv6 uses the extension header mechanism to introduce multicast information. Figure 6 shows an overview of the sequence of headers employed in a MEADcast transmission with standard fields removed for clarity. The whole header includes the standard IPv6 header, Hop-by-Hop Options header, MEADcast Routing header and Destination Options header.

The fields depicted in the figure have the following purpose and structure:

- Source address and Destination address $1 \cdots n$ are normal IPv6 addresses.
- $\#$ of Dest: Number of destinations in IPv6 address encoded in the MEADcast Routing header.
- Disc. Flags: Discovery flags to mark a MEADcast message as discovery request, response or data delivery.
- Disc. Hop-count: Discovery hop-count, which is the MEADcast distance from a MEADcast router to the sender.
- Delivery Bitmap: to mark if a MEADcast router has received the MEADcast message.
- Router Tag Bitmap: to mark the position of the MEADcast routers in the MEADcast Routing header's address list.
- Port $1 \cdots n$: transport protocol port number (specifically UDP port number) for receivers.

F. Protocol mechanics by example

Figure 7 describes the network where the proposed scheme is effective, consisting of five end-points S, E_1, E_2, E_3, E_4 and three routers R_0, R_1, R_2. Their connections are shown in Figure 7(a) (without the rings). Two scenarios are described, the first one with all routers being MEADcast-capable, the second one with only R_2 being a MEADcast router.

1) All routers are MEADcast-capable: The communications between the sender S and the receivers E_1, E_2, E_3 and E_4 via the network in the first scenario are as follows.

1) S transmits unicast data to E_1, E_2, E_3, E_4.
2) S sends four different *MEADcast discovery requests req($E_i, 0$), $i \in \{1, 2, 3, 4\}$.*
3) For unicast messages, R_0, R_1, R_2 simply forward it to the intended receiver.
4) R_0 receives *req($E_1, 0$)*, it reacts to the presence of the Hop-by-Hop header and analyses the content of the MEADcast header. R_0 sends a *MEADcast discovery response resp(E_1, 1, R_0)* to S. It also sends *req($E_1, 1$)*

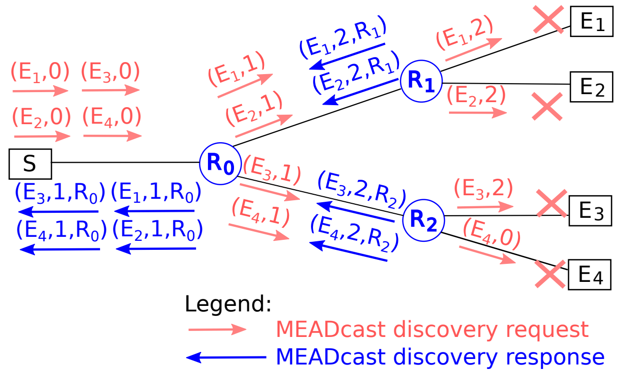


Figure 8. Discovery in all MEADcast network (case a).

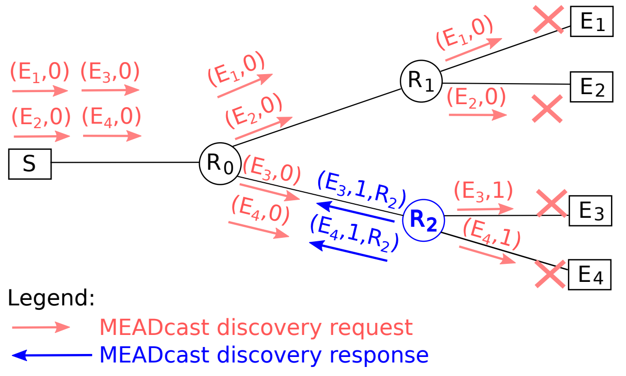


Figure 10. Discovery in sparse MEADcast network (case b).

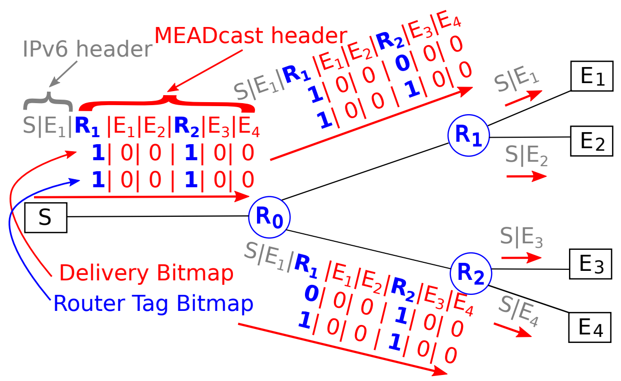


Figure 9. Data delivery in all MEADcast network (case a).

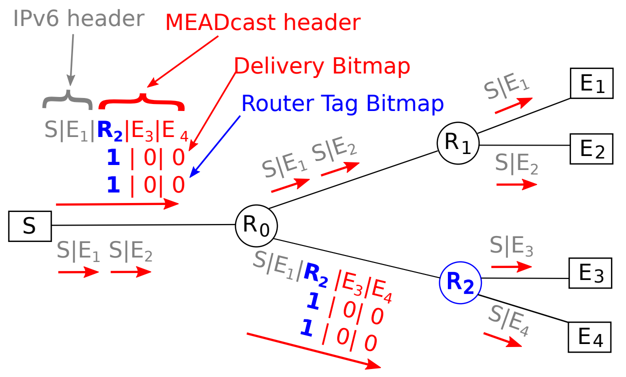


Figure 11. Data delivery in sparse MEADcast network (case b).

to E_1. The same procedure is carried out for *req(*$E_2, 0$*)*, *req(*$E_3, 0$*)*, *req(*$E_4, 0$*)*.

5) R_1 receives *req(*$E_1, 1$*)*, it sends *resp(*E_1*, 2,* R_1*)* to S. It also sends *req(*$E_1, 2$*)* to E_1. The same procedure is carried out when R_1 receives *req(*$E_2, 1$*)*.
6) R_2 receives *req(*$E_3, 1$*)* and *req(*$E_4, 1$*)*, it sends *resp(*E_3*, 2,* R_2*)* and *resp(*E_4*, 2,* R_2*)* to S. It also sends *req(*$E_3, 2$*)* to E_3 and *req(*$E_4, 2$*)* to E_4.
7) E_1, E_2, E_3, E_4 receive the unicast messages normally. For the *MEADcast discovery request*, they do not understand and simply drop it.
8) S receives *MEADcast discovery responses* and updates its network topology viewpoint as $(R_0, 1, E_1, E_2, E_3, E_4), (R_1, 2, E_1, E_2), (R_2, 2, E_3, E_4)$. The topology viewpoint of sender can be illustrated by the rings in Figure 7(a), where the sender is at the center, R_0 lies on the first ring with a distance of one, R_1 and R_2 are on the second ring with a distance of two and all receivers are always at the outermost ring.

All the steps above are depicted in Figure 8. The corresponding data delivery phase described in the next steps is shown in Figure 9. MEADcast routers and their associated entries in bitmap fields of MEADcast headers are marked in blue bold text. The same marking scheme is used in Figures 10 and 11.

9) S stops transmitting data via unicast and starts MEADcast data sending phase. S transmits a *MEADcast data message* consisting of:
   - E_1 as the destination IP address,
   - $\{R_1, E_1, E_2, R_2, E_3, E_4\}$ in the *MEADcast Routing header*'s address list,
   - *Router Tag Bitmap* being 100100, where bit 1 indicates a router and 0 an end-point, *Delivery Bitmap* is initialized with the same value of *Router Tag Bitmap*. Note that, *Router Tag Bitmap* also points out, which MEADcast router is responsible for which end-point, in this case: R_1 is responsible for E_1, E_2 and R_2 for E_3, E_4. *Router Tag Bitmap* of the *MEADcast data message* stays unchanged after leaving the sender.
10) R_0 receives the *MEADcast data message*, sees that:
    - it does not have to deliver message to any receivers since its address is not in the MEADcast address list,
    - based on the *Router Tag Bitmap* and *Delivery Bitmap* fields, there are two other MEADcast routers, namely R_1, R_2, needing to receive *MEADcast data message*. R_0 duplicates the original *MEADcast data message*.
      - The *Delivery Bitmap* field of the first one is modified to be 100000, indicating that R_2 has received a *MEADcast data message*. R_0 sends this message to R_1.
      - R_0 changes the destination IP address of the second message to E_3, modifies the *Delivery Bitmap* field to be 000100, indicating that R_1 has received a *MEADcast data message* and sends it to R_2.
      - The checksum at the transport layer (specifically, UDP) of each message is changed accordingly and is discussed further in Section VI-B.
      - The *Router Tag Bitmap* fields in both messages are the same as in original one.
11) R_1 receives a *MEADcast data message*, reads the *Router Tag Bitmap* field and sees that it is responsible for E_1 and E_2. R_1 constructs two unicast messages with the

data from the *MEADcast data message* and transmits each to E_1 and E_2. The checksum at the transport layer of each message is changed accordingly. The *Delivery Bitmap* value of 100000 shows that there is no other MEADcast router who needs to receives this *MEADcast data message*.

12) R_2 receives a *MEADcast data message*, reads the *Router Tag Bitmap* field and sees that it is responsible for E_3 and E_4. R_2 constructs two unicast messages with the data from the *MEADcast data message* and transmits each to E_3 and E_4. The checksum at the transport layer of each message is changed accordingly. The *Delivery Bitmap* value of 000100 shows that there is no other MEADcast router who needs to receives this *MEADcast data message*.

2) Only R_2 is MEADcast-capable: The communications between the sender S and the receivers E_1, E_2, E_3 and E_4 via the network in the second scenario (only R_2 is a MEADcast router) are sketched in Figures 10 and 11, which have the same first three steps as in the first scenario. The further steps are as follows.

1) R_0 receives *req($E_1, 0$)*, it reacts to the presence of the Hop-by-Hop header and analyses the content of the MEADcast header, which it does not understand. It forwards the message further to the E_1 direction. R_0 does not drop the message since the option type identifier of MEADcast header is 00 [17]. The same procedure is performed for *req($E_2, 0$), req($E_3, 0$), req($E_4, 0$)*.
2) Similarly, R_1 receives *req($E_1, 0$)* and *req($E_2, 0$)*, it sends these messages to E_1 and E_2.
3) R_2 receives *req($E_3, 0$), req($E_4, 0$)*. It sends *resp(E_3, 1, R_2)* and *resp(E_4, 1, R_2)* to S. It also sends *req($E_3, 1$)* to E_3 and *req($E_4, 1$)* to E_4.
4) E_1, E_2, E_3, E_4 receive the unicast messages normally. For the *MEADcast discovery requests*, they do not understand and simply drop them.
5) S receives *MEADcast discovery responses*, updates its network topology viewpoint as $(R_2, 1, E_3, E_4)$. Its network topology viewpoint is illustrated in Figure 7(b). There is no MEADcast router on the paths to E_1, E_2, only R_2 lying on the first ring of distance one is on the paths to E_3, E_4. All receivers are on the outermost ring.
6) S starts MEADcast data sending phase. Based on its topology view, S sees that:
   - there is no MEADcast router on the paths to E_1, E_2. S unicasts data to these receivers.
   - R_2 is on the paths to E_3, E_4. S transmits a *MEADcast data message* with E_3 as the destination IP address and $\{R_2, E_3, E_4\}$ in the *MEADcast routing* header's address list, *Router Tag Bitmap* and *Delivery Bitmap* being 100.
7) For unicast messages, R_0, R_1 simply forward them to the intended receiver.
8) R_0 receives the *MEADcast data message*, which it does not understand. It forwards the message further to the E_3 direction. R_0 does not drop the message since the option type identifier of MEADcast header is 00 [17].
9) R_2 receives a *MEADcast data message*, reads the *Router Tag Bitmap* field and sees that it is responsible for E_3 and E_4. R_2 constructs two unicast messages with the data from the *MEADcast data message* and transmits each to E_3 and E_4. The checksum at the transport layer of each message is changed accordingly.

G. Sender API

All state information about the multicast tree is held by the sender, as MEADcast receivers are multicast-agnostic by design. Routers with MEADcast-capability need not keep state about the multicasted flows but require only the addition of handling the discovery mechanism of MEADcast and the processing of MEADcast data packets.

The sender has special requirements on the API by which the network socket is controlled. We build on the work described in RFC 7046 [18] ("Hybrid Adaptive Multicast", HAM; "Transparent Hybrid Multicast") as a generic API that aims to provide a common vertical interface for multicast sockets without regard to the multicast technology. Although the HAM API does not specifically target the sender-centric multicast schemes and does require extensions to fully support them, the expressive power of the API is sufficient to control most aspects of the sender and does not contain aspects that are unimplementable for MEADcast.

The HAM model of multicast is that an application uses a HAM socket, which can be bound to network interfaces and multicast groups to support an n:m communication pattern, as known from "traditional" multicast. The application effects the life-cycle management of the socket, its binding to network interfaces and the joining and leaving of multicast groups. Multicast group identifiers are separated from network addresses and noted as a URI (Uniform Resource Identifier) in order to avoid assumptions about the actual multicast technology being employed. As a tacit assumption, every host supporting the API is expected to manage its membership in a group: it is this tacit assumption that will require extension of the HAM API in order to support MEADcast.

We iterate through the four interface sections of the HAM API and apply them to MEADcast, before specifying the necessary extensions.

1) Group management: The `create()` and `delete()` calls act on abstract Multicast Sockets as described in the RFC.

The group management functions `join()` and `leave()` are strictly speaking not required for MEADcast, since group membership is only managed by the sender. They are restricted to enabling MEADcast to change a group's denomination during a session.

The calls for (de-)registering a multicast data source, `srcRegister(), srcDeregister()` duplicate the function of the socket creation and deletion functions for the purposes of the 1:n MEADcast.

2) Send and receive: The `send()` function signature can be applied as is. The `receive()` function is inert, given that the sender does not receive multicast and the receivers do not support the API.

3) Socket options: RFC 7046 specifies a number of socket management functions, which are useful for a sender application and can be supported in a MEADcast sender. The functions allow the management of interfaces bound to a socket with `getInterfaces(), addInterface()` and `delInterface()` and setting of network parameters for sockets/interfaces with `setTTL(), getTTL`.

The function `getAtomicMsgSize()` *"returns the maximum message size that an application is allowed to transmit per socket at once without fragmentation"*. Given that MEADcast performs a trade-off between this "atomic size" and the number of multicast targets addressable with a single PDU (Protocol Data Unit), applications should always consult this function to determine the application SDU (Service Data Unit) size offered by the MEADcast trade-off.

If an application cannot practically reduce the amount of data sent at once, it is useful to allow it to influence the trade-off performed by the MEADcast socket, i.e., to sacrifice the number of addressable targets in favour of larger messages, e.g., in order to fit frames of media streams into one message. We present an extension to support such hints later in Section IV-G5.

4) Service Calls: The service calls of the HAM API allow an application to retrieve information about the multicast environment of a HAM node.

The most relevant is the function `childrenSet()`, which returns the clients (receivers) served by this sender, queried by interface. For a complete list of receivers, the list of interfaces retrieved with `getInterfaces()` can be iterated through calls of `childrenSet()`.

The maximum message size transmittable (with possible fragmentation) over a socket can be queried with `getMaxMsgSize()`. For MEADcast, the value returned can be the maximum size transmittable by unicasted IP; an application sending messages of that size would either force transmission by unicast or the multicast of fragments.

Some service calls return information possibly useful to the application: `groupSet()` would retrieve the groups mapped to a given interface; the application can request the neighbour nodes (assumed to be those on the same link layer segment) by means of calling `neighborSet()` for an interface.

Some of the service calls lack a function for MEADcast but can be implemented to return sensible values for the application. The `designatedHost()` function determines *"whether this host has the role of a designated forwarder (or querier), or not."*. The `parentSet()` function returns a list of neighbours, from the node (in our case, the sender) receiving multicast; a MEADcast implementation can return an empty set.

Membership events are issued from a HAM socket instance *to* the application to notify of *other* members of the multicast group joining or leaving the group. In our case, that information originates with the application. Hence, supplying it again can be useful only for confirmation of changes to the group made within the MEADcast layer. Event flow management with `enableEvents()` and `disableEvents` can be implemented as specified.

5) Extension: The HAM API document does mention, in short, Xcast as an example for a sender-centric, agnostic-receiver-multicast protocol, and it sketches a few ideas for necessary information management for this class of protocols. The API itself, however, fails to take into account that Xcast as well as MEADcast require a means to inform a sender about the introduction or removal of a receiver to/from a group. Consequently, it is necessary to supply two fundamental functions in addition to those specified in RFC 7046.

a) Group management API extension: For the extension, we employ the same terminology and language as RFC 7046. In particular, the receivers of packets multicasted from one sender are termed *"children"* with respect to that sender.

The application can register or deregister a child at a socket. These functions are mandatory, as they constitute the only manner in which MEADcast can be made aware of status changes in children, i.e., in receivers.

```
joinChild (in Uri groupName,
           in Child childSpec,
           out Int error);
```

and

```
leaveChild (in Uri groupName,
            in Child childSpec,
            out Int error);
```

In each function, `groupName` identifies the multicast group that the receiver ("child") is to be added to. The implementation is responsible for adding it to the appropriate socket or sockets and, if enabled, to generate Events to this effect. An `error` return value of 0 indicates success, one of -1 indicates failure.

The `Child` structure contains a mandatory set of information items pertaining to the basic multicast service and a set of optional items to support optimisation or application-specific parameters. It is given as:

```
Child { Int IPversion ;
        IPAddress childIPAddress ;
        IPAddress routerIPAddress ;
        Int port ;
        Int pathMTU ; }
```

The `IPversion` mandatory field identifies the Internet Protocol version to be used when casting to this receiver. The `IPAddress` type is a binary encoded IPv6 or IPv4 (according to the value of `IPversion`) unicast address. The mandatory field `childIPAddress` identifies the receiver by IP address.

The optional field `routerIPAddress` identifies the initial router (if any) to be responsible for transmitting unicast to the receiver, giving its IP address on the path between sender and receiver. The optional field `pathMTU` contains the maximum transfer unit (MTU) on that same path. The optional field `port` carries the transport layer protocol port at the receiver.

The structure is intended to be extensible with respect to the optional items. Conceivable options include the receiver's preferences regarding quality, transmission volume, as well as accounting parameters.

b) Hints: It is necessary to control the trade-off between the number of targets addressable in a single MEADcast PDU and the maximum size of messages acceptable from upper layers and applications while avoiding fragmentation. While the MEADcast stack might strive to maximise the number of

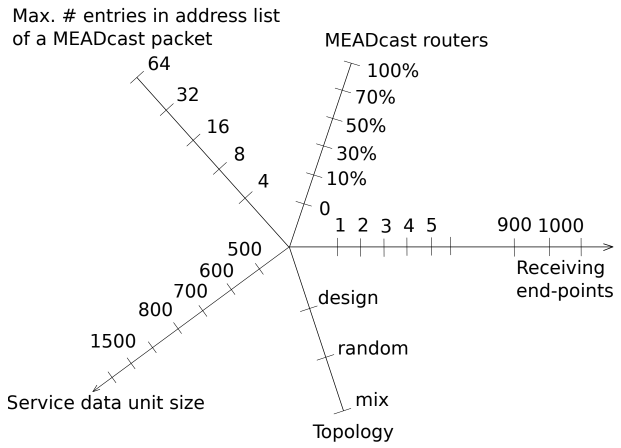


Figure 12. Parameters for experiments.

addressable targets, application-specific information regarding the structure of data can be helpful. For that purpose, we extend the API with an additional, optional function:

```
setExpectedSDUSize (in Int sduSize,
                    in Uri groupName,
                    out Int error);
```

The parameter `sduSize` is the size of the service data unit (SDU) of the upper layer, in bytes, that the application expects to `send()` to a group specified by `groupName`. The function returns a value of zero, if the size can be transmitted without fragmentation or -1 if the size cannot be transmitted.

Note that the SDU will encompass the payload and the transport protocol header: it is up to the application programmer, who decides on the transport protocol to be used, to calculate the value.

For example, an application transmitting a media stream can set the expected size of the payload to be the size of one or more media frames plus the size of an UDP header. In response, the MEADcast implementation can determine the maximum number of addresses included in its own header.

V. Evaluation

We have performed experiments within the parameter space illustrated in Figure 12. We simulate MEADcast for 100 routers both on random network topologies with a diameter of 16 (generated by GT-ITM [19]) and on topologies designed according to realistic scenarios using ns-2 [1]. To be more specific, the experiments are carried out by the following steps.

1) A core network of 100 routers is generated.
2) Three core networks are created by randomly enabling the MEADcast capability on 30, 50, 70 routers of the core network in step 1, respectively. One more designed core network by selectively enabling the MEADcast capability on 10 routers and another for all routers are added. In total, there are six core networks associated with 0, 10, 30, 50, 70 and 100 MEADcast-capable routers, in which the second one is intended for designed cases and the others for random ones.
3) The end-points are added to the core networks in step 2. The number of end-points ranges in 100, 200,...,1000. We actively distribute the end-points over MEADcast-capable routers for design cases while they are scattered arbitrarily in random cases. Each association of a chosen end-point set with a core network in either case forms a final topology, which means there are 50 random final topologies and 10 designed ones.
4) For each final topology in step 3, two experiments, one for unicast and another for MEADcast, are performed. An end-point is chosen as sender and the remaining end-points are receivers. The sender then transmits a data stream of 800 MB into the network to all the receivers. The trace file created by *ns-2* for each experiment to log the whole data transmission process is analysed. The traffic volume on all links of the whole network is calculated and logged for further analysis. It is 120 different experiments on the whole.

[1]https://www.isi.edu/nsnam/ns/

A screenshot of a random topology with 100 routers (red square nodes) and 100 end-points (grey round leaf nodes) is presented in Figure 13. Node 100 (leftmost leaf node) is chosen as sender, the remaining leaf nodes are receivers.

Table I shows an excerpt of the whole result when experimenting MEADcast and unicast for 100, 500 and 1000 end-points. Their corresponding total data volume in this table is plotted in Figure 14 whereas Figure 15 presents all the results. The percentage of traffic saving between MEADcast and unicast for the case of 1000 end-points is highlighted by double-headed arrows in the latter figure.

If there is no MEADcast router, the sender sends mainly unicast messages and periodically sends discovery messages that occupy only a little traffic volume over the whole network. This discovery overhead is indicated by the value "$-0.x$" in Table I and depends on how many times the discovery is performed. Hence, the traffic volume of the MEADcast protocol when there is no MEADcast router is approximately that of unicast, provided that sender has large traffic to send. The gap increases when the number of receivers and the percentage of MEADcast routers grow. The extreme case

Table I. Total traffic volume in the whole network [MB].
* indicates designed topology.

Topology		*Unicast*	*MEADcast without discovery*	*Discovery (one time)*	*Traffic saving [%]*
End-points	*MEADcast routers*				
100	0	761,504	761,504	0.075	0
	10(*)	761,504	271,104	0.149	64,4
	30	761,504	563,376	0.157	26
	50	761,504	433,504	0.210	43.1
	70	761,504	372,964	0.246	51
	100	761,504	272,420	0.500	64.2
500	0	4,136,544	4,136,544	0.409	$(-0.x)$
	10(*)	4,136,544	1,279,936	0.813	69.1
	30	4,136,544	2,513,296	0.897	39.2
	50	4,136,544	1,698,336	1.216	58.9
	70	4,136,544	1,069,276	1.389	74.2
	100	4,136,544	902,056	2.835	78.2
1000	0	8,341,776	8,341,776	0.826	$(-0.x)$
	10(*)	8,341,776	2,525,904	1.640	69.7
	30	8,341,776	4,978,384	1.802	40.3
	50	8,341,776	3,137,972	2.462	62.4
	70	8,341,776	1,866,456	2.819	77.6
	100	8,341,776	1,552,304	5.727	81.4

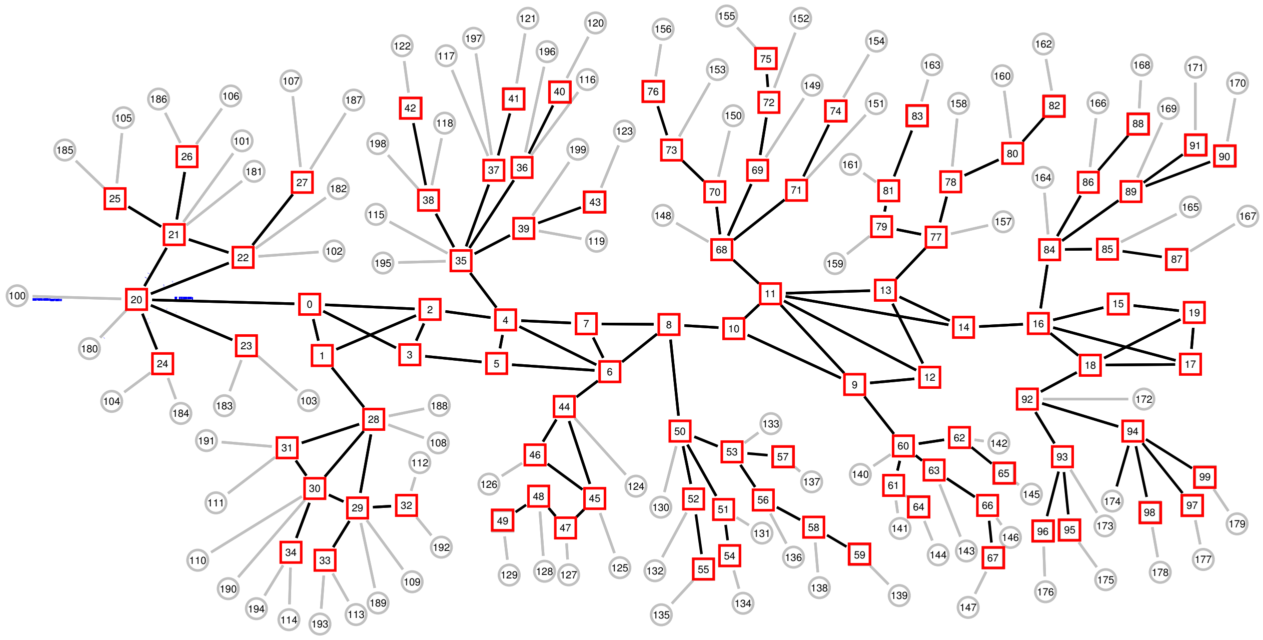


Figure 13. A network topology generated by GT-ITM.

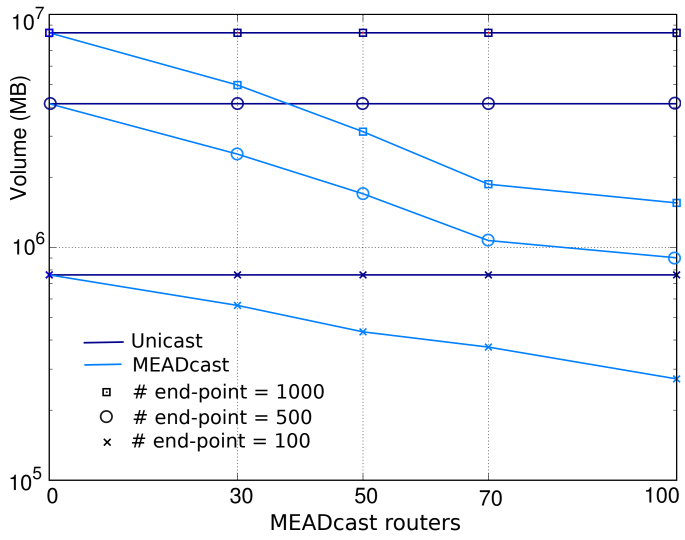


Figure 14. Total data volume of unicast and MEADcast - an excerpt.

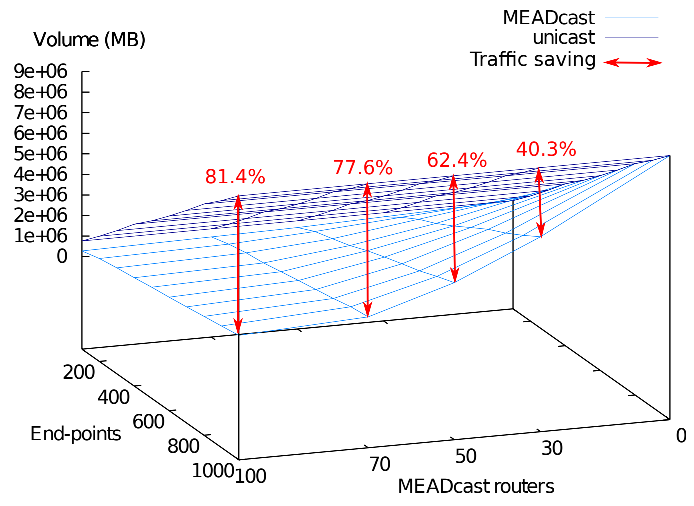


Figure 15. Total data volume of unicast and MEADcast when varying MEADcast support in the network and the number of receivers.

of 1000 end-points and 100% MEADcast routers shows the difference of 81.4% in total traffic volume.

The total traffic volume reduction is considerable in the presence of sufficient MEADcast routers, as shown by the designed cases. The link stress (i.e., the number of packets with the same payload sent by a protocol over each underlying link in the network) [20] at the sender is reduced to an even higher degree.

The impact of the service data unit size and the number of entries in the address list are discussed in Section VI.

VI. Discussion

We discuss a selection of aspects of MEADcast pertinent to its deployment, the relationship of the protocol implementation to other architectural components within the sender software stack, as well as limitations and the treatment of anomalies that may occur.

A. *Addressing scenario challenges*

The scenario (see Section II) describes challenges originating in organisational and technological distance between content providers in the sender role and the network operators hosting the receivers.

MEADcast allows the gradual shift from many unicast flows to a reduced number of multicast flows. Due to the protocol's properties, the receivers are not made aware of the use of the protocol or the manner in which it transports a flow to them. Certainly, the interception of discovery packets addressed to receivers allows them to actively detect the sender's ability to employ MEADcast but without discerning with certainty

- if multicast is actually in use for “their” flow, and
- which other addresses receive the same data stream.

The privacy of each receiver is thus preserved.

The content provider can deploy MEADcast within their own network in order to reduce the load between the senders and the MEADcast routers. The operators of neighboring networks to which the content provider organisation maintains contractual relationships can be informed of the option to deploy the protocol in order to improve distribution and thus reduce the transmission volume.

Our approach specifically targets the subspace of applications delivered via massive unicast into networks in different administrative domains than the sender. While MEADcast can be employed as a general-purpose multicast protocol, its benefits are most apparent in these scenarios: without an approach with native internetwork support and incremental deployment features it does not seem plausible that multicast would be deployed at all. In the special cases where a managed service is provided locally (e.g., in a symmetric conference-style application provided to participants within the same administrative domain) and pre-configured by that domain's network management, traditional, group-based multicast may be the more effective choice.

1) Data-driven deployment: MEADcast signalling, i.e., the discovery subprotocol, creates a technical means to advertise the technology being in use and at the same time to identify the number and location of receivers within a network to the network operator.

By observing MEADcast discovery traffic, a network operator is capable of determining

- the amount of traffic with potential MEADcast support
- whether the traffic terminates within the operator's network or constitutes transit traffic,
- the distribution of the terminating traffic within the network topology, and
- the distribution of the transit traffic onto egress points.

Based on this information, the network operator is capable to gauge the savings in terms of traffic volume if MEADcast capability is introduced at a certain point in the network.

Consequently, the mechanism allows MEADcast routers to be deployed at points where they have the greatest benefit, while maintaining the freedom from association between a content provider sending multicastable data and a network operator hosting multiple receivers.

2) Incentives for non-access networks:

a) Incentives for peer and transit networks: to support the protocol may originate in their own capacity management, in order to reduce volume destined to their own egress routers.

b) Carrier services: may not be motivated to support MEADcast, depending on their usage accounting model. However, the networks they serve might request MEADcast support as a service, when acting as transit networks to the MEADcast traffic or as the hosting network for many receivers.

3) Deploying m:n multicast applications: Applications like video conferencing (see Section II-C have an m:n (in their special case $m = n$) communication pattern. They can benefit from MEADcast by deploying the sender software stack on participants' hosts, thus placing those hosts in the role of a MEADcast sender.

This type of deployment has a number of beneficial properties:

- The sender stack is optional: if it is missing or inoperable on a given host, the outgoing traffic from that host is regular unicast.
- The support of the networks connecting the participants is employed to the degree that it is actually available, but network operators may decide to deploy MEADcast capability in order to manage the network load produced by video conference sessions.
- No mapping of participants' network identities to multicast groups is necessary, therefore allowing a conference to be configured by selecting, e.g., SIP identities.
- Finally, the extension software stack can be pre-configured by the administrative domain's system manager without regard to the location of other participants and without the need for user configuration.

B. Relation to upper layers

The concepts of MEADcast require a higher degree of interaction between the network layer and its upper layers (transport and application), which merits discussion. While our simulation results indicate significant performance gains for a wide range of parameters, MEADcast scenarios may be limited by properties of the protocol or the applications using it, and routers may experience a higher control plane load. After discussing these points, we conclude with remarks on fault and security issues.

Decomposition of MEADcast data packets may yield packets with different destination addresses and thus invalidate checksums in upper layer headers that include network addresses in the checksum (e.g., UDP for IPv6). For the new packet to be valid at the destination, MEADcast routers must re-compute these checksums for every new unicast packet and every new MEADcast packet with a different destination address. This issue is due to the re-use of network addresses in transport layer protocols, and problematic not only because of the increased load on routers' control plane but also because of the requirement to handle protocols other than IP. The checksum calculation in the transport layer by a MEADcast router is similar to that of an Xcast router, which is presented in [10]-Section 10.1.

Transport layer port numbers will differ at end-point sockets and have to be included in the MEADcast header along with the IP address of each end-point, thus creating an additional binding to the transport layer.

Network service primitives do not support addressing multiple receivers. Therefore, applications and higher protocols on the sender side must be modified to make use of MEADcast. A solution idea would be to use “regular” IGMP/MLD-based multicast on the first hop, thus allowing applications and higher protocols to employ multicast addressing as usual, then use a proxy function to translate between regular multicast and MEADcast before transmitting. While Path MTU discovery [21] is a standard function of the Internet, the application requirements on payload size are not readily available to allow the computation of optimum header size. We envision an interface to the network layer allowing the application to issue *hints* with respect to its intended use of the network.

We emphasize that these modifications are required for the sender only. The providers of asymmetric applications (IP-TV, Internet radio, etc.) can be assumed to correctly gauge the cost and benefit of introducing modifications to consolidate the multitude of unicast flows they create presently.

C. Limitations

Inherent limitations of the approach include the maximum number of entries in the address table, the overhead introduced by the address table, the time required to establish multicast structures and load introduced in the control plane of routers.

MEADcast routers do not keep group information, thus rendering MEADcast processing stateless, while nevertheless complex in contrast to multiple flows that may be handled by accelerators such as FPGAs.

MEADcast performs a gradual transition from a number of unicast packet flows to a (smaller) number of multicast flows as the availability of MEADcast-capable routers is discovered. Sessions that are shorter than the time for discovery not only forego the benefit of multicast but also carry the additional load for discovery; they are an application for unicast.

Given a path MTU value, the number of entries in the address table determines the remaining space for payload. If the service data units received from the upper layer is small, the sender may enlarge the number of entries, however, for large number of end-points even small payloads will require the sender to issue multiple multicast packets. Figure 16 shows the critical points where data volume is increased when the address table space of 32 entries is exhausted by one router multicasting to an increasing number of end-points. Conversely, a large address table leaves less space for payload and may lead to fragmentation, as illustrated in Figure 17.

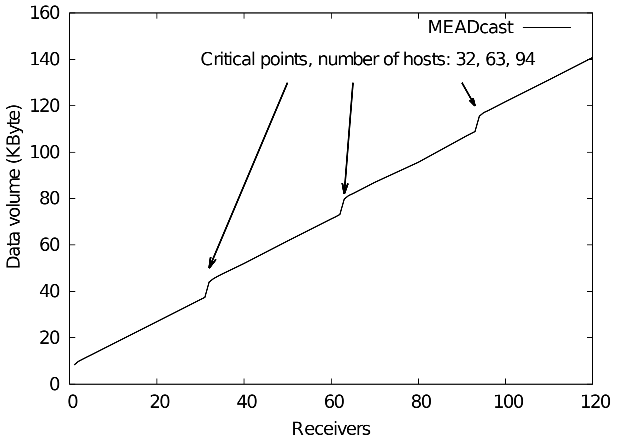


Figure 16. Total volume increased by exhausted address table.

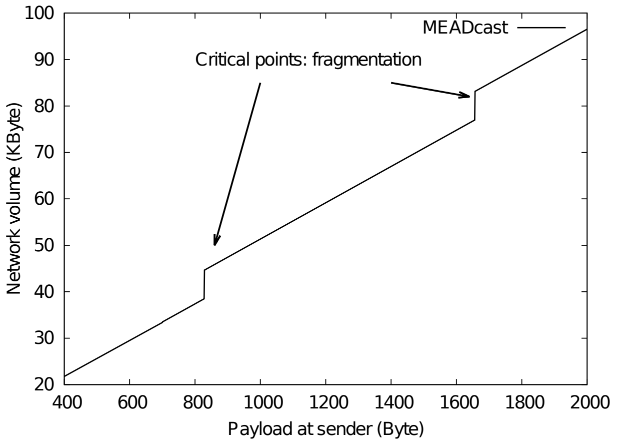


Figure 17. Fragmentation impact on total volume (1 router, 31 end-points).

D. Fault and security considerations

Packet loss naturally incurs a larger penalty for MEADcast than unicast, as more receivers are affected. In particular, the failure of a MEADcast path by changes in routing (by administrative action or by faults) will lead to continuous loss of packets until the periodic discovery mechanism informs the sender of the change in the network topology. A higher discovery frequency might lessen the consequences at the expense of increased control plane load in routers and an increase in the number of (albeit small) packets transmitted over a path.

Beyond the security issues noted for Xcast (see [10]), which also employs sender-based multicast, we note that the deprecation [22] of the Type 0 Routing Header in IPv6 to prevent amplification attacks suggests careful scrutiny of any mechanism that causes Internet routers to transmit more packets than they receive. We presume our mechanism to be reasonably safe due to the following properties: *i)* the total volume of transmitted multicast data does not exceed the corresponding unicast volume for the same data, with the exception of the lightweight signalling overhead required for consolidating multicast, *ii)* addresses are not modified by routers, i.e., data is transmitted via the same path in both unicast and multicast modes.

RFC 6398 [23] warns that the Router Alert Option (RAO) [24], which is useful to indicate MEADcast packets to routers, may be used as an attack vector. The authors recommend, as a measure to defend against attacks, to either forward packets with RAO without evaluating the RAO content or, as a last resort, to drop packets with the RAO. Given that the RAO is employed by a number of well-known protocols, e.g., Resource Reservation Protocol (RSVP), Internet Group Management Protocol (IGMP), we assume that support for RAO will be maintained.

E. Multiple unicast paths between two MEADcast routers

As MEADcast routing relies on unicast, there are cases where different data delivery unicast paths from the sender to receivers cause a MEADcast router to send discovery responses to the sender with different distances (MEADcast hop-count). Figure 18 illustrates a case where the distance of R_3 to S is 4 according to the path from S to E_1 (S-R_1-R_2-R_4-R_3-E_1) while it is 3 for the path from S to E_2 (S-R_1-R_2-R_3-E_2). This case is conceivable in reality for some reason, e.g.,

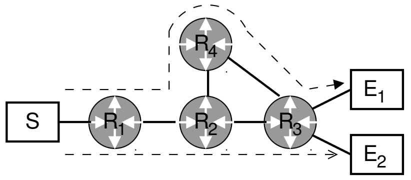


Figure 18. Hop-count conflict situation.

path load balancing deployment causes traffic to be directed on different paths. The network topology viewpoint building process at the sender is influenced: *i)* R_3 could have the same distance to S as R_4 and both are immediately after R_2 or *ii)* R_3 is one hop farther than R_4. Note that the sender's viewpoint is key for the MEADcast data sending phase and has to be a tree (with the sender as root), i.e., a loop-free viewpoint. Therefore, only one possibility of these two can appear in the final sender's viewpoint.

Several strategies could be employed by the sender in this case, e.g., for multiple discovery responses from a source (MEADcast router) with different distances (MEADcast hop-count), the sender could:

1) consider only the distance in the first response, which means this router always has this distance to the sender regardless of any different distance reflected in subsequent discovery responses,
2) choose the shortest distance extracted from all the discovery responses,
3) choose the longest distance,
4) choose the most popular distance, i.e., the distance with highest frequency in all responses.

Our current implementation is based on the second option since it intuitively induces the least latency in data delivery. The longer path can be seen as backup in case the shortest one is broken and comes into use after the next periodic discovery phase. So the path through router R_4 in Figure 18 is considered as redundant and the traffic is normally sent from S to E_1 and E_2 on the path R_1-R_2-R_3.

It is conceivable to amend the specification of router behaviour to effect the transmission of a responding router's IP addresses within the discovery response message, e.g., encoded in the destination address field.

F. Opportunities for optimisation

It is easy to upgrade MEADcast to control the data transmission's efficiency, e.g., which router is responsible for which receiver and how many of them. It is only the design matter at sender side, not at MEADcast router or other end-points. So the upgrade point is only the sender.

1) Reflection of unicast: It is safe to assume that the majority of routers in the Internet are either agnostic or phobic.

- If a datagram is forwarded through agnostic routers past the point where it should be branched, a receiving aware router can forward it back over the same path by unicast if it recognises this to be the case.
- The reflecting router should at the same time notify the sender of reflection being necessary on this path, e.g., by sending it an ICMP message containing the reflected destinations. In response to this message, the sender is expected to stop multicasting to those destinations and use unicast for them instead.
- The reflecting router should process (i.e., reflect) only a limited number of multicast datagrams from a certain source in order to avoid senders that are in violation of the protocol. Reflection, like multicast itself, might be a vector for an amplification attack, using a reflecting router to inject large amounts of traffic into the network.

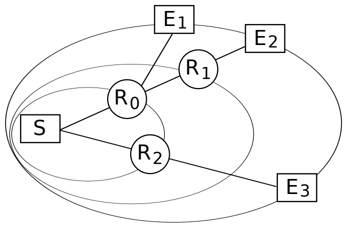


Figure 19. A scenario for optimization consideration.

2) Greedy unicast: If only few destinations are reachable via a given path, it may be more efficient to transmit unicast. E.g., it may not be worth the effort of transmitting and processing extra headers in the case where only one destination is addressed in the MEADcast header. Figure 19 raises such a scenario where according to the sender's viewpoint after the topology discovery process, there are some MEADcast routers (R_1, R_2) responsible for only one receiver. For the setting of allowing a MEADcast router to be responsible for at least one receiver, the data delivery tree is then exactly alike the topology viewpoint of the sender: the sender may build two different MEADcast messages for each branch: $\{R_0, E_1, R_1, E_2\}$ and $\{R_2, E_3\}$ and send them to the direction of E_2 and E_3, respectively. The data delivery process is similar to the steps described in Section IV-F. R_0 composes and sends a unicast message to E_1 and sends a MEADcast message to R_1, who then composes and sends a unicast message to E_2. Obviously, it is not so efficient compared to the case where the first message built by sender is: $\{R_0, E_1, E_2\}$, then R_0 will decompose the MEADcast message into two unicast ones and send them to E_1 and E_2. The transmitted message from sender to R_0 is smaller by saving up the space for R_1 in the address list, the unicast message sent from R_0 to R_1 is also more bandwidth-efficient compared to the MEADcast message in the former case. Similarly, S does not send MEADcast messages to E_3 but unicast ones, which is also more efficient.

VII. Conclusion

The MEADcast protocol introduced in this article addresses long-standing issues of the adoption of multicast. We find that the requirement on group participants to perform group management, as well as the addressing scheme and the intra-domain design of traditional multicast erect barriers to its adoption as a natural, common means for group communication. The massive unicast employed by many services, in consequence, leads to a significant waste of network resources due to the transmission of identical payload over all links to all receivers.

We have introduced MEADcast as a sender-centric multicast protocol that introduces a separation of the concerns of managing groups and holding state (at the sender), forwarding multicast payload without having to keep state (routers) and receiving the payload, without having to have knowledge of the network technology employed (receivers). The discovery mechanism of MEADcast allows a "fall-forward", incremental introduction of multicast transmission, depending on the actual support by the network, on a single receiver basis. At the same time, in contrast to a fall-back mechanism, it avoids the transmission of address tables to receivers and consequently avoids the potential breach of privacy between them. Our experiments have focused on the reduction of the total volume of traffic in the network compared with unicast in a wide range of simulated scenarios with varying support for MEADcast in the network. These experiments have employed both randomly generated topologies with randomized placement of MEADcast routers and topologies specified to reflect real-life networks. We have found that even a modest amount of MEADcast routers in the network yield a reduction in resource use. The gains in performance become larger with a higher degree of support.

Our discussion indicates several open questions and avenues for development, including the study of the load increase in router control planes and the real-world evaluation of streaming applications based on a module implementation for the Linux kernel and the development of an interface for the management of multicast groups and parameters on the sender side. We intend to employ the kernel implementation for studies of the protocol's behaviour during the experimental provisioning of services transported via MEADcast within a moderately controlled environment such as an academic network. A different point of interest is the realisation of MEADcast with virtual network functions, to be used in and between Software Defined Networks (SDN).

Acknowledgment

The authors wish to thank the members of the Munich Network Management (www.mnm-team.org), directed by Prof. Dr. Dieter Kranzlmüller, and Dr. Truong Khoa Phan for valuable comments on previous versions of this paper.

References

[1] C. N. Tran and V. Danciu, "Privacy-preserving multicast to explicit agnostic destinations," in *The Eighth International Conference on Advanced Communications and Computation (INFOCOMP 2018)*. IARIA XPS Press, 2018, pp. 60–65.

[2] S. Deering, "Host extensions for IP multicasting," RFC 1112 (INTERNET STANDARD), Internet Engineering Task Force, Aug. 1989, updated by RFC 2236. [Online]. Available: http://www.ietf.org/rfc/rfc1112.txt

[3] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan, "Internet Group Management Protocol, Version 3," RFC 3376 (Proposed Standard), Internet Engineering Task Force, Oct. 2002, updated by RFC 4604. [Online]. Available: http://www.ietf.org/rfc/rfc3376.txt

[4] H. Holbrook, B. Cain, and B. Haberman, "Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast," RFC 4604 (Proposed Standard), Internet Engineering Task Force, Aug. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4604.txt

[5] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, *Scalable application layer multicast*. ACM, 2002, vol. 32, no. 4.

[6] D. A. Tran, K. A. Hua, and T. T. Do, "A peer-to-peer architecture for media streaming," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 121–133, 2004.

[7] I. Wijnands, E. Rosen, A. Dolganow, T. Przygienda, and S. Aldrin, "Multicast using bit index explicit replication (BIER)," RFC 8279 (Experimental), Internet Engineering Task Force, Nov. 2017. [Online]. Available: http://www.ietf.org/rfc/rfc8279.txt

[8] I. Wijnands, E. Rosen, A. Dolganow, J. Tantsura, S. Aldrin, and I. Meilik, "Encapsulation for bit index explicit replication (BIER) in MPLS and non-MPLS networks," RFC 8296 (Experimental), Internet Engineering Task Force, Jan. 2018. [Online]. Available: http://www.ietf.org/rfc/rfc8296.txt

[9] L. Ginsberg, A. Przygienda, S. Aldrin, and J. Zhang, "Bit index explicit replication (BIER) support via IS-IS," RFC 8401, Internet Engineering Task Force, Jun. 2018. [Online]. Available: http://www.ietf.org/rfc/rfc8401.txt

[10] R. Boivie, N. Feldman, Y. Imai, W. Livens, and D. Ooms, "Explicit Multicast (Xcast) Concepts and Options," RFC 5058 (Experimental), Internet Engineering Task Force, Nov. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc5058.txt

[11] M. Hansen, R. Smith, and H. Tschofenig, "Privacy Terminology," Internet Engineering Task Force, Tech. Rep. draft-hansen-privacy-terminology-03, Oct. 2011, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-hansen-privacy-terminology-03

[12] W. Fenner, "Internet Group Management Protocol, Version 2," RFC 2236 (Proposed Standard), Internet Engineering Task Force, Nov. 1997, updated by RFC 3376. [Online]. Available: http://www.ietf.org/rfc/rfc2236.txt

[13] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment issues for the IP multicast service and architecture," *IEEE Network*, vol. 14, no. 1, pp. 78–88, 2000.

[14] M. Hosseini, D. T. Ahmed, S. Shirmohammadi, and N. D. Georganas, "A survey of application-layer multicast protocols," *IEEE Communications Surveys & Tutorials*, vol. 9, no. 3, pp. 58–74, 2007.

[15] K. T. Phan, N. Thoai, E. Muramoto, K. Ettikan, B. Lim, and P. Tan, "Treemap-the fast routing convergence method for application layer multicast," in *Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE*. IEEE, 2010, pp. 1–5.

[16] K. T. Phan, J. Moulierac, C. Tran, and N. Thoai, "Xcast6 treemap islands: revisiting multicast model," in *StudentWorkshop@CoNEXT*, 2012.

[17] R. Hinden, "Internet protocol, version 6 (IPv6) specification," RFC 8200 (Best Current Practice), Internet Engineering Task Force, Jul. 2017. [Online]. Available: http://www.ietf.org/rfc/rfc8200.txt

[18] M. Waehlisch, T. Schmidt, and S. Venaas, "A Common API for Transparent Hybrid Multicast," RFC 7046 (Experimental), Internet Engineering Task Force, Dec. 2013. [Online]. Available: http://www.ietf.org/rfc/rfc7046.txt

[19] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *INFOCOM'96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE*, vol. 2. IEEE, 1996, pp. 594–602.

[20] Y.-h. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *IEEE Journal on selected areas in communications*, vol. 20, no. 8, pp. 1456–1471, 2002.

[21] J. McCann, S. Deering, J. Mogul, and R. Hinden, "Path MTU discovery for IP version 6," RFC 8201 (Best Current Practice), Internet Engineering Task Force, Jul. 2017. [Online]. Available: http://www.ietf.org/rfc/rfc8201.txt

[22] J. Abley, P. Savola, and G. Neville-Neil, "Deprecation of Type 0 Routing Headers in IPv6," RFC 5095 (Proposed Standard), Internet Engineering Task Force, Dec. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc5095.txt

[23] F. L. Faucheur, "IP Router Alert Considerations and Usage," RFC 6398 (Best Current Practice), Internet Engineering Task Force, Oct. 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6398.txt

[24] C. Partridge and A. Jackson, "IPv6 Router Alert Option," RFC 2711 (Proposed Standard), Internet Engineering Task Force, Oct. 1999, updated by RFC 6398. [Online]. Available: http://www.ietf.org/rfc/rfc2711.txt

Evaluating Security Products: Formal Model and Requirements of a New Approach

Pierre-Marie Bajan* Christophe Kiennert† Hervé Debar†
*Université Paris-Saclay and Institut de Recherche Technologique SystemX (Saclay, France),
Email: {first.last}@irt-systemx.fr
†Télécom SudParis (Evry, France),
Email: {first.last}@telecom-sudparis.eu

Abstract—**In a previous paper, we presented a new method to generate evaluation data for the evaluation of security products and services. That approach tackles the issues of producing a workload with a rich semantic at a large scale. Testbed environments are the most appropriate tool for such task but induce a lot of effort and costs to implement. We presented a model to produce semantic data that can be implemented on light virtual networks and thus deployed at a large scale. This paper is an extension of our complete formal model. In this extension, we identify additional requirements for our model and define our ambitions. We translate those ambitions in verifiable properties of our model. Our prototype, although currently limited, provides the basis for an evaluation method that is customizable, reproducible, realistic, accurate and scalable. We generate realistic activity for up to 250 simulated users interacting with a real-world webmail server in an experiment to verify the properties of our model.**

***Keywords*–cybersecurity; simulation; evaluation; formal method.**

I. Introduction

Security products are composed of services and products designed to protect a service, machine or network against attacks. Like other products, they must be tested to guarantee adherence to specifications. In a previous paper [1] published at ICIMP 2018, we divided evaluation tests into two categories: *semantic tests* – tests of capability that require data with a high-level of semantic; and *load tests* – tests that subject the product to a large workload.

With current testing methods [2], *load tests* are semantically poor, thus not realistic. Meanwhile, *semantic tests* either require vast amount of resources to reach large scales (e.g., testbed environments), or rely on real life captures with their own set of challenges (e.g., elaboration of the ground truth and privacy concerns). Moreover, a complete evaluation of a security product tests different properties of the product. Thus the evaluator needs to select different methods with the right granularities. The granularity of interactions of the data corresponds to the level of control or precision of the data. For an evaluator, the right granularity for a testing method is a granularity that is fine enough to test specific vulnerabilities or properties. A granularity too large does not match the need of the evaluator and a granularity too fine may result in a drastic increase of the preparation burden of the evaluator for little to no improvement of the results. Rather than relying on several methods with different granularities, we aim to elaborate a method to produce data with a customizable granularity and the possibility to achieve large scale generation with appropriate semantic.

In this paper, we present a methodology to produce simulated evaluation data with different granularities independently of the network support. To achieve variable granularity of our model, we formally presents the concepts of our simulation and define properties that must be respected and verify. We also extract five requirements from our analysis of existing methods to determine the criteria of an ideal method: customizable, reproducible, realistic, accurate and scalable. To ensure that the method we propose is as ideal as possible, we convert those requirements in additional properties of our method. We then ensure that the developed prototype of our method respect all the raised properties in a series of experiments.

The remainder of this paper is organized as follows. Section II reviews the related work on the production of evaluation data and their limits. Section III is our analysis of current methods compiled into five criteria for an data generating method. Section IV defines the concepts of our methodology and uses those concepts to introduce our model. Section V explains the different choices we made for the implementation of our prototype and shows the experiment results to validate our model. Finally, we conclude our work in Section VI.

II. Related work

We first present existing related to the generation of semantic tests.

A. Semantic tests

Semantic tests generate evaluation data with high semantic value. Their goal is to generate realistic workloads to produce real-life reactions of the security product or to test specific functionalities and vulnerabilities of the product. There are two approaches to those tests: take data from the real world (the highest level of semantic), or execute specialized tools and homegrown scripts.

Real world data can come from several sources: provided by real world organizations, or obtained from honeypots where attackers were tricked into interacting with a recording system to learn about the current trends (ex: generation of intrusion detection signatures using a honeypot [3]). However, the evaluator does not have a complete knowledge of the content of the data. Some of them can be misidentified or the intent behind some actions misinterpreted. Moreover, real world data are difficult to obtain. Organizations are reluctant to provide data that can damage their activity, and data anonymization has the drawback of deleting relevant information (e.g., challenges of anonymization [4] and desanonymization techniques [5]). As for honeypots, the evaluator can never know beforehand the amount of data he can obtain or what kind of data he will gather.

Another way to obtain high semantic data is to generate them according to a defined scenario, relying on tools and scripts to produce specific and calibrated data. Those scripts can be homegrown scripts, exploits, or software testing scripts that try every function of a software to validate its specifications. Manually generating the data (e.g., video transcoding [6], file copy operations [7], compiling the Linux kernel [8], etc.) offers the greatest control over the interactions inside the data, but the automation of the activity generated through scripts with tools like exploit databases (Metasploit [9], Nikto [10], w3af [11], Nessus [12]) also offers good control. However, those methods are quite time-consuming or require in-depth knowledge of the evaluated product. Moreover, the granularity of control for varied for each tool and may not be appropriate for the evaluator.

B. Load tests

Load tests create stress on the tested product [13]. The most common tests use workload drivers like SPEC CPU2000 [8], ApacheBench [14] [15], iozone [15], LMBench [16] [8], etc. They produce a customizable workload with a specific intensity. The evaluator can also manually start tasks or processes known to stimulate particular resources (e.g., kernel compilation [14] [16], files download [17], or execution of Linux commands [17]). Those methods are designed to test particular resources of a system (like I/O, CPU and memory consumption) or produce large amount of workload of a specific protocol. For example, SPEC CPU 2017 generates CPU-intensive workloads while ApacheBench generates intensive HTTP workloads. However, the semantics of the workloads are low: the generated data are characteristic of the driver used and do not closely resemble real life data.

C. Deployment of scalable semantic tests

Evaluators prefer tests that are both intensive and with a high semantic, as the performance of security products like intrusion detection systems often deteriorate at high levels of activity [18]. The prefered method is to deploy semantic tests at a large scale. However, the deployment of tests with a realistic semantic at a large scale provides its own challenge.

Semantic tests are deployed on a large scale network support like a testbed environment where a large amount of resources and contributors are gathered to create a large-scale test. Evaluators must either have access to a testbed environment with enough resources to deploy large-scale experiments or use the results of other organizations that conducted large-scale experiments and made their data publicly available for the scientific community (DARPA/KDD-99 [19], CAIDA [20], DEFCON [21], MawiLab [22], etc.).

However, publicly available datasets, on top of often containing errors [2], are not designed for the specific needs of each evaluator. The evaluator needs to have an in-depth knowledge of the characteristics of the activities recorded in the dataset to avoid having an incorrect interpretation of the results of studies using those datasets. Finally, there is the issue of freshness of the traces. Those large-scale experiments produce one-time datasets that are quickly outdated.

III. Our analysis

From the analysis of existing method, we took into consideration the strengths and weaknesses of each method and came to the conclusion that, among the current existing methods, testbed environments are the best tool available for an evaluator to properly evaluate products. They allow the evaluator to have a single evaluation environment that can perform semantic tests that doubles as load tests. The large scale of the environments allows the evaluator to use a wide range of tools in a single experiment. With a proper activity model to conduct the evaluation, the evaluator can have an evaluation traffic close to reality with full control over parameters like intensity of the workflow or the introduction of incidents. The evaluator has full knowledge of the activity of the simulation and does not have to face issues from real world data like anonymisation.

However, testbeds environments have a serious flaw: the prohibiting cost of setting up, maintaining and generating large-scale experiments. We analyzed the reasons of that cost to find a way to make evaluation data with the same semantic richness than testbed environments more easily available. We came to the conclusion that one of the main culprit of the cost of such environments is the network infrastructures.

In this section we offer our analysis on how to reduce the cost of network infrastructure by crafting a method compatible with lower-end network simulator. We also share the analysis of the criteria of an ideal method to generate evaluation data. During our study of related work, we observed that the strengths and and weaknesses of all the methods mostly revolved over five major requirements: customization, reproducibility, realism, accuracy and scalability. We explain in this section those requirements in further details.

A. Using process virtual machines

One of the reasons for that cost is that the network infrastructures for testbed environments are composed of either physical machines or virtual machines (VMs) that create simulated hardware upon which the evaluator installs a complete operating system. There are virtual network infrastructures that use lighter VMs that do not simulate the hardware but only primary functions of the operating system (Wine, Mininet, IMUNES, etc.). However, evaluators do not use those network infrastructures because those light VMs do not support a wide range of real-world application and programs.

For example, IMUNES generates cloned kernels coupled with Jail in FreeBSD systems to use them as virtual machines [23]. Those cloned kernels do not handle a graphical interface and cannot launch a browser. Thus, using this type of virtual network for a testbed environment seriously limits the possible actions taken by the VMs, even if a lot more cost-effective for large scale virtual network.

Rather than working on improving the virtualization tools, we decided to take another approach on the issue and to improve the method to generate evaluation data with a semantic richness. We propose a new production method that generates controlled activity data from short traces independently of the network support. This method consists in a single program reproducing the simulation data of a large variety of programs with a level of quality sufficient for an evaluator need. This program does not complete the tasks of the real-world programs but only reproduce model data of such programs. Even if the reproduced data are not perfect, they must at least comply with the evaluation requirements of the evaluator.

Thus, rather than deploying VMs that can handle commonly used real-world programs, we can deploy a large number of light VMs that can run our program producing evaluation data from model data. It can be implemented on a testbed environment or on a network support with lower requirements like a lower end network simulator. We also want our method to meet the need of evaluators to generate tests with a rich variety (different systems, properties of the data, etc.) and to devise hybrid tests, both semantic and load oriented.

B. Essential requirements for an data generation method

Our ambition is to propose an evaluation data method that can incorporate most of the strengths of current data generation method while leaving out their weaknesses.

In a cross analysis of what was presented as strengths or weaknesses of other methods, we highlight five goals, or requirements for an ideal method, that represent the most relevant aspect of the current state of evaluation data method.

Ideally, an evaluation data method must be:

- **Customizable**: one of the main strengths of executable workloads is that the evaluators can customize the generated activity to match their needs. It allows them to use the same tools to test a security product with different metrics. Meanwhile traces only allow the test of one scenario per trace. We want our method to be able to offer a wide variety of parameters to modulate and produce evaluation data according to the needs of the evaluator. We also want our method to avoid the usual issue of the freshness of traces.
- **Reproducible**: one of the biggest weakness of executable workloads is that it is often time-consuming to restore the victim environment to its previous state, especially in a sophisticated setting like a testbed. A significant advantage of traces is that it is easy to feed to a security product. In consequence, traces are used as a standard for the community. We want a method that can provide evaluation data with little to no overhead to restore a victim environment.
- **Realistic**: real-world production traces and honeypots allow evaluators to confront security products and services with real-life data and attacks. With a realistic activity model, manual generation can produce a close to real life evaluation data. Its capture, after deployment at a large scale in a testbed environment, are the publicly available traces. The community considers that method of generating data as sufficiently realistic to be used as a reference. As such, we want our method to be able to provide realistic evaluation data to the level of publicly available traces provided with a realistic activity model.
- **Accurate**: one of the main weakness of traces obtained from real-world production and high level interaction honeypots is the difficulty to generate the ground truth of traces. The evaluator cannot guarantee the accuracy and correctness of the generated ground truth. Isolating attacks from one another or legitimate traffic is challenging. An ideal evaluation data method should have full knowledge of the activity generated.
- **Scalable**: large-scale testbeds allow the evaluators to generate complex and realistic traffic despite the large cost of doing so. Publicly available traces aim to provide that complex traffic at structures that cannot afford that high cost. We want our method to be able to generate the evaluation data of large scale networks while offering the possibility to choose the size of this network.

These five requirements are the ambition of our evaluation data method. Even if our method ends up not being as ideal, we want our approach to respect as much as possible those requirements.

We also want to be able to provide evaluation data for live testings and offline testings. As such, we want our method to be close to executable workloads.

IV. Data production method model

In this section, we present a formal model for our data production method. A formal representation of our model allows us to present a generic approach that is not restrained to a single implementation. Network simulators do not use similar simulation techniques, even among the same category of network simulators. The support of different network infrastructures provides different strengths that may be of interest to the evaluators. In the same way, implementation choices can also impact the properties and capacity of our model. A formal model presents a generic model that allows for different implementation approaches. By clearly defining the properties of our model, we can devise tests to verify that the implementation is in accordance with the model.

In our model, we define four core concepts: *Elementary actions*, *Data generating functions*, *Scenarios*, and *Scripts*.

After those definitions and the presentation of our simulation model, we deduce formal requirements on our model according to the five ideal criteria. In short, we add several properties to our model to ensure to respect as closely as possible the criteria of an ideal method.

A. Concepts and definitions

1) Elementary action: We call *Elementary action* (A) a short ordered set of interactions that represents an action between two actors of the activity. Those actors are either a *Host* – a source of generated data – or a *Service* – a set of functionalities available to a *Host*. A *Service* can be an external server or an internal service.

The goal of *Elementary actions* is to divide the activity we simulate in actions that correspond to an entry of the ground truth, such as "connection to the web interface of a webmail server". The ground truth is an exact representation of the activity generated. Therefore, a finer set of *Elementary actions* for an activity means a finer representation of the simulated activity and a finer control of the activity model for the evaluator. Roughly translated, even if the model does not forbid it, an *Elementary action* is not meant to be a large set of interactions like "a day of activity of user U". An *Elementary action* intends to represent a short action like "connection to service S" or "adding an entry to service S'".

For each *Elementary action*, we acquire *Model data* that are the captured data of the execution of this *Elementary action* during real activity (activity not issued from our simulation). *Model data* take different forms (traces, logs, values, etc.) according to the nature of the data the evaluation target can handle.

Furthermore, the evaluator can classify the *Model data*. The evaluator uses that classification to help label the resulting *Simulation data* and create a labeled ground truth. However, the evaluator is in charge of deciding a classification as it can change according to the need of the evaluator. For example, the evaluator can create two classes of *Model data* to represent malicious activity and benign activity, respectively. In other contexts, the evaluator can define other classes. For the evaluation of administration tools of a network, the actors to consider are different (security: attacker/user, administration: admin/user/client) and the evaluator will have to define classes of data accordingly.

After capturing *Model data* for every *Elementary action* relevant for the evaluation, the resulting set of *Model data* is then given to a *Data generating function*.

2) Data generating function: We define a *Data generating function* (f) as a function that creates *Simulation data* from *Model data*. *Simulation data* ($d^{simulation}$) is the execution of an *Elementary action* (A) during a simulated activity.

A *Data generating function* (f) takes two inputs: *Model data* (d^{model}) and a set of *Elementary action parameters* (p^A). We define later the *Elementary action parameters*.

The properties of the *Simulation data* and *Model data* represent the level of realism as seen by the evaluator. They correspond to a set of specific features of the data, either qualitative or quantitative. The properties of the data are of different forms: acknowledgement of the data by the *Service*, size of sent packets, value of a measure, etc. and they represent the level of realism chosen by the evaluator.

Our definition of *Data generating function* does not include requirements on the output, the *Simulation data*. We express our demands for *Simulation data* as *Equivalence*. We call *Equivalence* ($\sim$) the fact that two activity data have the same properties.

$$d_A^{activity} \sim d_A'^{\,activity}$$
$$\Longleftrightarrow$$
$$Properties(d_A^{activity}) = Properties(d_A'^{\,activity})$$

It is important to specify that two data are only equivalent in the eyes of evaluator. The only identical properties of two equivalent data are the properties of interest to the evaluator. So if not all properties are necessarily identical, it also means that two equivalent data are not necessarily identical. For example, if the evaluator is only interested in the volume of the traffic and the size of the packets, two packets with the same size but different content will be equivalent.

A *Data generating function* can produce *Simulation data* that are not identical but equivalent for the same inputs but executed at different times. For instance, if the generation of the *Simulation data* includes the addition of randomly generated parameters like tokens. However, we do not consider times as an input of our formalism of *Data genration functions*. We do not include time as an input because, although time can impact if the *Simulation data* are identical, it does not impact the equivalence of *Simulation data*.

The evaluator selects a set of *Elementary actions* to decide the finesse of control over the simulation and he chooses a *Data generating function* to reproduce the properties of the data he requires. If the *Data generating function* that produces *Simulation data* from a dataset of *Model data* cannot produce data with the same properties, it is useless for the evaluator. Thus, we define the following verification property of *Data generating functions*:

Property 1: a *Data generating function* f is said to be **useful to a set of *Model data*** $\mathcal{D}$ if all *Simulation data* generated by f from any *Model data* that belong to $\mathcal{D}$ is equivalent to the data used as model.

$$\forall\ p^A, \forall\ d \in \mathcal{D}\ and\ f\ /f(d, p^A) = d^{simulation}$$
$$f\ is\ useful\ to\ \mathcal{D} \Leftrightarrow\ \forall d \in \mathcal{D}, d \sim d^{simulation}$$

The evaluator can select the *Data generating function* among a pool of *Data generating functions* with *Simulation parameters* ($p^{simulation}$). The evaluator chooses the function that is useful to his *Model data* and provides the *Data generating function* with additional parameters called *Elementary action parameters* (p^A). *Elementary action parameters* allow the evaluator to modify the behavior of the *Data generating function*. It can be to match a larger dataset of *Model data* or to provide a finer control.

We call *Elementary action parameters* (p^A) associated to an *Elementary action* (A) a set of parameters provided to a *Data generating function* (f) to produce *Simulation data* that can impact positively the usefulness of the *Data generating function*.

$$\forall p^A, \quad \left. \begin{array}{l} f(d_A, p^A) \ is \ useful \ to \ \mathcal{D} \\ f(d_A, \emptyset) \ is \ useful \ to \ \mathcal{D}' \end{array} \right\} \Rightarrow \mathcal{D}' \subseteq \mathcal{D}$$

For example, we take a *Data generating function* that preserves the property "acknowledgement of the data by the *Service*" of traces given to it. That function, in order to be useful to the dataset that has the same property, will change some time-sensitive information on the input data like a cookie or a session ID. However, different *Services* may mark this information differently. A service S_1 might mark the session ID with the tag "&session_id=" while a service S_2 will mark it with the tag "&_session=". Some services might also have other additional change in their interactions that other services do not have. To avoid relying on a different *Data generating function* for every service due to those insignificant differences, we want to parameter the transformations applied by the *Data generating function*.

Moreover, the evaluator might want to produce *Simulation data* from the same *Model data* but with different results. For example, the evaluator wants to reproduce the *Elementary action* "connect to a webmail" with the conservation of the property "acknowledgement of the data by the *Service*". However, left as it is, the *Simulation data* always present the same credentials. The evaluator may want to simulate connection to the webmail with different IDs without being force to provide a different *Model data* for every set of credentials. It is necessary to be able to parameter the function to make small modifications rather than having a lot of *Model data* for the same *Elementary action*.

The *Data generating functions* are controlled and fed by a simulation control program. This program gives order to virtual hosts to execute specific *Data generating functions* with specific inputs. The orders are issued on a separate virtual network. The resulting data exchanged between the simulation control program and the virtual hosts are called *Control data*.

To sum up, *Data generating functions* are selected with *Simulation parameters* by the evaluator for the properties they preserve and the evaluator adapts or controls the *Simulation data* with *Elementary action parameters*. The data exchanged by the program that controls the simulation and the *Host* that runs a *Data generating function* is the *Control data* ($d^{control}$) and is essentially the ground truth of the simulation. The compilation of the *Control data* informs us of all the actions taken during the simulated activity.

3) Scenario and Scripts: A *Script* is the representation of a realistic behavior of a *Host*. We define a *Script* as an ordered set of *Elementary actions* coupled with *Elementary action parameters*. A *Script* ($Script_H$) is defined for each individual *Host* and describes the activity it must generate during the simulation. The set of defined *Scripts* is called the *Scenario* (Sce) of the simulation.

A *Script* can be represented as a graph of actions, as illustrated in Figure1.

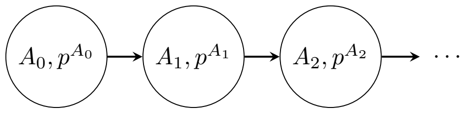


Figure 1. Example of a *Script*

B. *Our model*

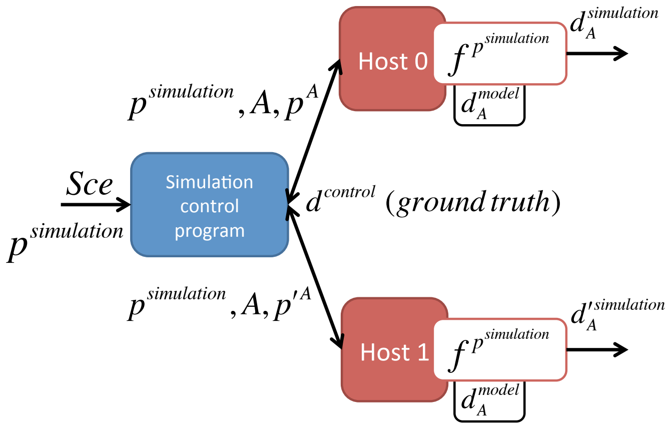


Figure 2. Generation of simulated activity from short traces

Figure 2 is a representation of our model. In that figure, the evaluator provides the simulation control program with the *Simulation parameters* ($p^{simulation}$) and the *Scenario* (Sce):

$$Sce = \{Script_{H_0}, Script_{H_1}\} = \{([A, p^A], \ldots), ([A, p'^A], etc.)\}$$

The simulation control program interprets the *Scenario* and the *Simulation parameters* and deduces the number of *Hosts* in the current simulation. It instructs the *Hosts* H_0 and H_1 to reproduce the *Elementary action* (A) with the parameters $p^{simulation}$ and p^A. Then, each *Host* retrieves the *Model data* associated to the *Elementary action* and executes the *Data generating function* (f) selected in the *Simulation parameters*. That function produces *Simulation data*, which are sent to a *Service*. The use of different *Elementary action parameters* by H_0 and H_1 results in the generation of different *Simulation data* even when the *Data generating function* and the *Model Data* are the same:

$$\left. \begin{array}{l} d_A^{simulation} = f^{p^{simulation}}(d_A^{model}, p^A) \\ d'^{simulation}_A = f^{p^{simulation}}(d_A^{model}, p'^A) \end{array} \right\} \not\Rightarrow d_A^{simulation} = d'^{simulation}_A$$

However, while those two *Simulation data* may not be equal, they are equivalent.

After the *Hosts* inform the simulation control program that they finished simulating the *Elementary action* A, they await the next simulation orders from the simulation control program.

The model we presented is the situation where all the *Hosts* are simulated and the *Services* are real services. If some *Hosts* also acted as *Services*, they could also initiate the generation of *Simulation data* according to requests received from other *Hosts* in the form of other *Simulation data*.

In our model, the processing of the *Control data* of *Hosts* simulated by our model generates the ground truth of the simulation. Therefore, we do not process data from *Hosts* unrelated to the simulation (external hosts connected to the simulation). The evaluators must incorporate those elements to their ground truth.

Lastly, we must present one of the significant issues of our model: the parameterization of the *Elementary actions*. The parameterization is the addition of *Elementary action parameters* to extend the scope and variability of the *Data generating function* while still preserving various data properties. However, the higher level the preserved properties are, the more complex the reproduction of *Elementary actions* becomes. Therefore, designing a *Data generating function* for a highly realistic simulation, where for instance not only packet size is preserved but also data acknowledgment, requires to consider three main aspects:

- **typing**: identification and generation of short-lived data like tokens, identifiers of session, etc.
- **semantics**: modification of inputs with a high semantic value in the *Model data*: credentials, mail selection, mail content, etc.
- **scalability**: a large scale execution of the *Data generation function* can have consequences on the previous aspects and requires additional changes (e.g., creation of multiple user accounts in the *Service* database).

These three aspects are integrated to the *Elementary action parameters*. However, a few in-depth issues still require further consideration and development in order to elaborate a model able to adapt to various test situations without the intervention of the evaluator. The typing issue can be solved with methods based on machine learning, but others may require specific methodologies according to the context of the evaluation. For example, in the case of the reproduction of a real-life network, the semantic and scalability issues can be solved with analysis of an extended *Model data* acquisition period. The evaluator can identify and highlight inputs in the *Model data* with a high semantic value for the simulation.

C. Formalisation of data generation method requirements

Here, we discuss how to translate the requirements into properties of the model of our method or, in case it is not possible, into implementation requirements.

1) Reproducibility: Reproducibility concerns two different aspects: an experiment carried out several times in the same condition must produce the same results, and a previous experiment should not impact a new experiment – which comes down to the ability to restore the simulation environment to a starting state.

The first requirement of reproducibility means that when reproducing a similar event in the same context, our simulation must provide a similar result. We translate it as a property on *Data generating functions*:

Property 2 (Reproducibility property of Data generating function): A *Data generating function* f is said to be **reproducible** if, for any *Model data*, the resulting *Simulation data* generated at any instant is equivalent to any *Simulation data* previously produced with the same input data.

$$\forall d \in \mathcal{D},\ \forall t' \neq t \quad \left.\begin{array}{l} at\ instant\ t, f(d, p^A) = d^{simulation} \\ at\ instant\ t', f(d, p^A) = d'^{\,simulation} \end{array}\right\}$$

$$f\ is\ reproducible\ \Leftrightarrow\ d^{simulation} \sim d'^{\,simulation}$$

The second requirement of reproducibility is not a requirement on our model but on the virtual infrastructure of our simulation. The implementation of our model must allow the creation of networks in similar conditions without any lasting impact from any previous use of the simulation. This requirement will impact the choice we make for the network support of our simulation implementation.

2) Realism: The realism of our simulation depends on two factors:

- The *Data generating function* and *Model data*: the *Simulation data* produced are only as realistic as the *Data generating function* and the input data allow it. The verification property of the Data generating function (cf. Property 1) ascertains that the *Simulation data* produced is as realistic as the *Model data* used as input of the *Data generating function*. Thus, for *Simulation data* to be realistic, the *Model data* must also be considered realistic for the same criteria of realism. Moreover, the *Data generating function* must preserve the properties of the *Model data* that the evaluator considers as realistic.
- The *Scenario*: the activity our simulation produce is only as realistic as the *Scenario*. At its core, the *Scenario* is the activity model of the evaluator for the simulation. A realistic activity model with realistic *Model data* will result in a realistic activity. We want our model to verify that property.

Property 3 (Verification property of the Scenario): The *Model data* generated by a real activity according to a *Scenario* must be equivalent to the *Simulation data* generated by a simulated activity of the same *Scenario*.

In short, our simulation model can generate an activity according to a *Scenario* with the guarantee that the properties of the *Simulation data* are as realistic as the input data. As to what properties are guaranteed, it will depend on the *Data generating function* chosen by the evaluator. In a more concrete example, if our method makes a simulation of a network activity with a *Data generating function* that preserves the packet size of the model data, we can guarantee that the produced *Simulation data* will present a volumetric network activity as realistic as the *Scenario* of the simulation and the *Model data*.

3) Adaptability: The evaluator can adapt a simulation on our model with several aspects.

Firstly, the *Scenario*. The *Scenario* allows the evaluator to generate the activity of different activity models. With a large variety of *Elementary actions*, the evaluator can create complex *Scenarios* and obtain realistic *Hosts*' behavior during the simulation. Moreover, the more atomic the *Elementary actions* are, the more precisely the evaluator can control the simulation and create *Scenarios* adapted to his needs.

Secondly, the *Elementary action parameters*. They can preserve the realism of the *Simulation data* while offering another degree of customization. The evaluator can significantly improve the semantic value of the *Simulation data* by using those parameters to modify customizable inputs in the *Model data* (credentials, POST form inputs, etc.).

Lastly, the *Data generating functions*. The variety of *Data generating functions* to select from is one strength of our model. Each *Data generating function* offers a guarantee of realism as seen in Section IV-C2. We can define levels of realism that correspond to the selection of different *Data generating functions*. As the realism property of our model partly depend on the *Data generating function*, a customizable *Data generating function* means an adaptable realism of the simulation.

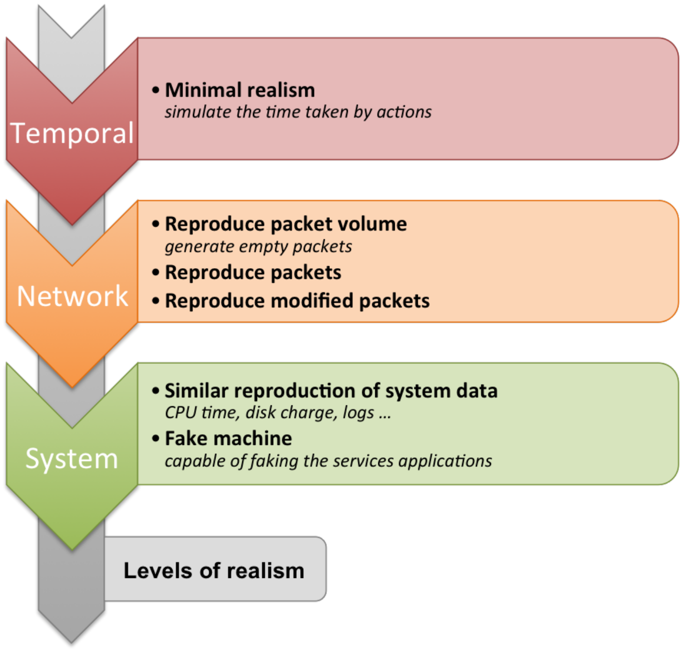

Temporal
• Minimal realism
simulate the time taken by actions
Network
• Reproduce packet volume
generate empty packets
• Reproduce packets
• Reproduce modified packets
System
• Similar reproduction of system data
CPU time, disk charge, logs ...
• Fake machine
capable of faking the services applications
Levels of realism

Figure 3. Levels of realism

In Figure 3, we exhibit several levels of realism associated with several *Data generating functions*. We divide these levels into three main types: **Temporal**, **Network** and **System**.

The first type, **Temporal**, is the minimal level of realism possible and is useful only when the evaluator is solely interested in the elapsed time of different *Scenarios*. In this case, the simulation of activity consists of *Hosts* waiting the average time of each *Elementary action*.

The second type of levels of realism, **Network**, is chosen when an evaluator is solely interested in the simulated network traffic. We identify three levels for this type:

- **Reproduce packet volume**: if the evaluator has an interest in the network charge of the simulation, with no interest for the content. From the *Model data* of each *Elementary action*, the *Data generating function* reproduces all the packets up to the transport layer before padding the payload with random bytes to match the size of the input data's packets.
- **Reproduce packets**: if the evaluator is interested in reproducing the input data with the full payload. The *Data generating function* reproduces the packets while changing the appropriate fields (tokens, session IDs, etc.) for a correct exchange between the *Host* and the *Service*.
- **Reproduce modified packets**: if the evaluator needs more precise control of the previous **Reproduce packets** level of realism. The *Data generating function* produces the packets as described above, but *Elementary action parameters* customize the output. For example, if the *Data generating function* reproduces the *Model data* to connect to the webmail server, the evaluator also modify the credentials used to connect to the *Service* by the *Model data*.

Finally, the third type of realism is **System**. It is the highest level of realism we imagined for the simulation. With **Reproduction of system data**, we want to reproduce the system data measured on the *Hosts* so that it matches the *Model data*. For the last degree of realism we want the *Data generating function* to reproduce the data of service applications – to provide network and system data without executing the applications. It is, in essence, a **Fake machine**.

All those different levels of realism correspond to different *Data generating functions* that the evaluator can select for his simulation. It is also possible to imagine and add other levels of realism or *Data generating functions* with different uses. Those levels of realism represent our implementation goal for a customizable realism of our model.

To be noted, the nature of the *Model data* described in each type of level of realism is different. **Temporal** uses measure values as *Model data*, **Network** has network traces, and the *Model data* of **System** are a combination of system data and network traces.

To sum it up, the evaluators can customize the simulation of our model according to:

- The scenario and network topology (size, topology, connexion to external networks, . . .)

- The *Model data*: the scope of *Elementary actions* (smaller or larger number of interactions)
- The *Elementary action parameters*: customize inputs with semantic value (credentials, information fields, submission contents, etc.)
- The *Data generating function*: the evaluator can select a level of realism useful for a specific dataset or the preservation of specific properties.

4) Accuracy: The constitution of the ground truth is deduced from the analysis of the *Control data*. We want the *Control data* of our model to contain the information on the input of every event ($p^{simulation}, A, p^A$), to know when the *Host* has reproduced the *Elementary action*, how long it took to do it and what the result was (success, failure, error, etc.).

Our simulation must guarantee that information. We translate it into an implementation requirement of our model:

Property 4 (Implementation requirement): The *Control data* of the simulation must include the following information for every entry of the *Scenario* associated to an *Elementary action*:

- The *Elementary action* an *Host* took
- The timestamp of that action
- The *Elementary action parameters* sent with the action
- The result of that *Elementary action*
- The time it took to carry it

By combining this information with the traces used for each *Elementary action*, it is possible to label a record of the activity generated by the simulation of our model quite simply.

In our model, each *Host* only receives instructions to replay one *Elementary action* at a time. Only the simulation control program knows the full *Scenario* and knows which *Elementary action* and *Elementary action parameter* any given *Host* will replay next. Each *Host* is in a stateless situation at any given point of the simulation. We could choose to give the *Script* of the *Host* to each of them at the start of the simulation and let them deal with the execution of the *Script*. However, it would risk having the *Hosts* act out of sync with each other, especially when introducing elements of randomness in the *Scripts* of the *Hosts*.

With a centralization of the decision-making process of the simulation, we have a central point with full knowledge of the situation of the simulation at any given point in time, which facilitates the constitution of the ground truth of the simulation.

However, our model is not connected to the output of the tested product. Our model solely handles the data sent to the tested product and can label the *Simulation data* produced. While our model also receives data, it does not know if the data was correctly processed by external components nor does it have access to their logs. Those elements are necessary for a complete ground truth. For example, if our simulation asks an *Host* to reproduce the *Elementary action* "connect to the webmail server of the company", we will know what *Data generating function* it used, when and how long it took for the *Host* to do it and if the execution of the function went well. However, we do not have access to the log of the webmail server telling us if the connection was successful. That information is not present in the *Control data*. The evaluator would have to provide that information to the ground truth generated by the simulation.

In this example, we assumed that we are in the case where we simulate only the clients of real-life services. If we also simulated the *Services*, then the *Control data* would also contain the information on the output of the tested product.

The scope we guarantee for the constitution of an accurate ground truth only goes as far as the elements simulated by our simulation goes. The constitution of the ground truth with the data of external elements connected to our simulation is at the charge of the evaluator.

5) Scalability: Scalability is an essential issue for evaluation data. It is a property that rapidly increases the difficulty and cost of most methods to generate evaluation data. However, it is an interesting property because it allows the constitution of complex and large-scale activity close to a real-world environment. One of the interesting property of having a scalable network simulation is that the *Simulation data* generated by two *Hosts* (H_0 and H_1) is not the same thing as the *Simulation data* of a single *Host* H_0 doing the same action twice.

$$\exists\{d, f^{p^{simulation}}, p^A\}/ \quad \left.\begin{aligned} d_{H_0}^{simulation} &= f_{H_0}^{p^{simulation}}(d, p^A) \\ d_{H_1}^{simulation} &= f_{H_1}^{p^{simulation}}(d, p^A) \end{aligned}\right\}$$

$$d_{H_0}^{simulation} \cup d_{H_1}^{simulation} \neq d_{H_0}^{simulation} * 2$$

In the same way, having 50 *Hosts* making one connection is not the same thing that having one *Host* making 50 connections at the same time. The impact on the tested product is also not the same, especially if that security product must follow the activity of each user separately.

Our model guarantees that if two *Hosts* use the same useful *Data generating function* on the same *Model data*, then the resulting *Simulation data* are equivalent.

Proof.

- According to Property 1:

$$f\ is\ useful\ to\ \mathcal{D},\ if\ \forall d \in \mathcal{D}, d \sim d^{simulation} = f(d, p^A)$$

- So, if the *Hosts* H_0 and H_1 produce *Simulation data* with f then:

$$\forall\{d, p^A\}/ \quad \left.\begin{aligned} d_{H_0}^{simulation} &= f_{H_0}(d, p^A) \\ d_{H_1}^{simulation} &= f_{H_1}(d, p^A) \end{aligned}\right\}$$

$$\Rightarrow \quad d_{H_0}^{simulation} \sim d\ and\ d_{H_1}^{simulation} \sim d$$

- The definition of equivalence is that two data are equivalent if and only if their properties are the same. So:

$$\left.\begin{aligned} Properties(d_{H_0}^{simulation}) &= Properties(d) \\ Properties(d_{H_1}^{simulation}) &= Properties(d) \end{aligned}\right\}$$

$$\Rightarrow \quad Properties(d_{H_0}^{simulation}) = Properties(d_{H_1}^{simulation})$$

$$and \quad Properties(d_{H_0}^{simulation}) = Properties(d_{H_1}^{simulation})$$

$$\iff d_{H_0}^{simulation} \sim d_{H_1}^{simulation}$$

- So, we have:

$$\forall\{d, p^A\}/ \quad \left.\begin{array}{l} d_{H_0}^{simulation} = f_{H_0}(d, p^A) \\ d_{H_1}^{simulation} = f_{H_1}(d, p^A) \end{array}\right\}$$

$$\Rightarrow \quad d_{H_0}^{simulation} \sim d_{H_1}^{simulation}$$

We also know that two equivalent data are not necessarily equal to each other ($d \sim d' \quad \not\Rightarrow \quad d = d'$). Therefore, it exists a case where two *Model data* are equivalent and not equal. So:

$$\exists\{d, f^{p^{simulation}}, p^A\}/ \quad \left.\begin{array}{l} d_{H_0}^{simulation} = f_{H_0}^{p^{simulation}}(d, p^A) \\ d_{H_1}^{simulation} = f_{H_1}^{p^{simulation}}(d, p^A) \end{array}\right\}$$

$$d_{H_0}^{simulation} \sim d_{H_1}^{simulation} \; and \; d_{H_0}^{simulation} \neq d_{H_1}^{simulation}$$

So, if we produce $d_{H_0}^{simulation}$ once more on each side we obtain the following result on our model:

$$\exists\{d, f^{p^{simulation}}, p^A\}/ \quad \left.\begin{array}{l} d_{H_0}^{simulation} = f_{H_0}^{p^{simulation}}(d, p^A) \\ d_{H_1}^{simulation} = f_{H_1}^{p^{simulation}}(d, p^A) \end{array}\right\}$$

$$d_{H_0}^{simulation} \cup d_{H_1}^{simulation} \neq d_{H_0}^{simulation} * 2$$

We now have proof that one of the main interesting aspects of a scalable simulation is not in opposition to our model. With our model, we can produce the activity of n *Hosts* that would be different from doing the same activity n times simultaneously on a single *Host*.

The other aspect to consider with scalability is the network infrastructure. To achieve a scalable simulation, we must impose some requirements on the network support:

Property 5 (Implementation requirement 2): The network infrastructure supporting the simulation model must be favorable to scalability. It should:

- Use virtual hosts: it is not possible to have a scalable network with physical hosts so our network support must use virtual hosts.
- Have low setup time and resources requirements: resources consumption and the workforce required for setting up a vast network are what often prevent evaluation data methods from being scalable.

Following the second requirement, we aim to make our simulation model work on a virtual network using process VMs. We detail the implementation of a prototype in the following section.

V. Implementation of the proposed method

In this section, we describe the implemention of a prototype that follows the requirements of our model. This prototype uses Mininet [24] as the network support of our simulation. Mininet is an open-source network simulator that deploys lightweight virtual machines to create virtual networks, and able to create hundreds of lightweight virtual machines in a short amount of time.

A. Model of the prototype

Our prototype contains several *Data generating functions* that preserve each of these properties: execution time, packet size, acknowledgement of the data by the *Service*. Based on these *Data generating functions* we simulate the activity of 50 to 200 *Hosts* representing regular employees of a small company interacting with the *Service* of a webmail server Roundcube on a Postfix mail server. A simulation control program follows the *Script* described in Figure 4 for all the *Hosts* of the simulation. In Figure 4, the *Elementary actions* are in italics while actions that do not generate activity data are in a regular font. The *Host* can simulate two different series of *Elementary actions* after a waiting period of X seconds each time. The intensity of the *Script* can be modulated by modifying the value of X.

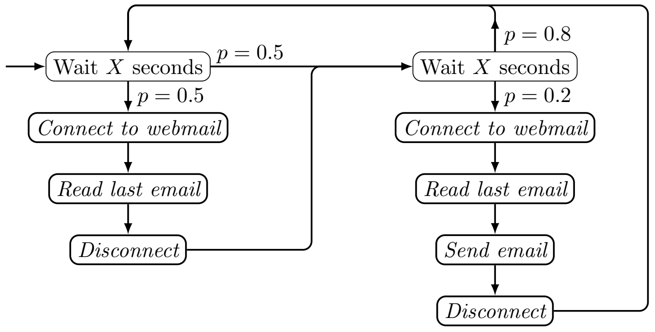


Figure 4. Generation of simulated activity from short traces

To make sure that our method improves the existing methods, we must verify that our implementation respects the properties we highlighted:

- the verification property of *Data generating functions* (c.f. Property 1)
- the reproducibility property of *Data generating functions* (c.f. Property 2)
- the verification property of the *Scenario* (c.f. Property 3)
- the implementation requirement on *Control data* (c.f. Property 4)
- the implementation requirement for scalability (c.f. Property 5)

TABLE I. NUMBER OF LINES IN THE WEBMAIL LOG FILES.

	5 VMs		**5 Hosts**		**50 Hosts**		**100 Hosts**		**150 Hosts**		**200 Hosts**		**250 Hosts**	
Filenames	**avg**	**stdev**	**avg**	**stdev**	**avg**	**stdev**	**avg**	**stdev**	**avg**	**stdev**	**avg**	**stdev**	**avg**	**stdev**
userlogins	90	9	112	10	1032	36	2084	45	3085	52	4121	53	5118	74
imap	43245	5070	57775	5306	487883	22742	984642	28820	1450507	27792	1933823	21117	274825	235985
sql	4955	525	6703	563	56081	1886	113031	2452	167138	2964	223427	2906	265354	4688

B. *Verification of the implementation requirements*

The verification of the implementation requirements (Properties 4 and 5) are the easiest to verify.

We select the network simulator Mininet in accordance to Property 5: it uses lightweight virtual machines and can create a large amount of virtual hosts (around a thousand) connected with virtual links in a matter of minutes on a regular computer. Mininet uses the lightweight virtualization mechanisms built into the Linux OS: processes running in network namespaces, and virtual Ethernet pairs. Mininet can emulate links, hosts, switches, and controllers at a very low resource cost.

Concerning the Property 4, we simply ask the simulation control program to write in a file the information required by that property.

C. *Experiments on the prototype*

We verify the other properties with two separate experiments.

The first experiment is a control experiment. We deploy 5 virtual machines on the network simulator Hynesim [25] and make them generate the activity of our simulation. We script the *Elementary actions* of the *Script* described in Figure 4 with the web driver Selenium [26] and make the virtual machines use their browser to interact with the webmail server. This experiment provides referential values for our second experiment. We expect proportionality between these values and the results of our simulation, with respect to the number of *Hosts*.

In the second experiment, we simulate different number of *Hosts* (5, 50, 100, 150, 200 and 250) and make them generate the activity of regular users using a webmail service for 30 minutes. We measure the activity at three different points: the webmail server, the network simulator Mininet and the server hosting the simulation. Every 30 seconds, we measure four parameters: CPU usage, memory usage, network I/O, and disk I/O. Figure 5 is an example of the measured activity. It represents the network traffic received and sent by the webmail server with 50 simulated *Hosts*. Each *Host* follows the *Script* described in Figure 4, with $X = 30$.

We also retrieve the logs produced by the webmail server during both experiments. The quantity and content of the logs is analyzed in Table I and Table II. Both experiments are done twenty times for each set of parameters to ensure the consistency of the results (Property 2). In the results we express the average value and standard deviation of those twenty experiments.

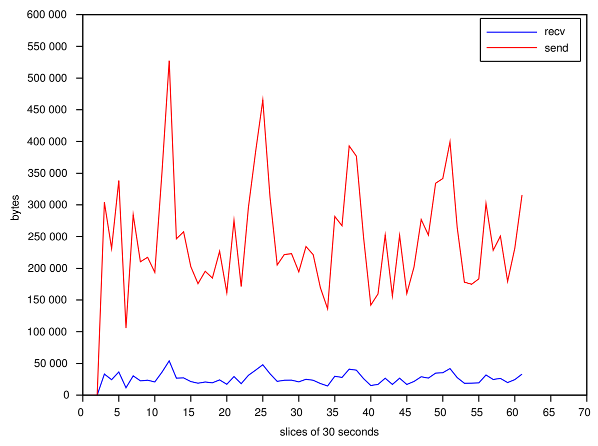


Figure 5. Network traffic of the webmail server for a single experiment (50 *Hosts*)

In the second experiment, we use the *Data generating function* with the highest level of realism: the adapted replay. That *Data generating function* preserves the data acknowledgement by the *Service* and allows *Elementary action parameters* to modify the inputs of submitted forms. Concretely, it means that a server cannot distinguish the adapted replay from an interaction with a real user. Also, with the help of *Elementary action parameters*, the evaluator can freely change the credentials replayed to the webmail server.

The analysis of the logs aims to verify the Properties 1 and 3. We verify that the Postfix server correctly accepted the data generated by the *Data generating function* for each *Elementary actions* (Property 1) and that the logs of the simulation reflects our estimation from the logs of the control experiment (Property 3). We also verify that the results are consistent over multiple instances (Property 2).

Table I represents the quantity of logs produced by the webmail server during both experiments. We express the average number of lines in the log files of the webmail and their standard deviation. The first column is the name of the main log files produced by the server: ”userlogins” logs every connection (successful or not), ”imap” logs every instruction from the server that uses the IMAP protocol, and ”sql” logs every interaction between the server and its database. The entries under the name ”5 VMs” correspond to the results of the control experiment while the other entries are the results of the simulation experiment.

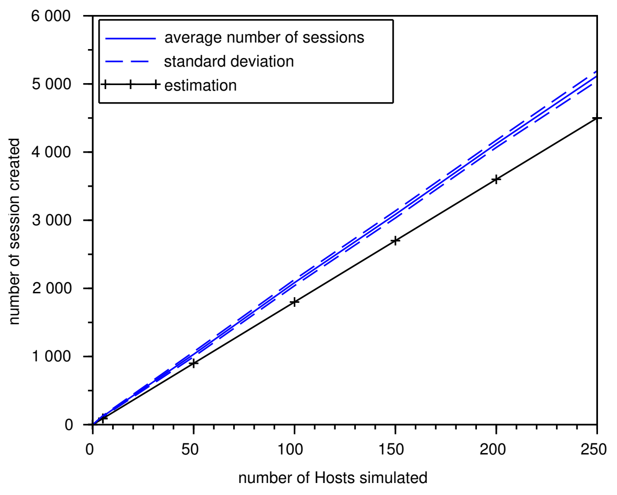


Figure 6. Number of sessions created during simulation (blue) compared to estimation (black)

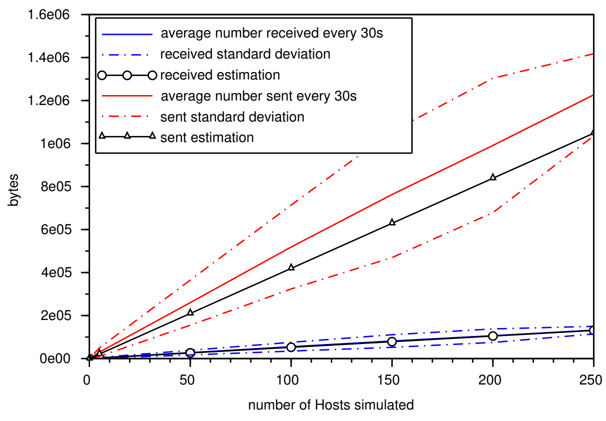


Figure 7. Network traffic of the webmail server

The number of lines in ”userlogins” represents the number of connections during the experiments (one line per connection) and can be used to calculate the number of sessions created during both experiments. Figure 6 shows the average number of sessions created during the second experiment and its standard deviation according to the number of simulated *Hosts*. We also estimate the average number of sessions inferred from the results of the control experiment, based on proportionality ($avg(\text{”5 VMs”}) \times \frac{number\ of\ Hosts}{5}$).

We observe that the number of sessions created during the second experiment is close to our estimation. Our simulation produces more sessions than expected but it can be explained by the fact that our *Data generating function* reproduces the *Model data* of an *Elementary action* faster than the browser of the virtual machines. Hence, in a period of 30 minutes, the simulated activity has gone through more cycles of the *Script* than the control experiment. A projection of the number of lines of the other log files (”imap” and ”sql”) displays similar results.

These results establish that the simulated activity produces a consistent amount of logs. In Figure 7, we examine the network traffic produced by our simulated activity. The blue and red parts represent the average number of bytes, respectively, received and sent by the webmail server every 30 seconds, along with the standard deviation. For comparison, the black lines correspond to the estimation of the expected results based on the control experiment. As before, the results of the second experiment are close to our estimation. The deviation can be justified with the same explanation regarding the activity speed difference. This deviation is also partly due to the cached data. Since these data are stored on the host after the first connection, the amount of exchanged data during the first connection is higher than during subsequent sessions.

However, our *Data generating function* does not take cached data into account. Therefore, our simulated connections request more data from the webmail server than estimated. This observation is part of the parametrization issues of the *Data generating function* raised at the end of Section IV. Adding *Elementary action parameters* to modify the behavior of the function can solve this issue as we did for previous typing and semantic issues. However, the addition of new *Elementary action parameters* is made from empiric observation and could be improved by adding new methods to our model like machine learning.

Despite those issues, we have shown that the simulated activity of the second experiment generated a large network activity proportionally to the number of simulated *Hosts*, as expected. We now focus on proving that the activity semantics was also preserved.

For each *Elementary action* of the activity *Script*, we look for log entries that could act as signatures for the action. We select those signatures by comparing the logs of the different *Elementary actions*. The log entries that appeared for only one *Elementary action* are selected as signature of that action.

Those log entries are used to verify that the server acknowledges the simulation data as it would real actions. We also manually verified the correctness of the *Data generating function* for some *Elementary actions*. For example, we ask the simulation to do the action ”read the email” for a different email that the one in the *Model data*. Or to do the action ”connect to the webmail” with purposely wrong credentials. In both cases, the manual analysis of the simulated data showed that the webmail server properly acknowledged the simulated data.

Signatures from log entries is a more global form of verification. By comparing these signatures in both experiments, we obtain the results displayed in Table II.

From Table II, the following observations can be made:

- the number of signatures for the ”connect” *Elementary*

TABLE II. Signature log entries.

		5 VMs		**5 Hosts**		**50 Hosts**		**100 Hosts**		**150 Hosts**		**200 Hosts**		**250 Hosts**	
Signatures	**Actions**	**avg**	**stdev**	**avg**	**stdev**	**avg**	**stdev**	**avg**	**stdev**	**avg**	**stdev**	**avg**	**stdev**	**avg**	**stdev**
imap.sign1	connect	90	9	122	10	1032	36	2079	44	3075	55	4118	54	4874	87
imap.sign2	connect	90	9	122	10	1032	36	2079	44	3075	55	4118	54	4874	87
imap.sign3	connect	90	9	122	10	1032	36	2079	44	3075	55	4118	54	4874	87
imap.sign4	connect	90	9	122	10	1032	36	2079	44	3075	55	4118	54	4874	87
imap.sign5	connect	90	9	122	10	1032	36	2079	44	3075	55	4118	54	4874	87
imap.sign6	connect	90	9	122	10	1032	36	2079	44	3075	55	4118	54	4874	87
imap.sign7	connect	90	9	122	10	1032	36	2079	44	3075	55	4118	54	4874	87
imap.sign8	connect	90	9	122	10	1032	36	2079	44	3075	55	4118	54	4874	87
imap.sign9	connect	90	9	122	10	1032	36	2079	44	3075	55	4118	54	4873	88
imap.sign10	connect	90	9	122	10	1032	36	2079	44	3075	55	4118	54	4873	88
sql.sign1	connect	90	9	122	10	1032	36	2079	44	3075	55	4118	54	4874	87
sql.sign2	disconnect	90	9	122	10	1032	36	2079	44	3085	52	4121	53	5118	74
imap.sign11	open	89	9	122	10	1028	36	2069	44	3059	52	4090	53	4808	91

action is slightly inferior to the number of sessions (the number of lines from ”userlogins”) observed for 150 *Hosts* and above. It is explained by the fact that the signatures correspond to the number of successful connections to the webmail server. If we remove the number of lines in the ”userlogins” file that correspond to failed connections, we find the exact number of signatures for the ”connect” *Elementary action*.

- the number of signature for the ”disconnect” *Elementary action* corresponds to the exact number of sessions observed in Table I.
- the number of signatures for the ”open” *Elementary action* is slightly inferior to the number of signatures for the ”connect” *Elementary action* for 50 *Hosts* and above. It is likely due to the experiment ending before the last *Script* cycle ended for a few *Hosts*.
- no characteristic entry for the ”send an email” *Elementary action* could be found in the ”userlogins”, ”imap” and ”sql” log files.

The failure of several connections in our simulation may also be due to the parameterization of the *Data generating function*. The adapted replay *Data generating function* was designed to modify short-lived information from the *Model data* like the token or the session identifier according to the server reply from the requests. However, such modification was not included in the first request. The webmail server possibly refused some connections because they contained the same information at the same time. Therefore, an improvement of the typing of the adapted replay *Data generating function* should raise the number of successful connections with a high number of simulated *Hosts*. Table II shows that for each successful session in our simulated activity, the webmail server correctly interpreted the *Elementary actions*.

To sum up the results analysis, our prototype generates a simulated activity that produces a realistic amount of network traffic and logs from the webmail server (Property 3). Moreover, the webmail server produces the appropriate number of logs reflecting the correct semantics. Each simulated *Elementary action* resulted in the same log entries as a real one (Property 1). Each result was verified twenty times (Property 2). Therefore, our prototype succeeds in providing scalable and realistic data generation, thus validating our model.

VI. Conclusion

In this paper, we establish a new methodology to generate realistic evaluation data on a network support (Mininet) with far fewer requirements than the common network testbeds. This methodology takes into consideration the need for an evaluator to test different properties and evaluate different vulnerabilities in a security product. Therefore, an evaluator can select the *Data generation function* that matches the properties of the product that need to be tested. The evaluator also has a control on the granularity of the activity *Elementary actions*. The finesse of the simulated activity can be improved by introducing new *Elementary actions* or adding *Elementary action parameters* to the *Data generation function*.

We add to our previous published paper several aspects. First, we elaborate the ambition of our method into five identified requirements: customizability, reproducibility, realism, accuracy, and scalability. Second, we explain with greater details the different concepts of our methods and translate our added requirements into verifiable properties of our model. It allows us to introduce the concepts of levels of realism with clear example of the current capacities of our model and its future potential. We validate our model with a prototype able to generate realistic activity up to 250 users interacting with a webmail server. The traffic can be customized in terms of *Hosts* numbers as well as *Scripts* content. With our added verifiable properties of the five requirements of our model, we define experimental tests to make sure our prototype complies with our ambitions. In another article [27], we use this prototype to define evaluation methodologies and evaluate a security product: the intrusion detection system Suricata.

However, our prototype still has a few limitations. The existing *Data generating functions* mostly focus on the creation of network activity and does not generate system activity for host-based security products. The parametrization for more realistic *Data generating functions* also raises additional issues that need to be addressed with further work. Finally, our prototype is currently limited to the simulation of *Hosts*. In parallel with the testing of network-based intrusion detection systems based on our prototype, the next steps of our work will focus on extending our prototype to include the simulation of *Services* and develop new *Data generating functions* that focus on the generation of system data rather than network data.

References

[1] P.-M. Bajan, H. Debar, and C. Kiennert, “A new approach of network simulation for data generation in evaluating security products,” in ICIMP 2018, The Thirteenth International Conference on Internet Monitoring and Protection, 2018.

[2] A. Milenkoski, M. Vieira, S. Kounev, A. Avritzer, and B. D. Payne, “Evaluating computer intrusion detection systems: A survey of common practices,” ACM Computing Surveys (CSUR), vol. 48, no. 1, 2015, p. 12.

[3] C. Kreibich and J. Crowcroft, “Honeycomb: creating intrusion detection signatures using honeypots,” ACM SIGCOMM computer communication review, vol. 34, no. 1, 2004, pp. 51–56.

[4] V. E. Seeberg and S. Petrovic, “A new classification scheme for anonymization of real data used in ids benchmarking,” in Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on. IEEE, 2007, pp. 385–390.

[5] S. E. Coull et al., “Playing devil’s advocate: Inferring sensitive information from anonymized network traces.” in NDSS, vol. 7, 2007, pp. 35–47.

[6] A. Srivastava, K. Singh, and J. Giffin, “Secure observation of kernel behavior,” Georgia Institute of Technology, Tech. Rep., 2008.

[7] F. Lombardi and R. Di Pietro, “Secure virtualization for cloud computing,” Journal of Network and Computer Applications, vol. 34, no. 4, 2011, pp. 1113–1122.

[8] J. Reeves, A. Ramaswamy, M. Locasto, S. Bratus, and S. Smith, “Intrusion detection for resource-constrained embedded control systems in the power grid,” International Journal of Critical Infrastructure Protection, vol. 5, no. 2, 2012, pp. 74–83.

[9] K. Nasr, A. Abou-El Kalam, and C. Fraboul, “Performance analysis of wireless intrusion detection systems,” in International Conference on Internet and Distributed Computing Systems. Springer, 2012, pp. 238–252.

[10] K. Ma, R. Sun, and A. Abraham, “Toward a lightweight framework for monitoring public clouds,” in Computational Aspects of Social Networks (CASoN), 2012 Fourth International Conference on. IEEE, 2012, pp. 361–365.

[11] J.-K. Ke, C.-H. Yang, and T.-N. Ahn, “Using w3af to achieve automated penetration testing by live dvd/live usb,” in Proceedings of the 2009 International Conference on Hybrid Information Technology. ACM, 2009, pp. 460–464.

[12] F. Massicotte, M. Couture, Y. Labiche, and L. Briand, “Context-based intrusion detection using snort, nessus and bugtraq databases.” in PST, 2005.

[13] N. J. Puketza, K. Zhang, M. Chung, B. Mukherjee, and R. A. Olsson, “A methodology for testing intrusion detection systems,” IEEE Transactions on Software Engineering, vol. 22, no. 10, 1996, pp. 719–729.

[14] R. Riley, X. Jiang, and D. Xu, “Guest-transparent prevention of kernel rootkits with vmm-based memory shadowing,” in International Workshop on Recent Advances in Intrusion Detection. Springer, 2008, pp. 1–20.

[15] H. Jin et al., “A vmm-based intrusion prevention system in cloud computing environment,” The Journal of Supercomputing, vol. 66, no. 3, 2013, pp. 1133–1151.

[16] J. Morris, S. Smalley, and G. Kroah-Hartman, “Linux security modules: General security support for the linux kernel,” in USENIX Security Symposium, 2002, pp. 17–31.

[17] M. Laureano, C. Maziero, and E. Jamhour, “Protecting host-based intrusion detectors through virtual machines,” Computer Networks, vol. 51, no. 5, 2007, pp. 1275–1283.

[18] P. Mell, V. Hu, R. Lippmann, J. Haines, and M. Zissman, “An overview of issues in testing intrusion detection systems,” NIST Interagency, Tech. Rep., 2003.

[19] R. K. Cunningham, R. P. Lippmann, D. J. Fried, S. L. Garfinkel, I. Graf, K. R. Kendall, S. E. Webster, D. Wyschogrod, and M. A. Zissman, “Evaluating intrusion detection systems without attacking your friends: The 1998 darpa intrusion detection evaluation,” Massachusetts Inst. of Tech. Lexington Lincoln Lab, Tech. Rep., 1999.

[20] T. V. Phan, N. K. Bao, and M. Park, “Distributed-som: A novel performance bottleneck handler for large-sized software-defined networks under flooding attacks,” Journal of Network and Computer Applications, vol. 91, 2017, pp. 14–25.

[21] C. Cowan, S. Arnold, S. Beattie, C. Wright, and J. Viega, “Defcon capture the flag: Defending vulnerable code from intense attack,” in DARPA Information Survivability Conference and Exposition, 2003. Proceedings, vol. 1. IEEE, 2003, pp. 120–129.

[22] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, “Mawilab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking,” in Proceedings of the 6th International COnference. ACM, 2010, p. 8.

[23] Z. Puljiz and M. Mikuc, “Imunes based distributed network emulator,” in Software in Telecommunications and Computer Networks, 2006. SoftCOM 2006. International Conference on. IEEE, 2006, pp. 198–203.

[24] R. L. S. De Oliveira, A. A. Shinoda, C. M. Schweitzer, and L. R. Prete, “Using mininet for emulation and prototyping software-defined networks,” in Communications and Computing (COLCOM), 2014 IEEE Colombian Conference on. IEEE, 2014, pp. 1–6.

[25] “Homepage of Hynesim,” 2018, URL: https://www.hynesim.org [accessed: 2018-04-09].

[26] R. A. Razak and F. R. Fahrurazi, “Agile testing with selenium,” in Software Engineering (MySEC), 2011 5th Malaysian Conference in. IEEE, 2011, pp. 217–219.

[27] P.-M. Bajan, H. Debar, and C. Kiennert, “Methodology of a network simulation in the context of an evaluation: Application to an ids,” in ICISSP 2019, The Fith International Conference on Information Systems Security and Privacy, 2019.

Power Consumption Analysis of the New Covert Channels in CoAP

Aleksandar Velinov, Aleksandra Mileva, Done Stojanov
Faculty of Computer Science
University “Goce Delčev”
Štip, Republic of Macedonia
email: {aleksandar.velinov, aleksandra.mileva, done.stojanov}@ugd.edu.mk

***Abstract* — This paper presents several novel covert channels in the Constrained Application Protocol (CoAP) - a specialized Web transfer protocol used for Internet of Things. The suggested new covert channels are categorized according to the pattern-based classification, and, for each covert channel, the total number of hidden data bits transmitted per second, or per protocol data unit is given, together with the theoretical performance evaluation. Additionally, we present a methodology for power consumption analysis of these covert channels, and we give the experimental results of applying this methodology for one of the discovered CoAP covert channels.**

Keywords - CoAP; network steganography; Contiki OS; Cooja; Copper.

I. Introduction

We have investigated several novel covert channels for the Constrained Application Protocol (CoAP) in the conference paper [1]. In this paper, we extend the results with an experimental methodology for power consumption analysis of these covert channels, and we give the experimental results of applying this methodology for one of the discovered CoAP covert channels.

Network steganography is a practice of hiding data in legitimate transmissions in communication networks, by deploying different network protocols as carriers, and concealing the presence of hidden data from network devices. It provides only security through obscurity.

Covert channels are first introduced by Lampson [10] as channels “not intended for information transfer at all” and they can be exploited by a process to transfer information in a manner that violates the systems security policy. The current distinction between the network steganography and covert channels is artificial, especially in a communication networks environment. Network steganography techniques create covert channels for hidden communication, but such covert channels do not exist in communication networks without steganography [14]. There is no some algorithm for exaustive search of all covert channels in a given protocol.

Covert channels can be divided in two basic groups: storage and timing channels. Storage covert channels are channels where one process writes (directly or indirectly) to a shared resource, while another process reads from it. In the context of network steganography, storage covert channels hide data by storing them in the protocol header and/or in the Protocol Data Unit (PDU). On the other hand, timing channels hide data by deploying some form of timing of events, such as retransmitting the same PDU several times, or changing the packet order.

Network-based covert channels may have black hat or white hat applications. Black hat applications include coordination of distributed denial of service attacks, spreading of malware (for example, by hiding command and control traffic of botnets), industrial espionage, secret communication between terrorists and criminals, etc. On the other hand, white hat applications include covert military communication in hostile environments, prevention of detection of illicit information transferred by journalists or whistle-blowers, circumvention of the limitation in using Internet in some countries (e.g., Infranet [4]), providing Quality of Service - QoS for Voice over Internet Protocol - VoIP traffic [12], secure network management communication [6], watermarking of network flows (e.g., RAINBOW [8]), tracing encrypted attack traffic or tracking anonymous peer-to-peer VoIP calls [21][22], etc.

Nowadays, there are a plenty of choices in the landscape of network protocols for carriers. There are several surveys about different covert channels in many TCP/IP (Transmission Control Protocol/Internet Protocol) protocols [15][26]. To the best of our knowledge, there are only a few papers about network steganographic research addressing protocols specialized for constrained devices in the IoT (sensors, vehicles, home appliances, wearable devices, and so on) [3] [9]. The Constrained Application Protocol (CoAP) [19] is a specialized Web transfer application layer protocol, which can be used with constrained nodes and constrained networks in the IoT. The nodes are constrained because they have 8-bit microcontrollers, for example, with limited random-access memory (RAM) and read-only memory (ROM). Constrained networks often have high packet error rates and small data rate (such as IPv6 over Low-Power Wireless Personal Area Networks - 6LoWPANs). CoAP is designed for machine-to-machine (M2M) applications and its last stable version was published in June 2014 in the RFC 7252 [19]. In fact, it is a Representational State Transfer - RESTful protocol with multicast and observe support. In this paper, we try to apply existing network steganographic techniques for creating covert channels in CoAP.

Wendzel et al. [24] presented a new pattern-based categorization of network covert channel techniques into 11 different patterns or classes. They represented the patterns in a hierarchical catalog using the pattern language Pattern

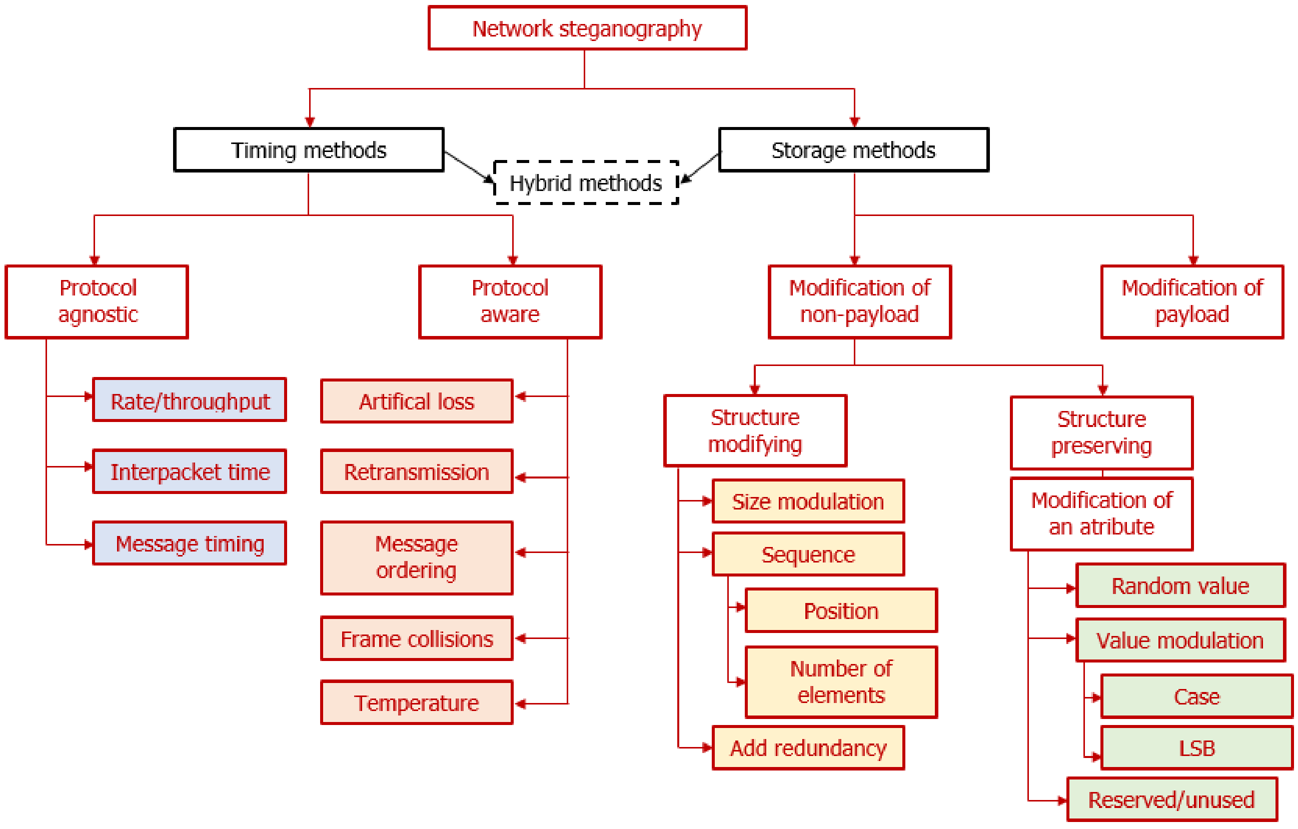


Figure 1. Pattern-based categorization of network covert channel techniques

Language Markup Language (PLML) v. 1.1 [5]. A modification of this categorization is made by Mazurczyk et al. [14]. In our paper, we use this joint classification (see Figure 1) to characterize our covert channels.

Covert channels are analyzed through the total number of hidden data bits transmitted per second (Raw Bit Rate - RBR), or through the total number of hidden data bits transmitted per PDU (for example, Packet Raw Bit Rate-PRBR) [13]. For each new CoAP channel, its PRBR value is given, where PDU is a CoAP message.

The rest of this article is structured as follows. The related work is presented in Section II. Details about the CoAP header, messages, functionalities and concepts are presented in Section III. The main Section IV describes eight groups of new covert storage and timing channels in CoAP, that can be used regardless its transport carrier (DTLS or clear UDP). Some possible applications of these covert channels are also briefly suggested in this section. In Section V we present the performance evaluation, while the experimental evaluation of one of the new covert channels is given in Section VI. We conclude the paper in Section VII.

II. Related Work

The research on network steganography for IoT has seen an increased interest recently.

One example for this is the work of Islam et al. [9], which uses Internet Control Message Protocol (ICMP) covert channels for authenticating Internet packet routers as an intermediate step towards proximal geolocation of IoT devices. This is useful as a defense from the knowledgeable adversary that might attempt to evade or forge the geolocation. Hidden data are stored in the data field of the ICMP Echo Request and ICMP Echo Reply messages.

Patuck et al. [18] present several storage covert channels in the Extensible Messaging and Presence Protocol (XMPP), a popular instant messaging protocol based on XML, which in the past was used by many messaging platforms such as Google Talk, AOL Instant Messenger, Microsoft Messenger Sevice, etc. These covert channels use some attributes in the XMPP messages, like Type, id and xml:lang attributes, or the message body. For example, for the Type attribute, three covert channels are presented: by changing cases of the value, by changing value, or by presence/absence of the attribute.

A storage covert channel with modulated sensor readings is presented by Tuptuh et al. [20] for wireless sensor networks. In this channel, LSBs of encrypted sensor readings are the cover bits. The sender performs the following algorithm: while LSB bit of the current reading is different from the cover bit, small offset is added to the sensor reading (e.g., temperature) and the value is encrypted.

Building Automation and Control Networking Protocol (BACnet) is another protocol for which two storage (message-type based and parameter-based) and one timing (with inter packet gaps) covert channels were given by Wendzel [23].

Wendzel et al. [25] have showed that even a cyber-physical system (CPS) can be used for network steganography. One can places hidden data in the CPS environment by slightly modifying some of its components, like actuators, sensors, controllers, and monitoring equipment. The authors apply the term scatter hoarding, which means that only small modifications of the CPS will be allowed but they will be applied to numerous carefully selected components, to avoid them being regularly modified, e.g., by a human user. One example is the temperature sensor, which comprises two 8-bit alarm registers with a lower and an upper warning threshold, and can be used to store hidden values. Another example is the state modulation of actuators, like heater, in which the heating value of 80% will be a binary "0", and of 79% will be a binary "1". Because the actuator states change and influence the physical environment, steganographic operations may not be robust and be easily detectable and thus need a reasonable storage strategy.

Some applications of steganography in IoT are not connected with the protocols themselves, but with the applications on top of these protocols. For example, Denney et al. [3] present a novel storage covert channel on wearable devices that sends data to other applications, or even to other nearby devices, through the use of notifications that are normally displayed on the status bar of an Android device. For that purpose, a notification listening service on the wearables needs to be implemented. Data are hidden in the notification ID numbers (32 bits), and their exchange is done by using two functions notify and cancel. If the notifying function is immediately followed by the canceling function, the notification is never displayed to the user although it can be seen in the log files, so the communication is hidden from the user who wears the device.

There are several papers that deploy steganography in the physical or medium access control (MAC) layers [7][11][16][19].

As far as we now, there is no paper (other than [1]) that analyze existance of covert channels in CoAP. Additionally, we try to give a methodology how one can perform a power consumption analysis of a given covert channel in the IoT device.

III. How CoAP Works

Similar to HTTP, CoAP uses client/server model with request/response messages. It supports built-in discovery of services and resources, Uniform resource identifiers (URIs) and Internet media types. The CoAP sends request message requesting an action (using a Method Code) to the resource (idenified by a URI) hosted on server. The server responds to this request by using the response message that contains the Response Code, and possibly some resource representation. CoAP defines four types of messages: Confirmable (CON), Non-Confirmable (NON), Acknowledgment (ACK) and Reset (RST). These types of messages use method and response codes to transmit requests or answers. The requests can be transmitted as Confirmable and Non-Confirmable types of messages, while the responses can be transmitted through these and via piggybacked and Acknowledgment types of messages.

CoAP uses clear UDP or DTLS on transport layer to exchange messages asynchronously between endpoints. As shown in Figure 2, each message contains a Message ID used for optimal reliability and to detect duplicates. A message that requires reliable transmission is marked as CON, and if does not, it is marked as NON. The CON message is retransmitted using a default timeout and binary exponential back-off algorithm for increasing the timeout between retransmissions, until the recipient sends an ACK message with the same Message ID. When the recipient is not able at all to process CON or NON message, it replies with a RST message.

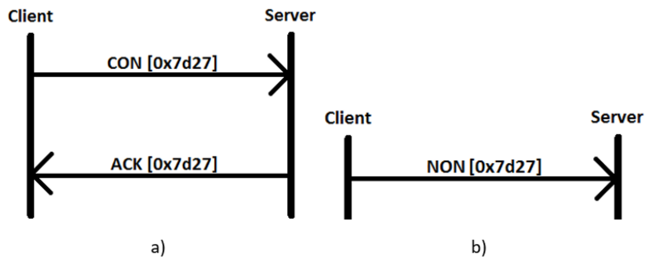


Figure 2. a) Reliable CoAP message transmission b) Unreliable CoAP message transmission.

CoAP messages are encoded into simple binary format (see Figure 3). Each message starts with a 4B fixed header, followed by a Token field, with size from 0 to 8B. Then comes the optional Options field and optional Payload field. If the Payload field is present it is preceded by one-byte Payload Marker (0xFF).

The fields that make up the message header are the following:

- Version (Ver) - 2-bit unsigned integer that idenitfies the CoAP version. Currently it must be set to 01.
- Type (T) – 2-bit unsigned integer that indicates the message type: Confirmable (0), Non-Confirmable(1), Acknowledgement (2), or Reset (3).
- Token Length (TKL) – 4-bit unsigned integer that stands for the length of the Token field (0-64 bits). Lengths 9-15 are reserved and must be processed as a message format error.
- Code – 8-bit unsigned integer. It is divided into two parts: 3-bit class (the most significant bits) and 5-bit details (the least significant bits). The format of the code is "c.dd", where "c" is a digit from 0 to 7 and represents the class while "dd" are two digits from 00 to 31. According to the class we can determine the type of the message, such as: request (0), a successful response (2), a client error response (4), or a server error response (5).

CoAP has a separate code registry that provides a description for all codes [2].

- Message ID - 16-bit unsigned integer that is used to detect duplicate messages and to connect Acknowledgment/Reset messages with Confirmable/ Non-Confirmable messages.

The message header is followed by the Token field with variable size from 0 to 64 bits. This field is used to link requests and responses.

The optional Options field defines one or more options. CoAP defines a single set of options that are used both for requests and for responses. These are: Content-Format, Etag, Location-Path, Location-Query, Max-Age, Proxy-Uri, Proxy-Scheme, Uri-Host, Uri-Path, Uri-Port, Uri-Query, Accept, If-Match, If-None-Match, and Size1.

The payload of requests/responses that indicates success typically carries the resource representation or the result of the requested action.

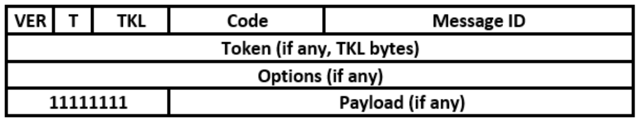


Figure 3. CoAP message format.

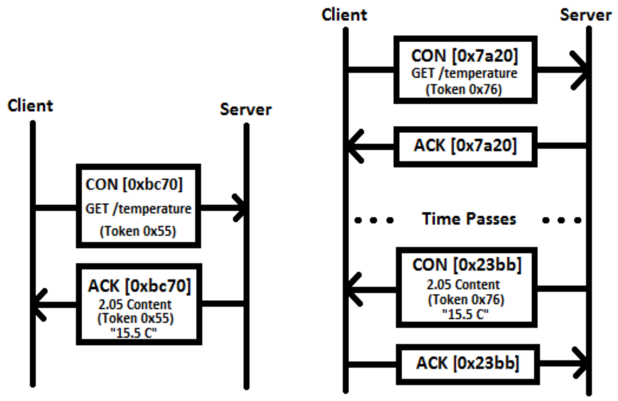


Figure 4. a) Piggybacked response b) Separate response.

There are two types of responses: piggybacked and separate (Figure 4). If the request is transmitted via CON or NON message, and if the response is available and transmitted via an ACK message, then it is piggybacked response. If the server is unable to respond immediately to the request, an Empty message (with code 0.00) is sent that tells the client to stop sending the request. If the server is able for later respond to the client, it sends a CON message that must then be confirmed by the client. This is called a separate response.

Similar to HTTP, CoAP uses GET (with code 0.01), POST (with code 0.02), PUT (with code 0.03), and DELETE (with code 0.04) methods.

IV. New Covert Channels in the CoAP

When someone creates a covert channel (CC) in network protocol, usually uses: a protocol feature that has a dual nature (i.e., the same feature can be obtained in more than one way), a feature that is not mandatory, a feature that can obtain random value, and so on. Therefore if we use some of these features, we can create new covert channels in CoAP. From the beginning, CoAP offers some protection against network steganography. For example, by introducing a proper order in the appearance of different options in message, the steganographic techniques that deploy different order of options can not be applied.

CoAP can be applied in different fields, such as: smart energy, smart grid, building control, intelligent lighting control, industrial control systems, asset tracking, environment monitoring, and so on. So, one useful scenario of application of the CoAP covert channels would be for support of the authentication of geolocation of IoT devices. Another possible scenario is clandestine communication between wearable devices in a hostile environment, for the needs of the soldiers, or, between nodes in a wireless sensor network.

As steganography offers only security through obscurity, a successful attack against any covert channel consists in detecting the existence of this communication. Next, the new CoAP covert channels are presented.

A. Covert Channel Using Token and/or Message ID Fields

The Message ID contains a random 16-bit value. In the case of piggybacked response for CON message, the Message ID should be the same as in the request, while in the case of separate response, the server generate different random Message ID (while the request Message ID is copied in the first sent Empty ACK message).

The same Message ID can not be reused (in the communication between same two endpoints) within the EXCHANGE_LIFETIME, which is around 247 seconds with the default transmission parameters.

The Token is another random generated field, with variable size up to 64 bits, used as a client-local identifier to make a difference between concurrent requests. If the request results in the response, the Token value should be echoed in that response. This also happens in the case when the server sends separate response. So, we can create an unidirectional or a bidirectional communication channel between two hosts, by sending 16 (from Message ID) plus/or 64 (from Token ID) bits per message (PRBR $\in \{16, 64, 80\}$). According to the pattern-based classification [14][24], this channel belongs to the following class:

```
Network Covert Storage Channels
 --Modification of Non-Payload
  --Structure Preserving
    --Modification of an Attribute
     --Random Value Pattern
```

B. *Covert Channel Using Piggybacked and Separate Response*

Since the server has a choice for sending piggybacked or separate response, one can create an one-bit per message unidirectional or a bidirectional covert channel (PRBR=1), such as:

- piggybacked response to be binary 1, and
- separate response to be binary 0.

At heavy load, the server may not be able to respond (sending binary 1), so this covert channel is limited to the times when the server has the choice. According to the pattern-based classification [14][24], this channel belongs to the following class:

```
Network Covert Timing Channels
 --Protocol aware
  --Message ordering pattern
```

C. *Covert Channel Using Payload of the Message*

Both requests and responses may include a payload, depending of the Method or the Response Code, respectively. Its format is specified by the Internet media type and content coding providen by the Content-Format option. The payload of requests or of responses that indicates success is typically a representation of the resource or the result of the requested action.

If no Content-Format option is given, the payload of responses indicating client or server error is a Diagnostic Payload, with brief human-readable diagnostic message being encoded using UTF-8 (Unicode Transformation Format) in Net-Unicode form.

The CoAP specification provides only an upper bound to the message size - to fit within a single IP datagram (and into one UDP payload). The maximal size of the IPv4 datagram is 65,535B, but this can not be applied to constrained devices and networks. According to IPv4 specification in the RFC 791, all hosts have to be prepared to accept datagrams of up to 576B, while IPv6 requires the maximum transmission unit (MTU) to be at least 1280B. The absolute minimum value of the IP MTU for IPv4 is 68B, which would leave at most 35B for a CoAP payload (the smallest CoAP header size with Payload Marker before the payload is 5B, assuming 0B for Token and no options). On the other hand, constrained network presents another restriction. For example, the IEEE 802.15.4's standard packet size is 127B (with 25B of maximum frame overhead), which leaves (without any security features) 102B for upper layers. The sizes of the input/output buffers in the constrained devices are another restriction of the maximal payload. Thus, we can create a unidirectional or a bidirectional communication channel between two hosts, by sending a Diagnostic Payload with the smallest maximal size of 35B per message (PRBR=280). According to the pattern-based classification [14][24], this channel belongs to the following class:

```
Network Covert Storage Channels
 --Modification of Payload Pattern
```

Another similar channel can be created by encoding the data in some specific Internet media format (for example, “application/xml” media type) and sending this format as payload of a message with appropriate Content-Format option (41 for “application/xml”).

D. *Covert Channel Using Case-insensitive Parts of the URIs*

CoAP uses “coap” and “coaps” URI (Uniform Resource Identifier) schemes for identification of CoAP resources and providing a means for locating the resource. The URIs in the request are transported in several options: URI-host, URI-Path, URI-Port and URI-Query. They are used to specify the target resource of a request to CoAP origin server. The URI-host and the scheme are case insensitive, while all other components are case-sensitive. So, we can create a unidirectional covert channel between the client and the server using, for example:

- capital letter in the URI-host option to be binary 1, and
- small letter in the URI-host option to be binary 0.

Taking into account that valid Domain Name System (DNS) name has at most 255B, we can send at most 255B per message, or in other words, the PRBR of this channel is up to 255B. According to the pattern-based classification [14][24], this channel belongs to the following class:

```
Network Covert Storage Channels
 --Modification of Non-Payload
  --Structure Preserving
   --Modification of an Attribute
    --Value Modulation
     --Case Pattern
```

CoAP supports proxying, where proxy is a CoAP endpoint that can be tasked by CoAP clients to perform requests on their behalf. Proxies can be explicitly selected by clients, using the Proxi-URI option, and this role is “forward-proxy”. Proxies can also be inserted to stand in for origin servers, a role that is named as "reverse-proxy". So, we can create similar covert channel using schema and host part from the Proxi-URI option. A request containing the Proxy-URI Option must not include URI-host, URI-Path, URI-Port and URI-Query options.

E. *Covert Channel Using PUT and DELETE Methods*

The PUT method requires the resource identified by the URI in the request, to be updated or created with the enclosed representation. If the resource exists at the request URI, the enclosed representation should be considered as a modified version of that resource, and a 2.04 (Changed) Response Code should be returned. If no resource exists, then the server may create a new resource with the same URI that results in a 2.01 (Created) Response Code.

The DELETE method requires deletion of the resource, which is identified by the URI in the request. Regardless if the deletion is successful, or the resource did not exist before the request, a 2.02 (Deleted) Response Code should be send.

If somebody has a known representation of the existing resource R1 on the server and if he knows that specific resource R2 does not exist on the same server, a unidirectional covert channel to the server can be created, in this way:

- send request with PUT method to create the resource R1 with enclosed known representation as binary 1, and
- send request with DELETE method to delete non-existing resource R2 as binary 0.

In this way, one bit per message can be sent (PRBP=1). According to the pattern-based classification [14][24], this channel belongs to the following class:

```
Network Covert Storage Channels
 --Modification of Non-Payload
  --Structure Preserving
   --Modification of an Attribute
    --Value Modulation Pattern
```

F. Covert Channel Using Accept Option

The Accept option can be used to indicate which Content-Format is acceptable to the client. If no Accept option is given, the client does not express a preference. If the preferred Content-Format if available, the server returns in that format, otherwise, a 4.06 "Not Acceptable" must be sent as a response, unless another error code takes precedence for this response. We can create a unidirectional one-bit per message covert channel (PRBP=1), in this way:

- sending a given message without Accept option to be binary 1, and
- sending a given message with Accept option to be binary 0.

According to the pattern-based classification [14][24], this channel belongs to the following class:

```
Network Covert Storage Channels
 --Modification of Non-Payload
  --Structure Modifying
   --Add Redundancy Pattern
```

G. Covert Channel Using Conditional Requests

Conditional request options If-Match and If-None-Match enable a client to ask the server to perform the request only if certain conditions specified by the option are fulfilled. In the case of multiple If-Match options the client can make a conditional request on the current existence or value of an ETag for one or more representations of the target resource. This is useful to update the request of the resource, as a means for protecting against accidental overwrites when multiple clients are acting in parallel on the same resource. The condition is not fulfilled if none of the options match. With If-None-Match option the client can make a conditional request on the current nonexistence of a given resource. If the target resource does exist, then the condition is not fulfilled.

If somebody knows for sure that given condition C1 is fulfilled (for example, the resource is created or deleted in previous message) and other C2 is not fulfilled, using either of If-Match and If-None-Match options, a unidirectional one-bit per message covert channel (PRBP=1) can be created in this way:

- sending a given message without fulfilled condition to be binary 1 (e.g., If-Match + C2), and
- sending a given message with fulfilled condition (e.g., If-Match + C1) to be binary 0.

According to the pattern-based classification [14][24], this channel belongs to the following class:

```
Network Covert Storage Channels
 --Modification of Non-Payload
  --Structure Preserving
   --Modification of an Attribute
    --Value Modulation Pattern
```

H. Covert Channel Using Re-Transmissions

If we are using CoAP in channels with small error-rate (to cope with the unreliable nature of UDP), we can create a unidirectional or a bidirectional covert channel using retransmissions with PRBP=1, in this way:

- sending a given message only once to be binary 1, and
- sending a given message two or more times to be binary 0.

In this way, one bit per message can be sent. According to the pattern-based classification [14][24], this channel belongs to the following class:

```
Network Covert Timing Channels
--Protocol aware
  --Re-Transmission pattern
```

V. Performance Evaluation

Suppose that two IoT devices communicate with CoAP every *t* seconds.

Any covert channel with a given PRBR will need at least

$$\mathrm{ceil}(l\ /\ \mathrm{PRBR}) \cdot t\ (s)$$

for sending a message with length *l* bits.

We can evaluate the minimum time for sending the message ”Hello, world!” using the newly suggested covert channels. The message has length of 13 7-bit ASCII characters or l=91 bits. Results are given in Table I.

So, we can see that not all suggested covert channels in CoAP are able to send short messages in real time, especially the ones with PRBR=1. Still, the covert channels 3 and 4 can be used for sending a short message per one CoAP message, without rising any suspicions. If the time for sending the message is not so important, one can choose covert channels 1 or 2, without rising any suspicions.

TABLE I. PERFORMANCE EVALUATION OF THE NEW COVERT CHANNELS FOR SENDING THE MESSAGE "HELLO, WORLD!"

No.	Type of CC	PRBR	Time (*s*)		
			t=1*s*	*t*=5*s*	*t*=10*s*
1	CC using token and/or message ID Fields	16	6	30	60
		64	2	10	20
		80	2	10	20
2	CC using piggybacked and separate response	1	91	455	910
3	CC using payload of the message	280	1	1	1
4	CC using case-insensitive parts of the URIs	≤2040	1	1	1
5	CC using PUT and DELETE Methods	1	91	455	910
6	CC using Accept option	1	91	455	910
7	CC using conditional requests	1	91	455	910
8	CC using re-transmissions	1	91	455	910

TABLE II. PERFORMANCE EVALUATION OF THE NEW COVERT CHANNELS WITH PRBR>1 FOR SENDING 320X240 RAW COLOR IMAGE (WITH 24-BIT PIXELS)

	Type of CC	PRBR	Time(*s*)	
			t=1*s*	*t*=5*s*
1	CC using token and/or message ID Fields	16	115200 (32h)	576000 (160h)
		64	28800 (8h)	144000 (40h)
		80	23040 (6,4h)	115200 (32h)
2	CC using payload of the message	280	6583 (>1,82h)	32915 (>9.1h)
3	CC using case-insensitive parts of the URIs	≤2040	904 (15 min)	4520 (76 min)

Additionally, we can evaluate the minimum time for sending the 320x240 raw color image (with 24-bit pixels) using the newly suggested covert channels. The size of the image is 225KB or *l*=1843200 bits. Results are given in Table II.

The results from Table II show that most of the new CoAP covert channels are not quite suitable for sending images, because of the large transmission time. The covert channel 3 is the most suitable for that purpose (it will send 225KB image in 15 minutes).

VI. EXPERIMENTAL EVALUATION

For our research we have used Contiki OS, and specially, Instant Contiki version 3.0 as a development environment. It is a Ubuntu Linux virtual machine that runs in VMWare player. It has all the development tools, compilers and simulators. We can develop our application and test it on one of the devices in simulator. We used Cooja simulator. With it, we can create different types of devices for which we can develop applications. This is practical because before we execute our application on real device we will make sure it works properly.

For the purposes of our research we used Z1 Zolertia Mote. It is an ultra low power wireless module for use in wireless sensor networks (WSN). Z1 has the second generation of MSP430F2617 low power microcontroller, which has a powerful 16-bit RISC CPU @16MHz clock speed. It also has built-in clock factory calibration, 8KB RAM and a 92KB Flash memory. Z1 module includes the CC2420 transceiver, which operates at 2.4GHz with data rate of 250Kbps and it supports 802.15.4 standard to interoperate with other devices. This module has a built-in temperature and 3-axis accelerometer sensors. Z1 allows flexible powering using the battery pack (2xAA or 2xAAA), Coin Cell, USB and with direct connection.

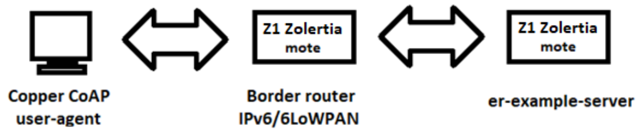


Figure 5. Implementation scenario

In our research, we used Copper (Cu) as a CoAP user-agent. It is a Firefox plugin that installs a handler for "coap" URI scheme and allows users to browse and interact with Internet of Things (IoT) devices. The scenario for our research is presented on Figure 5. We used the Cooja simulator to create a new simulation with, 2 Z1 Zolertia motes. One Z1 mote is for Border Router. As a source we used rpl-border router source code that is located in:

/home/user/contiki-3.0/examples/ipv6/rpl-border-router

The other Z1 mote is for CoAP server. In our research we used Erbium implementation of CoAP server for Contiki OS. The source code is located in:

/home/user/contiki-3.0/examples/er-rest-example

To adjust it to our needs, we made a change to the source code, specifically in the file “res-hello.c” that is located in the following path in Contiki:

/home/user/contiki-3.0/examples/er-rest-example/resources

According to this scenario, we have implemented the covert channel that uses the PUT and DELETE methods. By using Copper user-agent, we created request using the PUT and DELETE methods (with PUT in the 50th second of the execution of the simulation and with DELETE in the 60th second of the execution of the simulation). We also examined power consumption in case when we do not implement a covert channel and in the case of an implemented covert channel. To calculate the power consumption, we used the data obtained with the tool “Powertrace” for CoAP server with and without implemented covert channel. These data are printed at the mote output for Z1 module in Cooja. We made the calculations in a total time interval of 100 seconds as previously predefined interval for performing the simulation for both cases. These data show the total number of clock ticks in different states of the module: CPU (CPU in active mode), LPM (CPU in Low Power Mode), TX (Transmit) and RX (Receive) (Table III and Table IV).

TABLE III. DATA OBTAINED WITH “POWERTRACE” FOR “COAP” SERVER WITHOUT IMPLEMENTATION OF COVERT CHANNEL

ALL_CPU	ALL_LPM	ALL_TX	ALL_RX
4674	322863	149	294987
9879	645197	229	622586
15204	967576	412	950244
17500	1292676	412	1277763
19778	1617956	412	1605442
24933	1940346	514	1933022
27435	2265399	594	2260619
29721	2590672	594	2588298
32001	2915952	594	2915978
34271	3241241	594	3243658
39491	3563562	675	3571258

To calculate the power consumption, we used the following formula [27] :

$$Power\_consumptio = \frac{Energest\_value * Current * Voltage}{RTIMER\_SECOND * Runtime}$$

Energest_value is the difference between the number of clock ticks (in states CPU, LPM, TX and RX) between two time intervals. We used the Z1 datasheet to get the values for *Current* in different states (Approximate Current Consumption of Z1 circuits: Active Mode @16MHz - < 10 mA (approximate 9mA), Standby Mode - 0.5µA, RX Mode - 18.8mA, TX Mode - 17.4mA) [28]. The value for *Voltage* parameter is 3V. The value for *RTIMER_SECOND* is 32768. *Runtime* is the time interval (10 seconds in our case).

TABLE IV. DATA OBTAINED WITH “POWERTRACE” FOR “COAP” SERVER WITH IMPLEMENTED COVERT CHANNEL

ALL_CPU	ALL_LPM	ALL_TX	ALL_RX
4726	322829	149	294987
10020	645086	229	622586
15271	967393	332	950165
17738	1292480	413	1277762
20253	1617524	476	1605379
25731	1939607	641	1932896
28236	2264657	722	2260493
30521	2589930	722	2588173
32801	2915210	722	2915853
35071	3240499	722	3243533
40372	3562751	802	3571132

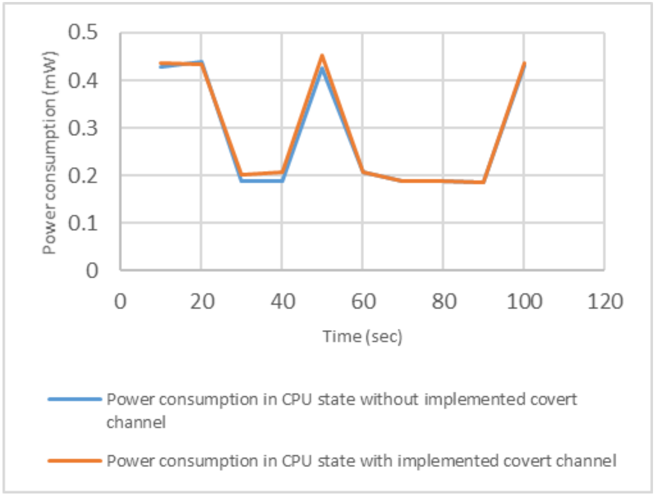


Figure 6. Power consumption for CoAP server (Z1) in CPU state (with and without implemented covert channel)

Figure 6 shows the power consumption for Z1 module (implemented as CoAP server) in CPU state with and without implemented covert channel. The average power consumption without implemented covert channel is 0.28688324 mW, while the average power consumption with implemented covert channel is 0.293713989 mW. We can see that the average power consumption with an implemented covert channel is bigger.

Figure 7 shows the power consumption for Z1 module (implemented as CoAP server) in LPM state with and without implemented covert channel. The average power consumption without implemented covert channel is 0.001483474 mW, while the average power consumption with implemented covert channel is 0.001483119 mW. We can see that the average power consumption with implemented covert channel is slightly smaller than the power consumption without implemented covert channel.

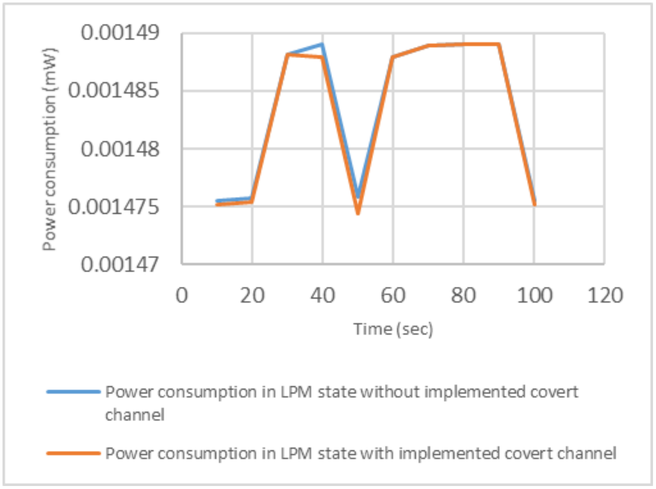


Figure 7. Power consumption for CoAP server (Z1) in LPM state (with and without implemented covert channel)

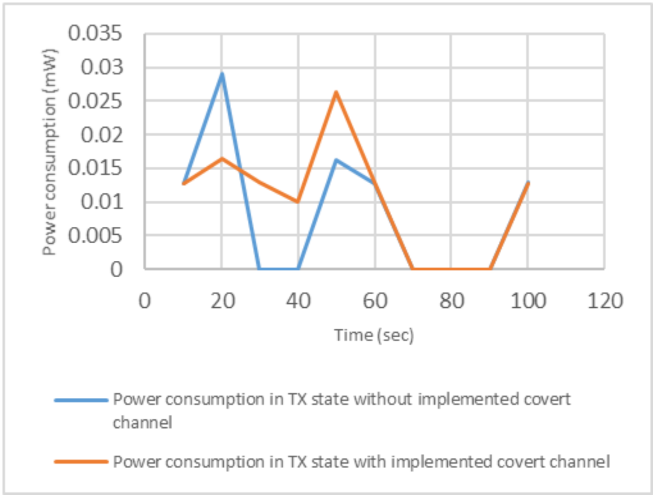


Figure 8. Power consumption for CoAP server (Z1) in TX state (with and without implemented covert channel)

Figure 8 shows the power consumption for Z1 module (implemented as CoAP server) in TX state with and without implemented covert channel. The average power consumption without implemented covert channel is 0.008379272 mW, while the average power consumption with implemented covert channel is 0.010402405 mW. We can see that the power consumption with implemented covert channel is around 1.24 times greater than the power consumption without implemented covert channel.

Figure 9 shows the power consumption for Z1 module (implemented as CoAP server) in RX state with and without implemented covert channel. The average power consumption without implemented covert channel is 56.3908949 mW, while the average power consumption with implemented covert channel is 56.3887262 mW.

We can see that the average power consumption with implemented covert channel is slightly smaller than the power consumption without implemented covert channel.

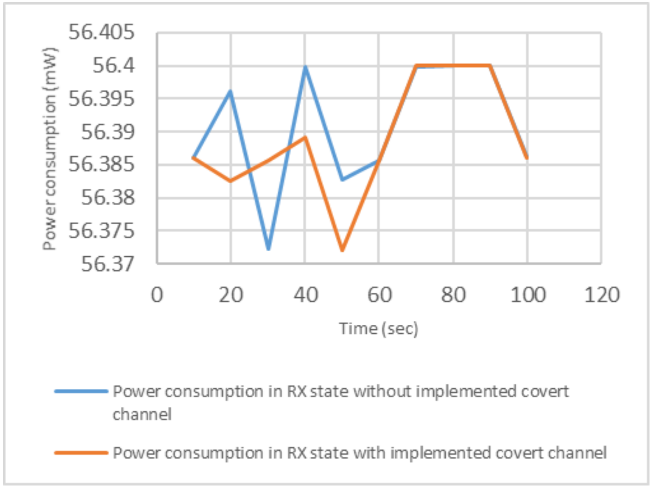


Figure 9. Power consumption for CoAP server (Z1) in RX state (with and without implemented covert channel)

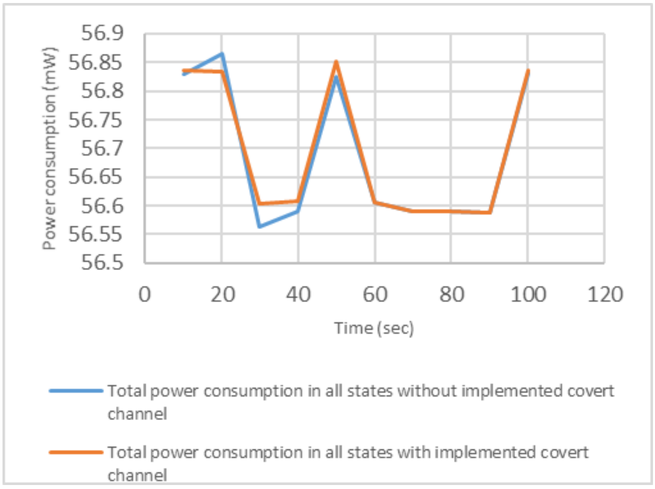


Figure 10. Total power consumption for CoAP server (Z1) in all states (with and without implemented covert channel)

Figure 10 shows the total power consumption for Z1 module (implemented as CoAP server) in all states, in each time interval with and without implemented covert channel. The average power consumption without implemented covert channel is 56.68764088mW, while the average power consumption with implemented covert channel is 56.69432571mW. We can see that the average power consumption (in all states) for Z1 module with implemented covert channel (when sending two bits) is slightly greater than the power consumption without implemented covert channel.

The power consumption of the Z1 module in the 50th second (the time when we sent a request with the PUT method) with implemented covert channel has increased very little, for only 0.02580548 mW.

We have the same case in the 60th second (the time when we sent a request with the DELETE method), when the power consumption with implemented covert channel has increased very little, for only 0.00040648 mW. The implementation of the covert channel using the PUT and DELETE methods does not greatly affect the power consumption of the Z1 module.

VII. CONCLUSION

Considering that IoT will consist of about 30 billion objects by 2020 [17], CoAP belongs to the group of possible most exploited protocols in the forthcoming years. The CoAP covert channels presented here, are suitable for sending short messages, as our performance evaluation showed. Additionally, the performed experimental evaluation of power consumption analysis on one of the covert channels, shows only a slight increase in the power consumption of the used device, when sending two bits. The consequence of all these results, is the importance of identifying as much as it can, the possible ways of hiding data in CoAP and trying to mitigate them. One can deploy active and passive wardens for this purpose, but this is left for later investigation.

REFERENCES

[1] A. Mileva, A. Velinov, and D.Stojanov, “New Covert Channels in Internet of Things,” Proc. 12th International Conference on Emerging Security Information, Systems and Technologies - *SECURWARE 2018*, Venice, Italy, 2018, pp. 30-36.

[2] Constrained RESTful Environments (CoRE) Parameters, CoAP Codes [Online]. Available at: https://www.iana.org/assignments/core-parameters/core-parameters.xhtml [retrieved: July, 2018]

[3] K. Denney, A. S. Uluagac, K. Akkaya, and S. Bhansali, “A novel storage covert channel on wearable devices using status bar notifications,” Proc. 13th IEEE Annual Consumer Communications & Networking Conference, *CCNC 2016*, Las Vegas, NV, USA, 2016, pp. 845-848, doi: 10.1109/CCNC.2016.7444898.

[4] N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, and D. Karger, “Infranet: Circumventing Web Censorship and Surveillance,” Proc. 11th USENIX Security Symposium, San Francisco, CA, 2002, pp. 247-262.

[5] S. Fincher et al., “Perspectives on HCI patterns: concepts and tools,” Proc. Extended Abstracts on Human Factors in Computing Systems (CHI EA ’03). ACM, New York, NY, USA, 2003, pp. 1044–1045, doi: 10.1145/765891.766140.

[6] D. V. Forte, “SecSyslog: An Approach to Secure Logging Based on Covert Channels,” Proc. First International Workshop of Systematic Approaches to Digital Forensic Engineering (SADFE 2005), Taipei, Taiwan, 2005, pp. 248-263, doi: 10.1109/SADFE.2005.21.

[7] M. Guri, G. Kedma, A. Kachlon, and Y. Elovici, “AirHopper: Bridging the Air-Gap between Isolated Networks and Mobile Phones using Radio Frequencies,” MALWARE 2014, 2014.

[8] A. Houmansadr, N. Kiyavash, and N. Borisov., “RAINBOW: A Robust And Invisible Non-Blind Watermark forNetwork Flows,” Proc. 16th Network and Distributed System Security Symposium (NDSS 2009), San Diego, USA, The Internet Society, 2009.

[9] M. N. Islam, V. C. Patil, and S. Kundu, “Determining proximal geolocation of IoT edge devices via covert channel,” Proc. 18th International Symposium on Quality Electronic Design, ISQED 2017, Santa Clara, CA, USA, 2017, pp. 196-202, doi: 10.1109/ISQED.2017.7918316.

[10] B. W. Lampson, “Note on the Confinement Problem,” Commun. ACM vol. 16, 10, Oct. 1973, pp. 613-615, doi: 10.1145/362375.362389.

[11] D. Martins and H. Guyennet, “Attacks with Steganography in PHY and MAC Layers of 802.15.4 Protocol,” Proc. Fifth International Conference on Systems and Networks Communications (ICSCN), Nice, France, 2010, pp. 31-36, doi: 10.1109/ICSNC.2010.11.

[12] W. Mazurczyk and Z. Kotulski, “New Security and Control Protocol for VoIP Based on Steganography and Digital Watermarking,” Annales UMCS Informatica AI 5, 2006, pp. 417-426, doi: 10.17951/ai.2006.5.1.417-426.

[13] W. Mazurczyk and K. Szczypiorski, “Steganography of VoIP Streams,” in On the Move to Meaningful Internet Systems (OTM 2008) Robert Meersman, Zahir Tari (Eds.). LNCS, vol. 5332, 2008, pp. 1001-1018, doi: 10.1007/978-3-540-88873-4_6.

[14] W. Mazurczyk, S. Wendzel, Z. Zander, A. Houmansadr, and K. Szczypiorski, “Information Hiding in Communication Networks” Wiley / IEEE Comp. Soc. Press, (2016).

[15] A. Mileva and B. Panajotov, “Covert channels in TCP/IP protocol stack - extended version-,“ Central European Journal of Computer Science vol. 4, 2, 2014, pp. 45-66, doi: 10.2478/s13537-014-0205-6.

[16] A. K. Nain and P. Rajalakshmi, “A Reliable Covert Channel over IEEE 802.15.4 using Steganography,” Proc. IEEE 3rd World Forum on Internet of Things (WF-IoT), Reston, VA, USA, 2016, pp. 711-716, doi: 10.1109/WF-IoT.2016.7845486.

[17] A. Nordrum, “Popular Internet of Things Forecast of 50 Billion Devices by 2020 Is Outdated,” IEEE Spectrum. 18 August, 2016.

[18] R. Patuck and J. Hernandez-Castro, “Steganography using the Extensible Messaging and Presence Protocol (XMPP),” Computing Research Repository arXiv:1310.0524, 2013.

[19] Z. Shelby, K. Hartke, and C. Bormann, “The Constrained Application Protocol (CoAP),” RFC 7252, 2014.

[20] N. Tuptuk and S. Hailes, “Covert channel attacks in pervasive

computing," IEEE PerCom, pp. 236–242, 2015.

[21] X. Wang and D. S. Reeves, "Robust correlation of encrypted attack traffic through stepping stones by manipulation of inter packet delays," Proc. 10th ACM Conference on Computer and Communications Security (CCS'03), 2003, pp. 20-29, doi: 10.1145/948109.948115.

[22] X. Wang, S. Chen, and S. Jajodia, "Tracking anonymous peer-to-peer VoIP calls on the Internet," Proc. 12th ACM Conference on Computer and Communications Security (CCS'05), Alexandria, VA, USA, 2005, pp. 81-91, doi: 10.1145/1102120.1102133.

[23] S. Wendzel, " Covertand Side Channels in Buildings and the Prototype of a Building-aware Active Warden" IEEE ICC, pp. 6753-6758, 2012.

[24] S. Wendzel, S. Zander, B. Fechner, and C. Herdin, "Pattern-Based Survey and Categorization of Network Covert Channel Techniques," ACM Computing Surveys vol. 47, 3, Article 50, 2015, doi: 10.1145/2684195.

[25] S. Wendzel, W. Mazurczyk, and G. Haas, "Don't You Touch My Nuts: Information Hiding in Cyber-physical Systems," IEEE SPW 2017, pp. 29-34, 2017.

[26] S. Zander, G. Armitage, and P. Branch, "A survey of covert channels and countermeasures in computer network protocols," IEEE Communications Surveys and Tutorials vol. 9, 3, 2007, pp. 44-57, 10.1109/COMST.2007.4317620.

[27] Internet of Things technology [Online]. Available at: http://thingschat.blogspot.com/2015/04 /contiki-os-using-powertrace-and.html [retrieved: January, 2019]

[28] Z1 Datasheet [Online]. Available at: http://zolertia.sourceforge.net/wiki/images/e/e8/ Z1_RevC_Datasheet.pdf [retrieved: January, 2019]

Secure Cooperation of Untrusted Components Using a Strongly Typed Virtual Machine

Roland Wismüller and Damian Ludwig
University of Siegen, Germany
E-Mail: {roland.wismueller, damian.ludwig}@uni-siegen.de

Abstract—**A growing number of computing systems, e.g., smart phones or web applications, allow to compose their software of components from untrusted sources. For security reasons, such a system should grant a component just the permissions it really requires, which implies that permissions must be sufficiently fine-grained. This leads to two questions: How to know and to specify the required permissions, and how to enforce access control in a flexible and efficient way? We present the design and implementation of a novel approach based on the object-capability paradigm with access control at the level of individual methods, which exploits two fundamental ideas: we simply use a component's published interface as a specification of its required permissions, and extend interfaces with optional methods, allowing to specify permissions that are not strictly necessary, but desired for a better service level. These ideas have been realized within a static type system, where interfaces specify both the availability of methods, as well as the permission to use them. In addition, we support deep attenuation of rights with automatic creation of membranes, where necessary. Thus, our access control mechanisms are easy to use and also efficient, since in most cases permissions can be checked when the component is deployed, rather than at run-time. Based on our type system, we have defined a secure intermediate representation, specified its semantics and sketched a correctness proof. The presented concepts have been implemented in a virtual machine called COSMA. When a component is loaded, COSMA type checks its intermediate representation and then compiles it into native machine code, thus enabling its execution with minimal run-time overhead. Thus, COSMA enables the secure, efficient, and flexible cooperation of untrusted software components.**

Keywords–Security; software component; type system; object-capability model; membrane; virtual machine.

I. Introduction

In today's computer based systems, the software environment is often composed of components developed by an open community. Prominent examples are web applications, and smart phones with their app stores. A major problem in such systems is the fact that the components' origin and therefore the components themselves may not be trusted [1] [2]. In order to ensure security in systems composed of untrusted components, the *Principle Of Least Authority* (POLA) should be obeyed, i.e., each component should receive just the permissions it needs to fulfill its intended purpose [3]. The term 'authority' denotes the effects, which a subject can cause. These effects can be restricted via permissions, which control the subject's ability to perform actions. An appealing and popular approach to implement POLA is the use of the object-capability model [4] [5], where unforgeable object references are used as a capability allowing to use the referenced object.

Based on the object-capability paradigm, several secure languages have been devised, such as the E language [4], Joe-E [6] or Emily [7]. In order to allow a fine grained access control at the level of individual methods, the programmer has to manually implement security-enforcing abstractions, e.g., membranes [5]. An inherent problem of language based approaches is that security is achieved mainly by restrictions of the source language and associated compile-time checks. Thus, they not only confine interactions between different components, but also limit the programmer's capabilities within a single component. A second drawback is that security can only be guaranteed, if all components are available in the form of source code, which in practice is infeasible for reasons of protecting intellectual property rights.

The second problem can be addressed by enforcing security at the level of a Virtual Machine (VM). However, existing VMs like Oviedo3 [8] only provide basic mechanisms for the management of access rights, i.e., adding and removing the permission to execute a method for a given object reference, and must check all these permissions at run-time. Thus, they are neither easy to use nor efficient.

To overcome the drawbacks of existing approaches, the goal of our work is to provide a VM that eliminates the shortcomings of existing capability systems and secure high-level languages, and addresses the special needs for the secure cooperation of untrusted components. In particular, it

- allows components to be distributed and deployed in binary form while still providing security,
- enables fine-grained access control without putting a relevant annotation or implementation burden on the components' programmers,
- does not restrict the code's expressiveness within a single component, and
- minimizes the number of required run-time checks by performing most checks when a component is deployed.

Our paper is organized as follows: We start with a discussion of related work in Section II, before in Section III, we present our security model and a component model, where each component specifies its minimal and desired permissions in a natural way using interfaces. We then outline the details of a type system that allows fine-grained access restrictions and optional methods (Section IV). In addition, we introduce the concepts and the implementation of a virtual machine based on a secure, strongly-typed byte code, which allows static type checking at deployment time and the automatic creation of membranes (Section V). We conclude the paper by giving an outlook to our future work (Section VI). The appendix contains the definition of the formal semantics of the virtual machine's instruction set and sketches a proof of its primary security property. This paper is an extended version of [1], which includes an elaborate discussion of our type system, the automatic creation of membranes and the implementation of our virtual machine.

II. Related Work

Component-based software development has a long tradition in software engineering. Large software systems are built by composing smaller parts, which interact only via well-defined interfaces. Recent research on composition aspects focuses on how to specify the semantics of these interfaces, such that properties of the system can be derived from the properties of the individual components [9] [10] [11]. While typically components are assumed to be trustworthy, secure cooperation of untrusted components is gaining more and more attention, e.g., in the area of telecommunication systems [12] or web application [2]. In many cases, the security is based on the object-capability model.

Capability based protection mechanisms date back as early as the 1960-ies with, e.g., the IBM System/38 and the Hydra operating system as prominent examples [13]. A good introduction to the object-capability model and the principle of least authority is provided in [14]. The general properties of capability systems, as well as some common misconceptions about capabilities are clearly pointed out in [15]. The authors show that capabilities have strong advantages over access control lists and can support both confinement and revocation of access rights. More details about the object-capability paradigm as well as revocation and confinement are discussed in [3]. A detailed discussion of several common object-capability patterns, including membranes, together with a formal modeling and proofs has been presented by Murray [5].

The object-capability paradigm has been used as a basis for secure languages. A pioneer in this area is the work of Mark Samuel Miller [4] on the E language, which points out the prerequisites for secure languages: memory safety, object encapsulation, no ambient authority, no static mutable state, and an API without security leaks. In addition to these features, E provides capabilities for access control at the object level. Access control at the granularity of methods is also possible, but requires the programmer to explicitly implement security-enforcing abstractions like membranes. Based on E, Joe-E [6] restricts Java such that access to objects is only possible via capabilities that have been explicitly passed to a component. It also supports immutable interfaces allowing to implement secure plug-ins. Joe-E uses compile-time checking and secure libraries to disable insecure features of Java like, e.g., reflection and ambient authority. In a similar spirit, Emily [7] is a secure subset of OCaml. Emily does not have the notion of components, but dynamically grants permissions to launched applications via explicit use of the powerbox pattern. Maffeis et al. [2] specifically address the problem of mutual isolation of (third-party) web applications written in JavaScript. They define a language to be *authority safe*, if it satisfies the properties "only connectivity begets connectivity" and "no authority amplification", and formally show that authority safety implies isolation.

An implementation of capability-based access control at method granularity within a virtual machine is presented in [8] [16] [17]. Oviedo3 consists of an object oriented abstract machine and an accompanying operating system. It provides basic mechanisms for the management of access rights, i.e., adding and removing the permission to execute a method for a given object reference, and checks all these permissions at run-time.

A different area of research examines the use of type systems for enforcing security constraints. One goal that has been achieved is alias control, i.e., a means to restrict the sharing of references. Several different concepts have been developed, e.g., universes [18], ownership types [19] [20] [21] or confined types [22]. In [23], Philip Fong shows that these concepts can be used at the byte code level, enabling to enforce confined types at link time, while in [24] [25] the same author discusses how to formally model capabilities, such that confinement can be guaranteed. The common idea of these approaches is to augment class types, such that references to instances of these classes created in some context x cannot be passed to another context y.

There are also several proposals to assign more powerful and flexible security restrictions to types. As an example, both [26] and [27] allow the type system to enforce restrictions on information flow, similar to the Bell-LaPadula model. A powerful, but also complex capability type system is introduced in [28], which allows the programmer to define sequences of aliasing events that may occur to a reference type.

In [29], the authors present the idea of adding hidden capabilities to the interface description of components in order to separate protection definition from application code. Capabilities are implemented in the classical way using OS protection mechanisms. An approach to define the semantics of common type annotations used to specify access rights for, e.g., reading and writing objects is outlined in [30]. While the code can be checked statically in this approach, it does not support fine-grained access permissions for methods. Strategies for fine-grained access control with link-time enforcement have been developed in IsoMod [31]. Based on the idea that if a loaded module does not know the name of an entry point, it cannot call the corresponding service, IsoMod defines a powerful, but also rather complex policy definition language. A problem, which is inherited from the underlying Java VM is that there are no explicit interfaces between the modules, making it extremely hard to determine the minimal rights required by a module. In addition, only nominal typing is supported and rights cannot be attenuated at run-time. The Safe Language Kernel [32] employs a similar basic idea by introducing type capabilities, which are checked at link time. However, since the paper was never officially published the concept is not fully worked out and has not been integrated into a type system.

The above discussion shows that although there are some partial solutions for supporting the secure cooperation of untrusted components, there is no satisfying practical approach for this problem yet. The language-based approaches achieve security by imposing restrictions on the programming language, which also affects its expressiveness for programming within a single component. In addition, they require components to be distributed as source code, which is not desirable. Approaches based on the object-capability paradigm or on type systems for alias control in general just support access control at the object-level, but not at the method-level. Fine-grained access control either requires manually programmed security abstractions or a run-time management and enforcement of method permissions implying a significant overhead. The more sophisticated type based approaches, like [28] result in a significant annotation burden for the programmer, which hinders their widespread usage.

The contribution of our work is to provide an approach that avoids the mentioned shortcomings by combining existing techniques with completely new ideas. Our solution is based on a VM executing a strongly typed intermediate language where types are used to represent permissions. A central idea is to specify fine-grained access control by just using standard interface definitions, thus avoiding complex code annotations. Since the VM performs a static type check, which in our approach also corresponds to a permission check, when a new component is loaded, most run-time checks can be avoided. If necessary, the VM automatically builds the required membranes for method-level access restrictions. The VM's underlying type system has been designed in such a way that it does not limit expressiveness within a component, but just restricts the permissions of object references passed from one component to another. While we did not implement annotations for alias control yet, this may easily be added in future.

III. Component and Security Model

A. Component Model

Our work is based on the established definition of a software component, as given by Szyperski: "*A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties*" [33]. We assume that components are distributed as compiled byte code for a VM, rather than source code. Their internal structure is not relevant, however, we require that a component defines a purely object oriented interface, i.e., to its environment it appears to be composed of classes. One of these classes, the *principal class*, is the component's entry point, i.e., when the component is loaded, an instance of the principal class is created and its constructor is invoked.

B. Security Model

In order to support qualified statements about security properties, we structure the computing system under consideration into a disjoint set of security domains that we call *contexts*. We assume that the system features a trusted computing base, which just consists of the VM and a minimal operating system (OS) kernel, and is referred to as OS context. Any other service available in the system is implemented by loading components from an external, untrusted source. When the OS loads a component, it creates a new context c and instantiates the component's principal class within this context. In turn, when a method of any object in context c instantiates a new object, the new object will also be enclosed in c. Thus, c comprises all objects created on behalf of the loaded component.

Our threat model now is as follows: Code executing within a context c may try to compromise the confidentiality, integrity, authenticity and availability of any data contained in or service provided by a context c' by accessing or invoking data or code in other contexts, including the OS context.

As a prerequisite, we assume that all interactions between different contexts are based on the object-capability paradigm, i.e., access to data or code is possible only via references that can be passed between contexts and act as capabilities. In addition, data can be accessed only by calling an object's methods. This implies that there is no ambient authority, thus, the OS kernel must also exhibit a purely object oriented interface.

The goal of our work now is to parry the depicted threat by enforcing the following property: Code of a component that has been loaded into a context c can only perform actions (i.e., execute methods on objects) in contexts different from c, which (1) have been explicitly documented by that component, and (2) have been explicitly permitted to context c.

Compared to the traditional object-capability model, there are two significant extensions. First, the granularity of access control is more fine grained, since our model controls the ability to execute certain actions on an object, rather than just the overall access to the object. Second, components include an explicit definition of their required permissions.

C. An Example

Consider a component `Calendar` that manages and provides appointments. In a Java-like language (where access modifiers have been omitted for brevity), this component may be defined as shown in Listing 1.

Listing 1. Calendar interface

```
principal class Calendar {
  interface Appointment {
    int startTime();
    int endTime();
    String subject();
    String notes();  // confidential notes
  }
  void createAppointment(...) { ... }
  // return next upcoming appointment
  Appointment getNextAppointment() { ... }
  ...
}
```

Now assume that there is a component C whose task is just to display the next upcoming appointment in some widget. In this situation, POLA requires that C can just get the next appointment, but not, e.g., create a new one. However, the standard object-capability model will allow C to invoke `createAppointment()` once it receives a reference to a `Calendar` object. This is true, even when the reference is passed using a restricted type (i.e., an interface type just containing `getNextAppointment()`), since virtually all popular object oriented languages will allow C to downcast the reference to the type `Calendar` again.

D. Fine Grained Access Control

In principle, an object-capability system can be extended with a more fine grained access control by supplementing

Let T_{pc}	denote the type of the principal class of C,
$\mathcal{T}_{\text{arg}}(T)$	the set of the argument types of all methods defined in a type T,
$\mathcal{T}_{\text{res}}(T)$	the set of the result types of all methods defined in type T,
$\mathcal{T}_{\text{in}}$	the set of all types of references (or values) that component C receives from its environment (required interface), and
$\mathcal{T}_{\text{out}}$	the set of all types of references (or values) passed from component C to its environment (provided interface).

Then $(\mathcal{T}_{\text{in}}, \mathcal{T}_{\text{out}})$ is the least fixed point of the following equations:

$$\begin{aligned}\mathcal{T}_{\text{in}} &= \mathcal{T}_{\text{arg}}(T_{\text{pc}}) \cup \mathcal{T}_{\text{arg}}(\mathcal{T}_{\text{out}}) \cup \mathcal{T}_{\text{res}}(\mathcal{T}_{\text{in}}) \\ \mathcal{T}_{\text{out}} &= \{T_{\text{pc}}\} \cup \mathcal{T}_{\text{res}}(\mathcal{T}_{\text{out}}) \cup \mathcal{T}_{\text{arg}}(\mathcal{T}_{\text{in}})\end{aligned}$$

Figure 1. Computation of the provided and required interfaces of a component C.

each reference with a list of permissions, which allow or disallow the available methods. However, this approach, which is, e.g., implemented in Oviedo3 [8] [17] imposes a significant overhead both for storing the access rights in each reference and for checking them at run-time whenever the reference is used. In addition, it does not directly allow to restrict the permissions for references returned by calls to a permitted method. In our example, permissions included in the `Calendar` reference passed to C can prevent C from calling `createAppointment()`, but not from calling `notes()` on the appointment returned by `getNextAppointment()`.

A better solution is to use the membrane pattern [4] [5]: Instead of providing C with a reference to the real calendar, we create a membrane object, which acts similar to a proxy in the sense that it delegates method calls to the calendar object. The important difference is that the membrane will only provide a `getNextAppointment()` method. In addition, it can also wrap the returned appointment into another membrane that does not provide the `notes()` method. Thus, membranes can also support deep attenuation of rights.

However, the classical membrane pattern has two severe drawbacks: First, the manual creation of membranes to enforce a minimal set of permissions is a difficult and error prone task for large object systems, since the necessary membrane structure may be deeply nested or may even be recursive. Second, if permissions are attenuated in several steps (e.g., component A passes an attenuated version of the calendar to B, which passes a more attenuated version to C), membranes will be cascaded, thus sacrificing run-time performance due to multiple delegation. As we will show, both drawbacks can be avoided by automatically generating membranes, when necessary.

E. Specifying Required Permissions

Another general problem related to POLA is that the user of a component C must know the minimal permissions required by C in order to work properly. One of the central ideas of our work is to use the already available type definition of an object reference as a specification of the permissions that are requested (in the case of an input variable) or granted (in the case of an output variable) by this reference. This perception of type definitions is possible when the run-time system executing the code of a component does not allow any of its input references to be downcasted to a less restrictive type.

Now, given our purely object oriented component model, we can exactly determine the minimal permissions that a component C requires from its environment by determining the types of all references that C can receive. Vice versa, we also can identify the permissions that C grants to its environment from the types of all references that C returns. In order to increase the model's flexibility, we also allow optional methods (i.e., permissions) in interfaces. In this way, the type of a component C's principal class explicitly defines

- $\mathcal{T}_{\text{in}}$: the minimum and maximum permissions that C requests from its environment, where C will use optional methods, if they are available, but does not require them for its correct operation, and
- $\mathcal{T}_{\text{out}}$: the minimum and maximum permissions that C grants to its environment, where for each optional method, C may decide at run-time whether or not to provide it.

The set $\mathcal{T}_{\text{in}}$ ($\mathcal{T}_{\text{out}}$) is determined by recursively computing the types of all references that the component can receive from (pass to) its environment, as shown in Figure 1.

As an example, consider the calendar component in Listing 1. As the component has no input (we omitted the parameters of `createAppointment()` for simplicity), `Calendar` does not request any permissions from its environment, so $\mathcal{T}_{\text{in}} = \emptyset$. However, it grants permission to use the appointment returned by `getNextAppointment()` via the `Appointment` interface, which results in $\mathcal{T}_{\text{out}} = \{\texttt{Calendar}, \texttt{Appointment}, \texttt{int}, \texttt{String}\}$.

The calendar client displaying upcoming appointments may now have an interface similar to Listing 2.

This interface specifies the permissions the client needs from a `Provider`: it must be able to call the `getNextAppointment()` method, which returns an object of type `Event`. On an `Event`, the client must be able to call `startTime()` and `endTime()`, and it will use `subject()`, if available. Thus, for the calendar client component we have $\mathcal{T}_{\text{out}} = \{\texttt{CalendarClient}\}$ and $\mathcal{T}_{\text{in}} = \{\texttt{Provider}, \texttt{Event}\}$. Since we use structural typing for component interfaces, a reference to the `Calendar` component can be passed to `setProvider()`, as `Appointment` provides all the methods required by `Event`.

LISTING 2. CALENDAR CLIENT INTERFACE

```
principal class CalendarClient {
  interface Provider {
    Event getNextAppointment();
  }
  interface Event {
    int startTime();
    int endTime();
    optional String subject();
  }
  void displayEvents();
  void setProvider(Provider c);
}
```

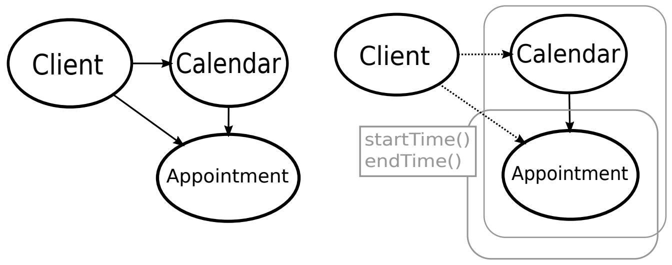


Figure 2. Full access to `Appointment` (left) versus restricted permissions (right): the client has access to `Calendar` only through a membrane. The membrane's `getNextAppointment()` method in turn returns a membrane for the `Appointment` object, which only allows two methods to be called.

In this example, the calendar client will not be able to call the `notes()` method on events received from any `Provider`, because it is not part of the `Event` interface. Formally, a component C can invoke method m on an object o from another component, only if o can be assigned to a reference of some type $T \in \mathcal{T}_{\text{in}}$, which allows to call m. Especially, a component can only execute the operations explicitly specified in its published interface. This means that everything the component can do is explicitly visible in its published interface, so the user can decide not to install the component or to only provide it with a reference to a restricted object where (some of) the optional permissions are not granted. Traditionally, this requires to manually program a membrane for the `Calendar` component, such that the object returned by `getNextAppointment()` does not have a `subject()` method (see Figure 2). In our model, the same effect can be achieved by just casting the `Calendar` reference to a more restricted interface, where the `subject()` method is missing.

In principle, if the `Calendar` component declared the `subject()` method in `Appointment` as optional, it also could decide at run-time whether or not to expose this method to the client invoking `getNextAppointment()`, e.g., based on some authentication procedure. However, we believe that this decision should usually be left to the user assembling the components.

Note that a component's published interface (what it pretends to do) may differ from its actually implemented interface, e.g., a component may try to call a method not declared in its published interface. However, because the component will always be *used* via its published interface, such a deviation

TABLE I. INTENDED SEMANTICS OF A TYPE T.

Status of method m in type T	Assertion that the referenced object has method m	Permission to call method m
m is not in T	no	no
m is optional in T	no	yes
m is required in T	yes	yes

will result in a type error when the component is loaded. We will present a detailed discussion of our type system in the next section.

IV. TYPE SYSTEM

In order to ease the presentation of our type system, we make a few simplifying assumptions for the following discussion: First, we assume that components are written in a statically typed, object oriented language. This will usually be the case today and is also assumed by our current implementation. However, the only real requirement of our approach is just that the interface between a component and its environment is purely object oriented. Second, the presentation only covers object types, i.e., class and interface types, although our implementation of the type system also provides simple types and array types. Finally, we will assume the use of structural typing for interface types. An extension allowing a flexible mixture of structural and nominal typing is currently being developed (see Section VI).

A. Types as Permissions

A central idea of our type system is to interpret a component's type as a specification of access permissions at the granularity of single methods. In addition, we retain the traditional interpretation, which asserts that all objects of a given type will offer the methods specified by that type. We achieve both goals by using the concept of optional methods, as specified in Table I.

As the main goal of our type system is security, it must enforce the access restrictions defined by Table I in such a way that no component can amplify its rights by type conversions, i.e., downcasting. Whenever possible, we ensure this property statically, i.e., at the time a component is deployed, rather than by using run-time checks. In addition, we avoid delayed type failures: once a component C is deployed and a reference to C's primary object has successfully been assigned to a variable of some type I, all methods in I can be invoked without run-time type errors. Finally, the type system supports an easy attenuation of rights by just upcasting a reference, without the need to manually code a membrane.

B. Subtyping Rules

From Table I, we can immediately derive the subtype rules required for our type system: if we have two types S and T, which only differ in the status of a method m, then S is subtype of T, if and only if

- S contains m, but T does not (classical situation),

$$\textsc{S-Refl}\ \frac{}{S <: S} \qquad \textsc{S-Trans}\ \frac{S <: U \qquad U <: T}{S <: T}$$

$$\textsc{S-Rcd}\ \frac{\{l_i\ ^{i \in 1..n}\} \subseteq \{k_j\ ^{j \in 1..m}\} \quad k_j = l_i\ \Rightarrow\ (n_j \Rightarrow o_i) \wedge (S_j <: T_i)}{\{(k_j : S_j,\, n_j)\ ^{j \in 1..m}\} <: \{(l_i : T_i,\, o_i)\ ^{i \in 1..n}\}}$$

$$\textsc{S-Arrow}\ \frac{T_1 <: S_1 \qquad S_2 <: T_2}{S_1 \to S_2 <: T_1 \to T_2}$$

Figure 3. Subtyping rules

$$\textsc{CC-Refl}\ \frac{}{S \prec:_c S} \qquad \textsc{CC-Trans}\ \frac{S \prec:_c U \qquad U \prec:_c T}{S \prec:_c T}$$

$$\textsc{CC-Rcd}\ \frac{\{l_i\ ^{i \in 1..n} \mid \overline{o_i}\} \subseteq \{k_j\ ^{j \in 1..m}\} \quad k_j = l_i\ \Rightarrow\ (n_j \Rightarrow o_i) \wedge (S_j \prec:_c T_i)}{\{(k_j : S_j,\, n_j)\ ^{j \in 1..m}\} \prec:_c \{(l_i : T_i,\, o_i)\ ^{i \in 1..n}\}}$$

$$\textsc{CC-Arrow}\ \frac{T_1 \prec:_c S_1 \qquad S_2 \prec:_c T_2}{S_1 \to S_2 \prec:_c T_1 \to T_2}$$

Figure 4. Defining rules for $\prec:_c$. If $S \prec:_c T$, then $r : S$ can be assigned to $r' : T$ without the need for a type check at run-time.

$$\textsc{CM-Refl}\ \frac{}{S \prec:_m S} \qquad \textsc{CM-Trans}\ \frac{S \prec:_m U \qquad U \prec:_m T}{S \prec:_m T}$$

$$\textsc{CM-Rcd}\ \frac{\{l_i\ ^{i \in 1..n}\} \subseteq \{k_j\ ^{j \in 1..m}\} \quad k_j = l_i\ \Rightarrow\ S_j \prec:_m T_i}{\{(k_j : S_j,\, n_j)\ ^{j \in 1..m}\} \prec:_m \{(l_i : T_i,\, o_i)\ ^{i \in 1..n}\}}$$

$$\textsc{CM-Arrow}\ \frac{T_1 <: S_1 \qquad S_2 <: T_2}{S_1 \to S_2 \prec:_m T_1 \to T_2}$$

Figure 5. Defining rules for $\prec:_m$. If $S \prec:_m T$, then $r : S$ can be assigned to $r' : T$ without the need to introduce a membrane.

$$\textsc{C-Refl}\ \frac{}{S \prec: S} \qquad \textsc{C-Trans}\ \frac{S \prec: U \qquad U \prec: T}{S \prec: T}$$

$$\textsc{C-Rcd}\ \frac{\{l_i\ ^{i \in 1..n} \mid \overline{o_i}\} \subseteq \{k_j\ ^{j \in 1..m}\} \quad k_j = l_i\ \Rightarrow\ S_j \prec: T_i}{\{(k_j : S_j,\, n_j)\ ^{j \in 1..m}\} \prec: \{(l_i : T_i,\, o_i)\ ^{i \in 1..n}\}}$$

$$\textsc{C-Arrow}\ \frac{T_1 \prec:_c S_1 \qquad S_2 \prec:_c T_2}{S_1 \to S_2 \prec: T_1 \to T_2}$$

Figure 6. Defining rules for $\prec:$. If $S \prec: T$, then $r : S$ can be assigned to $r' : T$.

- or m is required in S, but optional in T (since this means that every object implementing S also implements T).

The formal subtyping rules are shown in Figure 3. We model classes and interfaces as record types, whose members are functions. Functions have just one argument and return type, however, multiple argument and/or result values are possible, since the type can again be a record. The notation $\{(l_i : T_i,\, o_i)\ ^{i \in 1..n}\}$ denotes a type with n methods named l_i modeled as functions of type $T_i = T_i' \to T_i''$, where the Boolean value o_i indicates whether or not the method is optional. Compared to traditional type systems (see, e.g., [34]), the rule S-Rcd representing structural subtyping is modified with an extension for optional methods: assume that $S = \{(k_j : S_j,\, n_j)\ ^{j \in 1..m}\}$ and $T = \{(l_i : T_i,\, o_i)\ ^{i \in 1..n}\}$, then for $S <: T$ we additionally require that for all common members m of S and T either m is not optional in S, or it is also optional in T.

C. Well-Typed Programs

For safety and security reasons, we may allow the VM to load a component only if the component's code is *well-typed*. According to Cardelli [35], this means that the code will not exhibit any unchecked run-time errors (although controlled exceptions are allowed). The main question in this context is: when can we allow to assign a reference from a variable r of type S to a variable r' of type T? Compared to traditional type systems, the important restriction here is that we must not allow r' to gain more permissions than r via downcasting.

Assume that there exists a method m that is optional in S, but required in T. Table I shows that there are no security concerns in this situation, since both S and T allow to call m. However, since T asserts that the referenced object has method m, we must check this condition at run-time when assigning r to r'. Thus, we can assign $r : S$ to $r' : T$ *without* a run-time type check, if and only if

- there is no optional method in S that is required in T,
- each required method of T is also present in S,
- each method of S can be assigned to its corresponding method in T without a run-time check, i.e., all its arguments and results can be assigned without check (this avoids delayed type failures).

Using the rules shown in Figure 4, we denote this situation as $S \prec:_c T$.

A different situation arises if there exists a method m that is optional in T, but is not present in S. In this case, Table I shows that T actually allows to call m (which may, however, result in a run-time error, if the referenced object o does not provide that method), while S does not. Thus, we actually can assign $r : S$ to $r' : T$ in a secure way, if after this assignment r' references an object that does *not* provide m. We ensure this by using a coercion semantics, where the result of the assignment is a reference to a membrane for o that does not provide method m. Vice versa, this means that we can assign $r : S$ to $r' : T$ *without* introducing a membrane, if and only if

- each method of T is also declared in S, and

- all methods of S can be assigned to the corresponding method of T without a need for a membrane.

This is formalized in Figure 5. Note the rule CM-ARROW, which states that when we assign a method $m : S_1 \to S_2$ to a method $m' : T_1 \to T_2$, and must perform a run-time check or introduce a membrane for the method's argument or its result (i.e., $T_1 \not<: S_1$ or $S_2 \not<: T_2$), we need to wrap m with some code performing these tasks when it is called at run-time. This is done by introducing a membrane for the object providing m.

We can summarize the above considerations into a single relation $\prec:$ defined by the rules in Figure 6. $S \prec: T$ denotes that an assignment from $r : S$ to $r' : T$ is (statically) type safe, if and only if

- each required method of T is also declared in S, and
- all methods of S can be assigned to the corresponding method of T.

The latter is only the case, if the argument and result can be assigned *without* a run-time type check. This is a deliberate decision in order to avoid delayed type failures, as mentioned in the second paragraph of Section IV-A.

D. Run-Time Actions

With the relations introduced above, a component's code is well-typed, if for each assignment from $r : S$ to $r' : T$ the condition $S \prec: T$ holds. However, when at run-time an assignment with $S \not\prec:_c T$ is about to execute, we need to perform an additional type check. Likewise, when $S \not\prec:_m T$, we need to introduce a membrane. If we must both create a membrane and perform a run-time check, the run-time check will be the last action. For the following detailed discussion, we assume that variable $r : S$ contains a reference to an object o of class C.

During the run-time type check, we must simply verify that a reference to o can be assigned to r' without any further actions, i.e., $C <: T$.

Introducing a membrane is more complex. The coercion semantics requires that after $r : S$ has been assigned to $r' : T$, r' refers to an object (i.e., the membrane) that actually has type T. This membrane must be computed from the types S and T, and the class C of the object o referenced by r. We perform this computation in two steps: First, from S and T we compute the *cast action* a that needs to be performed by the membrane. This can be done at the component's load-time. Second, we apply a to the actual object o at run-time in order to obtain the concrete membrane. The rules to compute the cast action are shown in Figure 7. Informally speaking, the cast action instructs the membrane to (recursively) retain only those methods that are allowed by both S and T. Thus, the application of a cast action a to an object o results in a membrane for o that (recursively) provides just these methods. This is formalized in Figure 8, where f_a is the actual run-time operation performed for the cast action a.

A special case occurs, if o already is a membrane (with cast action a') for some other object o'. In this case, the membranes will not be cascaded, but applying a to o will result in a merged membrane for o', reflecting both cast actions a and a', as shown in Figure 9. Informally, the merge operation corresponds to the (recursive) intersection of the allowed methods.

We define the set $\mathcal{CA}$ of cast actions inductively as follows:

1) $\top \in \mathcal{CA}$
   $\top$ denotes that the membrane does not need to perform any action, i.e., no need for a membrane.
2) $\forall_{i=1..n} : a_i \in \mathcal{CA} \;\Rightarrow\; \{l_i : a_i \;^{i=1..n}\} \in \mathcal{CA}$
   This means that the membrane (just) provides the methods l_i that perform the actions defined by a_i.
3) $a \in \mathcal{CA} \;\wedge\; b \in \mathcal{CA} \;\Rightarrow\; (a, b) \in \mathcal{CA}$
   This denotes that a method first applies the action a to its argument, then forwards the call to the real object, and finally applies the action b to the result.

For two types S and T, the cast actions $ca(S, T)$ that need to be performed when assigning $r : S$ to $r' : T$ are defined by the following rules:

$$\text{CA-NONE}\;\frac{S \prec:_m T}{ca(S,T) = \top}$$

$$\text{CA-RCD}\;\frac{\begin{array}{c} S = \{(k_j : S_j,\, n_j) \;^{j\in 1..m}\} \\ T = \{(l_i : T_i,\, o_i) \;^{i\in 1..n}\} \\ S \prec: T \\ S \not\prec:_m T \end{array}}{ca(S,T) = \{l_i : ca(S_j, T_i) \;^{i\in 1..n,\; j\in 1..m,\; k_j = l_i}\}}$$

$$\text{CA-ARROW}\;\frac{\begin{array}{c} S = S_1 \to S_2 \\ T = T_1 \to T_2 \\ S \prec: T \\ S \not\prec:_m T \end{array}}{ca(S,T) = (ca(T_1, S_1), ca(S_2, T_2))}$$

Figure 7. Rules for the cast actions specifying the behavior of a membrane.

E. Downcasting

The type system outlined above achieves its security properties by strictly limiting downcast operations. However, this limitation is only necessary when code executing in a context x has a reference to an object in a different context y. If the reference points to an object o in the local context x, i.e., an object created by context x, there are no security issues at all. This is because x is able to retain the original (unrestricted) reference when it creates o, thus, it can not gain additional authority by downcasting any reference to o. So in order not to restrict the expressiveness of our type system, we should allow downcasting in this situation.

We achieve this by a simple extension: We distinguish between *local* types, which assert that references of this type will always point to objects contained in the local context, and *non-local* types, which do not provide such a guarantee. We then extend the subtyping rules from Figure 3, such that $S <: T$ does *not* hold if T is local, but S is non-local. It is still possible to assign from a variable r with a non-local type

$$\begin{array}{rcl} f_a(null) & = & null \\ f_\top(o) & = & o \\ f_{\{k_j:a_j\ {}^{j\in 1..m}\}}(\{l_i : m_i\ {}^{i\in 1..n}\}) & = & \{l_i : f_{a_j}(m_i)\ {}^{i\in 1..n,\ j\in 1..m,\ k_j=l_i}\} \\ f_{(a,b)}(m) & = & f_b \circ m \circ f_a \end{array}$$

Figure 8. Semantics of the generated membrane.

Assume that o is a membrane, i.e., $o = f_{\{k'_j:a'_j\ {}^{j\in 1..m}\}}(o')$. Then

$$\begin{array}{rcl} f_{\{k_j:a_j\ {}^{j\in 1..m}\}}(o) & = & f_{\{k_j:a_j\ {}^{j\in 1..m}\}}(f_{\{k'_j:a'_j\ {}^{j\in 1..m}\}}(o')) \;=\; (f_{\{k_j:a_j\ {}^{j\in 1..m}\}} \circ f_{\{k'_j:a'_j\ {}^{j\in 1..m}\}})(o') \\ & = & f_{\mathrm{merge}(\{k_j:a_j\ {}^{j\in 1..m}\},\{k'_j:a'_j\ {}^{j\in 1..m}\})}(o') \end{array}$$

where the function merge is defined by

$$\begin{array}{rcl} \mathrm{merge}(\top, a) & = & a \\ \mathrm{merge}(a, \top) & = & a \\ \mathrm{merge}(\{l_i : a_i\ {}^{i\in 1..n}\}, \{k_j : b_j\ {}^{j\in 1..m}\}) & = & \{l_i : \mathrm{merge}(a_i, b_j)\ {}^{i\in 1..n,\ j\in 1..m,\ k_j=l_i}\} \\ \mathrm{merge}((a_1, b_1), (a_2, b_2)) & = & (\mathrm{merge}(a_2, a_1), \mathrm{merge}(b_1, b_2)) \end{array}$$

Figure 9. Fusion of cascaded membranes.

S to a variable r' with local type T, provided that a run-time check is performed to ensure that r actually refers to an object in the local context. With this extension, we can allow all the traditional downcast operations when the source type is local.

F. An Example

Assume that we have the types `Calendar` and `Appointment` from Section III, as well as the three restricted interfaces shown in Listing 3.

LISTING 3. RESTRICTED CALENDAR PROVIDER INTERFACES

```
interface Provider1 {
  Event1 getNextAppointment();
}
interface Event1 {
  int startTime();
  int endTime();
}

interface Provider2 {
  Event2 getNextAppointment();
}
interface Event2 {
  int startTime();
  int endTime();
  optional String subject();
}

interface Provider3 {
  Event3 getNextAppointment();
}
interface Event3 {
  int startTime();
  int endTime();
  String subject();
}
```

In this example, the type `Calendar` can be converted to `Provider1` without any precautions, since `Calendar` $<:$ `Provider1` (Rules S-RCD and S-ARROW in Figure 3). Note that in this case the restricted access permissions, like the fact that it is not allowed to call `subject()` on the object returned by `getNextAppointment()`, are statically enforced by the type `Provider1` without a need for any run-time checks.

A further conversion from `Provider1` to `Provider2` is also allowed, since `Provider1` $\prec:$ `Provider2` (Rule C-RCD in Figure 6). However, we have `Provider1` $\not\prec:_m$ `Provider2` (Rules CM-RCD and CM-ARROW in Figure 5), since the members of `Event2` are a not a subset of the members of `Event1`. Thus, at run-time a membrane for the `Calendar` object is created that wraps the method `getNextAppointment()`, such that its return value is wrapped into a second membrane that does not have a method `subject()` (c.f. Figure 2). Now, the access restrictions are enforced by the membrane.

Our type system does not allow a conversion from `Provider2` to `Provider3`. Although this conversion would be safe if the reference actually points to an object of type `Calendar`, allowing it could result in unexpected run-time errors: A component that has a reference of type `Provider3` would assume that `getNextAppointment()` always returns an object that provides the method `subject()`. However, since `subject()` is declared as optional in `Event2`, the implementation of `getNextAppointment()` in the referenced object may actually return a reference that does not allow to call this method.

V. COSMA

The *Component Oriented Secure Machine Architecture* (COSMA) is a secure VM based on the outlined component model and type system. It comes with a specification for an object oriented byte code, called *Component Intermediate Language*. As outlined in Figure 10, the structure of this byte code

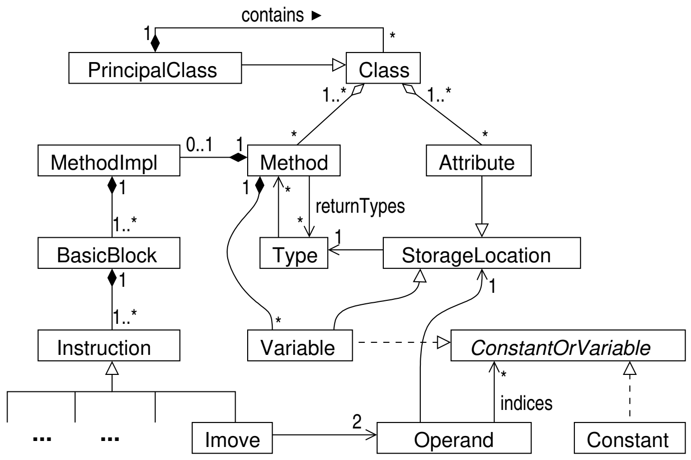


Figure 10. Simplified structure of a COSMA program

reflects that of a component: The entry point for a component's code always is its principal class, which logically contains all other classes. Method implementations are structured into basic blocks, which are sequences of instructions (cf. Figure 11). Basic blocks within the local method are the only admissible targets of branch instructions. Instructions do not allow direct access to the memory. Instead, they use typed operands to access abstract storage locations. There is also no visible call stack, but a high-level method call instruction, where lists of operands are passed for arguments and results. This ensures that a malicious program cannot forge references (e.g., by abusing an untyped stack), which is the major requirement for a secure object-capability system. Since the byte code does not contain any names except the obligatory method names in interfaces, it also protects the component developer's intellectual property rights.

We need a secured byte code, since secure high-level languages "*can still be attacked from below*" [36]. In order to prevent such attacks, we must use "*computers on which only capability-secure programs are allowed*" [36]. Thus, new programs can only be loaded into COSMA as components represented in our byte code.

When a component is deployed into the VM, it is associated with a new context that serves as a trust (or protection) unit. Within this context, the component's principal class is instantiated, and a reference (capability) to this *principal object* is returned and gets casted to the component's published interface. Initially, this reference is the only way to interact with the component. When an object in a context c creates another object, the new object also is associated with c. Thus, a context comprises all objects that are (transitively) created by the principal object of a loaded component. COSMA ensures that references can point to objects in a different context, only if they have a non-local type and therefore are subject to the security restrictions outlined in Section IV. References with local types can only point to objects in the local context. Thus, we do not restrict the code's expressiveness within a component.

During deployment, a component's complete byte code is checked for consistency, which includes type checking. Since the byte code does not allow any untyped data accesses, this

`call r m args res`
: Calls method m on the reference r passing $args$ as arguments and res as write-back operands.

`cjmp src nz block`
: Jumps to $block$ if $src \neq 0$ ($nz = \texttt{true}$) or if $src = 0$ ($nz = \texttt{false}$).

`chktype r t dst`
: Tests whether r has type t and stores the result in dst.

`inv r id args`
: Invokes a method whose name is stored in the string referenced by operand id on the reference r and passes $args$ as the arguments.

`jmp b`
: Jumps to block b.

`load const dst`
: Loads the constant $const$ into dst.

`mov src dst`
: Assigns src to dst.

`new c dst`
: Creates a new object of the given class c and stores the reference in dst.

`op lhs rhs op dst`
: Calculates $lhs\ o_{op}\ rhs$ (with $o_{op} \in \{+, -, *, /, \texttt{mod}, ...\}$) and stores the result in dst.

`ret r`
: Returns the given operands to the caller.

`test lhs rhs op dst`
: Tests for $lhs\ o_{op}\ rhs$ (with $o_{op} \in \{=, \neq, <, \leq, >, \geq\}$) and writes the result to dst.

Figure 11. Instructions supported by the virtual machine (without array-specific instructions).

can be done on a per-instruction basis, without a need for a complex verification of instruction sequences, as it is necessary, e.g., in Java byte-code [37]. Based on the type information available in the component's code, COSMA automatically generates the code for all required membranes, relieving the programmer from this burden. At run-time, membranes are automatically inserted via coercion semantics when security relevant downcasting is performed. Thus, security constraints are enforced mainly statically, leaving only a few run-time checks.

The first component that is deployed into the VM can receive a reference to a *native kernel object* as the first parameter of its constructor. This object offers basic operating system services (system calls), which are not expressible through the instruction set for security reasons. As described in Section III-B, the kernel object is part of the trusted computing base and built into COSMA. At the moment, we have implemented only a few indispensable services, thus, the reference has the interface type shown in Listing 4 (the type `Any` used here is a special type, which can be assigned to any reference type, using a run-time type check; the type `String` is currently implemented as an integer array).

LISTING 4. KERNEL INTERFACE

```
interface Kernel {
  // load component IR and
  // return principal object
  Any loadComponent(String filename);
  // print string to stdout
  void print(String msg);
  // read string from stdin
  String scan();
  // create a thread
  Thread createThread(Runnable r);
}
interface Thread {
  void start();
  void join();
}
interface Runnable {
  void run();
}
```

The initial component can share the reference to the kernel object with other components, while all the rules from Section IV still apply. In particular, utilization of the operating system can be restricted and checked at load time as if it were a normal object.

The basic idea is that the first component, wrapping the functionality of the native kernel object, provides additional operating system services with a cleaner and more powerful interface.

A. Implementation

We have implemented two variants of the COSMA virtual machine: A pure interpreter as a reference implementation of the machine's semantics (see Appendix), and a more realistic version that uses an ahead-of-time (AOT) compiler to translate a component's intermediate representation (IR) to native machine code at load time.

The AOT version of COSMA consists of two processes: One of them implements the loader and type checker, which are programmed in Java, while the other one executes the generated native code of all loaded components. When COSMA starts, the loader reads the IR of the initial component, performs type and consistency checks for each element of the IR, and then compiles the IR into C code, which is then compiled into a shared library using the GNU C compiler. This library is then dynamically loaded and linked into the native execution process using the dynamic linker (i.e., `dlopen()` in Linux systems).

The constructor of the initial component's principal class receives a reference to the kernel object described above. When the kernel's `loadComponent()` method is invoked, a synchronous request is sent to the loader process (using interprocess communication via UNIX pipes), which will then load the component's IR as described above.

When the native code executes an assignment, it may be necessary to perform a run-time type check and/or to introduce a membrane. Since a naïve implementation of these operations would require a recursive traversal of the involved type definitions, which is prohibitively expensive, we follow a different strategy. In the native code, each type is just represented as a globally unique integer number, which is assigned by the loader when it first sees the type. The loader also precomputes the relation $<:$ into a hash table, so that a type-check at run-time just requires a hash table lookup. The hash table is extended incrementally whenever a new component is loaded.

Furthermore, the loader precomputes the information that is needed to create the native code for the classes of all membranes that may be required at run-time. In Section IV-D, we have shown that when a reference r pointing to an object o of class C is converted from type S to T, the required membrane is determined in two steps:

1) compute the cast actions $ca(S,T)$ from S and T,
2) generate the membrane for o from $ca(S,T)$ and C.

In order to generate all membrane classes that may be required at run-time, we execute a simple data flow analysis that determines for each statement s performing a relevant type cast from S to T the set of all classes C, such that a reference to an object of class C may reach s. We then generate the code for the required membrane class from $ca(S,T)$ and C. Since the instances of these membrane classes may again need to be wrapped by a membrane, resulting in new membrane classes, we perform a fixed point iteration, which stops when the set of membrane classes does not change any more. The termination of this fixed point iteration is actually guaranteed by the fact that we avoid cascading of membranes by fusing them as outlined in Fig 9. As with the subtype relation, we incrementally expand the set of generated membrane classes whenever a new component is loaded.

Although the creation of all possibly required membrane classes at load-time requires additional time and storage when a component is loaded, it makes the creation of membranes at run-time extremely fast: We just need a hash table lookup (with the type number of the object's class as the key) to find the membrane class that must be instantiated.

B. Performance Considerations

Since in the majority of cases, the necessary access restrictions are enforced statically by COSMA's type system, we can achieve fine grained access control between components with minimal run-time overhead. This overhead, as compared to traditional designs for object oriented virtual machines, arises from the following three sources:

- The IR does not have a fixed format, like traditional byte codes, and therefore cannot be directly interpreted with comparable efficiently. However, as modern virtual machines are based on just-in-time or ahead-of-time compilation techniques, an efficient direct interpretation of the IR is no longer a necessity. If really required, a simple ahead-of-time compiler could easily transform the IR into a fixed format byte code at load time.
- Method calls on component interfaces require a more expensive dispatch mechanism, since we use a modified form of structural typing instead of nominal

typing. However, there is a number of established methods to minimize this overhead [38] [39].

- Finally, we need an additional method call for each method invoked via a membrane. Ignoring possible low-level optimizations, this effectively doubles the time required for an (empty) method call. However, membranes are only introduced for component interfaces that include optional methods, thus, this situation will not occur at high frequencies. We also will investigate the benefit of dynamically testing when membranes can safely be removed again. In the example in Section IV-F, the membrane introduced when the reference to the `Calendar` object was converted from `Provider1` to `Provider2` can be removed again when the reference is converted back to `Provider1`. The problem is that we need to trade the time required for this run-time test when assigning a reference against the time saved when calling methods via this reference. Thus, this optimization requires a more elaborate analysis during the ahead-of-time compilation, which is part of our future work.

So the only unavoidable run-time overhead of our approach is due to the introduction of membranes. However, membranes are only needed when a decision on the granted permissions should be possible at run-time, which imperatively implies that also the permission checks must be performed at run-time.

In summary, the implementation of COSMA proves that using our intermediate representation and type system, we can efficiently execute components while offering a high degree of protection by enforcing fine grained access permissions.

VI. Conclusion and Future Work

In this paper, we proposed a novel concept for the secure cooperation of untrusted components. This involves a component model, where each component declares its required and granted permissions via a self-explanatory public interface. This interface can then be used to connect it to other components. Components are distributed in the form of a secure byte code with high-level instructions that preserves typing information, but still protects intellectual property rights. The corresponding VM implements a type system ensuring that a component cannot gain more permissions than those explicitly defined in its public interface. Type checking is done at deployment time, with some additional run-time checks, where necessary. Coercion semantics is used to automatically insert membranes.

At present we have a fully operational implementation of the type system and the VM, as well as a compiler translating a Java-like language into our byte code. The implementation is freely available at the COSMA website [40]. A formal specification of the type system, including subtyping and coercion, is also available, along with the semantics of the implemented instructions and a formal proof that no instruction sequence can amplify a component's permissions (see Appendix).

In the current implementation all components are executed by the same VM, thus, security of network connections is not an issue. In the future, the model can be extended to distributed systems using remote method invocation, provided that the communication link between the VMs uses a secure protocol ensuring authentication and integrity.

We are currently working on an extension of our type system that integrates structural and nominal typing into a unified framework by explicitly considering the required semantics of methods. In this way, we enable a flexible and safe reuse of components and at the same time fulfill all the subtyping desiderata outlined in [41] (i.e., flexible assignment of responsibility, modular extensibility of the subtyping relation, unique name introduction and traceability) without the idiosyncrasies of that approach, especially avoiding non-transitiveness of the subtype relation.

We are also working on a more complete operating system that meets the special requirements of safe and secure component-based software. This includes in particular memory management for strongly interacting components, such as paging and garbage collection. The latter is especially interesting in combination with fault tolerance and error recovery: If an error occurs in one component, the effects for all other components interacting with it must be as small as possible. To this end, we are looking for a solution to make reloading or replacing a component mostly transparent to its users.

Based on our current work, we will investigate the performance of our VM in more detail, comparing it against plain Java, in order to assess the costs for run-time checks and the indirection caused by the use of membranes. Other topics, which we will address in future are the integration of alias control (cf. Section II), error and exception handling, mechanisms for the revocation of permissions and optimizations such as the removal of unnecessary membranes discussed in Section V-B. Our ultimate goal is to provide a complete programming system, consisting of a VM, an OS and a high-level language compiler, which can be used to develop and deploy component-based software in an easy, secure and efficient way.

References

[1] R. Wismüller and D. Ludwig, "Secure Cooperation of Untrusted Components," in Twelfth Intl. Conf. on Emerging Security Information, Systems and Technologies (SECURWARE 2018). Venice, Italy: IARIA, Sep. 2018, pp. 103–107.

[2] S. Maffeis, J. C. Mitchell, and A. Taly, "Object Capabilities and Isolation of Untrusted Web Applications," in Proc. of IEEE Symp. Security and Privacy. Oakland, CA, USA: IEEE, May 2010, pp. 125–140.

[3] M. S. Miller and J. S. Shapiro, "Paradigm Regained: Abstraction Mechanisms for Access Control," in Advances in Computing Science - ASIAN 2003. Progamming Languages and Distributed Computation, ser. LNCS, vol. 2896. Springer, 2003, pp. 224–242.

[4] M. S. Miller, "Robust composition: Towards a unified approach to access control and concurrency control," Ph.D. Thesis, Johns Hopkins University, Baltimore, Maryland, May 2006.

[5] T. Murray, "Analysing object-capability security," in Proc. of the Joint Workshop on Foundations of Computer Security, Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security, Pittsburgh, PA, USA, Jun. 2008, pp. 177–194.

[6] A. Mettler, D. Wagner, and T. Close, "Joe-E: A Security-Oriented Subset of Java," in Network and Distributed Systems Symposium. Internet Society, Jan. 2010, pp. 357–374.

[7] M. Stiegler, "Emily: A High Performance Language for Enabling Secure Cooperation," in Fifth Intl. Conf. on Creating, Connecting and Collaborating through Computing C5'07. Kyoto, Japan: IEEE, Jan. 2007, pp. 163–169.

[8] D. A. Gutierrez, F. T. Martínez, F. A. García, M. A. D. Fondón, R. I. Castanedo, and J. M. C. Lovelle, "An Object-Oriented Abstract Machine as the Substrate for an Object-Oriented Operating System," in Object-Oriented Technology ECOOP, Workshop Reader, ser. LNCS, vol. 1357. Jyväskylä, Finland: Springer, Jun. 1997, pp. 537–544.

[9] R. P. e Silva and R. T. Price, "Component Interface Pattern," in Proc. 9th Conf. on Pattern Language of Programs, Monticello, IL, USA, Sep. 2002. [Online]. Available: http://hillside.net/plop/plop2002/final/plop2002_rpsilva0_1.pdf [last access: 17.05.2019]

[10] S. Mouelhi, K. Agrou, S. Chouali, and H. Mountassir, "Object-Oriented Component-Based Design using Behavioral Contracts: Application to Railway Systems," in Proc. 18th International ACM SIGSOFT Symposium on Component-Based Software Engineering (CompArch '15). Montreal, QC, Canada: ACM, May 2015, pp. 49–58.

[11] F. Bachmann, L. Bass, C. Buhman, S. Comella-Dorda, F. Long, J. Robert, R. Seacord, and K. Wallnau, "Volume II: Technical Concepts of Component-Based Software Engineering, 2nd Edition," Carnegie Mellon University, Software Engineering Institute, Pittsburgh, PA, USA, Technical Report CMU/SEI-2000-TR-008, ESC-TR-2000-007, May 2000. [Online]. Available: https://resources.sei.cmu.edu/asset_files/TechnicalReport/2000_005_001_13715.pdf [last access: 17.05.2019]

[12] J. Andronick, D. Greenaway, and K. Elphinstone, "Towards proving security in the presence of large untrusted components," in Proc. 5th Intl. Workshop on System Software Verification, Vancouver, BC, Canada, Oct. 2010. [Online]. Available: https://www.usenix.org/legacy/events/ssv10/tech/full_papers/Andronick.pdf [last access: 17.05.2019]

[13] H. M. Levy, Capability-Based Computer Systems. Digital Press, 1984. [Online]. Available: http://homes.cs.washington.edu/∼levy/capabook [last access: 17.05.2019]

[14] M. S. Miller, B. Tulloh, and J. S. Shapiro, "The Structure of Authority: Why Security Is not a Separable Concern," in Proc. 2nd Intl. Conf. on Multiparadigm Programming in Mozart/Oz. Charleroi, Belgium: Springer, 2004, pp. 2–20.

[15] M. S. Miller, K. P. Yee, and J. Shapiro, "Capability Myths Demolished," Systems Research Laboratory, Johns Hopkins University, Technical Report SRL2003-02, 2003. [Online]. Available: http://srl.cs.jhu.edu/pubs/SRL2003-02.pdf [last access: 17.05.2019]

[16] M. A. D. Fondon, D. A. Gutierrez, L. T. Martinez, F. A. Garcia, and J. M. C. Lovelle, "Capability-based protection for integral object-oriented systems," in Proc. Computer Software and Applications Conf. COMPSAC '98. Vienna, Austria: IEEE, Aug. 1998, pp. 344–349.

[17] M. A. D. Fondon, D. A. Gutierrez, A. G. M. Sánchez, F. A. García, F. T. Martínez, and J. M. C. Lovelle, "Integrating capabilities into the object model to protect distributed object systems," in Proc. Intl. Symp. on Distributed Objects and Applications. Edinburgh, GB: IEEE, Sep. 1999, pp. 374–383. [Online]. Available: http://dx.doi.org/10.1109/DOA.1999.794067 [last access: 17.05.2019]

[18] P. Müller and A. Poetzsch-Heffter, "Universes: A type system for controlling representation exposure," in Programming Languages and Fundamentals of Programming, A. Poetzsch-Heffter and J. Meyer, Eds. Fernuniversität Hagen, 1999, pp. 131–140, Technical Report 263.

[19] W. Dietl and P. Müller, "Ownership Type Systems and Dependent Classes," in Intl. Workshop on Foundations of Object-Oriented Languages (FOOL'08), San Francisco, CA, USA, Jan. 2008.

[20] D. G. Clarke, J. M. Potter, and J. Noble, "Ownership types for flexible alias protection," in Proc. 13th ACM SIGPLAN Conf. on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA '98), Vancouver, Canada, Oct. 1998, pp. 48–64.

[21] S. Balzer, T. Gross, and P. Müller, "Selective ownership: Combining object and type hierarchies for flexible sharing," in Foundations of Object-Oriented Languages (FOOL), J. Boyland, Ed., 2012.

[22] B. Bokowski and J. Vitek, "Confined Types," in Proc. 14th ACM SIGPLAN Conf. on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA '99), Denver, CO, USA, Nov. 1999, pp. 82–96.

[23] P. W. L. Fong, "Link-Time Enforcement of Confined Types for JVM Bytecode," in Proc. 3rd Annual Conf. on Privacy, Security and Trust (PST'05), St. Andrews, Canada, Oct. 2005, pp. 191–202.

[24] ——, "Discretionary capability confinement," in Proc. 11th European Symposium On Research In Computer Security (ESORICS'06), ser. LNCS, vol. 4189. Hamburg, Germany: Springer, Sep. 2006, pp. 127–144.

[25] ——, "Discretionary Capability Confinement," International Journal of Information Security, vol. 7, no. 2, Apr. 2008, pp. 137–154.

[26] D. Volpano and G. Smith, "A Type-Based Approach to Program Security," in TAPSOFT '97: Theory and Practice of Software Development, ser. LNCS, M. Bidoit and M. Dauchet, Eds., vol. 1214. Springer, 1997, pp. 607–621.

[27] A. Gollamudi and S. Chong, "Automatic Enforcement of Expressive Security Policies using Enclaves," in Proc. 2016 ACM SIGPLAN Intl. Conf. on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA '16), Amsterdam, Netherlands, Nov. 2016, pp. 494–513.

[28] P. W. L. Fong and C. Zhang, "Capabilities as alias control: Secure cooperation in dynamically extensible systems," Dept. of Computer Science, Univ. of Regina, Regina, Canada, Technical Report CS-2004-3, Apr. 2004.

[29] D. Hagimont, J. Mossière, X. R. de Pina, and F. Saunier, "Hidden Software Capabilities," in Proc. 16th Intl. Conf. on Distributed Computing Systems (ICDCS '96), Hong Kong, May 1996, pp. 282–289.

[30] J. Boyland, J. Noble, and W. Retert, "Capabilities for Sharing - A Generalisation of Uniqueness and Read-Only," in 15th European Conf. on Object-Oriented Programming (ECOOP '01), Budapest, Hungary, Jun. 2001, pp. 2–27.

[31] P. W. L. Fong and S. Orr, "A Module System for Isolating Untrusted Software Extensions," in Proc. 22nd Annual Computer Security Applications Conf. (ACSAC'06), Miami Beach, Florida, USA, Dec. 2006, pp. 203–212.

[32] C. Hawblitzel, C. C. Chang, G. Czajkowski, D. Hu, and T. von Eicken, "SLK: A Capability System Based on Safe Language Technology," Cornell University, Technical Report, Mar. 1997. [Online]. Available: http://www.cs.cornell.edu/slk/papers/slk.pdf [last access: 17.05.2019]

[33] C. Szyperski, Component Software: Beyond Object-Oriented Programming, 2nd ed. Boston, MA, USA: Addison-Wesley, 2002.

[34] B. C. Pierce, Types and programming languages. MIT Press, 2002.

[35] L. Cardelli, "Typeful Programming," in Formal Description of Programming Concepts, E. Neuhold and M. Paul, Eds. Springer, 1991, pp. 431–507.

[36] M. Stiegler, "The E Language in a Walnut," 2000. [Online]. Available: http://www.skyhunter.com/marcs/ewalnut.html [last access: 17.05.2019]

[37] X. Leroy, "Java bytecode verification: Algorithms and formalizations," Journal of Automated Reasoning, vol. 30, no. 3, May 2003, pp. 235–269. [Online]. Available: https://doi.org/10.1023/A:1025055424017 [last access: 17.05.2019]

[38] A. M. Schiffman and L. P. Deutsch, "Efficient Implementation of the Smalltalk-80 System," in Proc. 11th ACM SIGACT-SIGPLAN Symp. on Principles of Programming Languages (POPL '84), 1984, pp. 297–302.

[39] J. Gil and I. Maman, "Whiteoak: Introducing Structural Typing into Java," in Proc. 23rd ACM SIGPLAN Conf. on Object-Oriented Programming Systems Languages and Applications (OOPSLA '08), Nashville, TN, USA, Oct. 2008, pp. 73–90.

[40] "COSMA – A Virtual Machine Supporting the Secure Cooperation of Untrusted Components." [Online]. Available: https://www.bs.informatik.uni-siegen.de/forschung/cosma [last access: 28.05.2019]

[41] K. Ostermann, "Nominal and Structural Subtyping in Component-Based Programming," Journal of Object Technology, vol. 7, no. 1, Jan. 2008. [Online]. Available: http://www.jot.fm/issues/issue_2008_01/article4 [last access: 17.05.2019]

Appendix

In this section we present the semantics of COSMA's instruction set. The auxiliary rules we use here can be found in Figure 12. We also provide a proof sketch that the semantics successfully prevents an amplification of access rights.

A. Basic Definitions

According to the structure of the intermediate representation, we use the following (abstract) types (cf. Figure 10):

- V for instances of `Variable`
- F for instances of `Attribute`
- $L = V \cup F$ for storage locations
- B for instances of `BasicBlock`
- T for instances of `Type`
  - $\tau_{ref} \in T$ is a reference type
  - $\tau_{obj} \in T$ is an object type
  - $\tau_{val} \in T$ is a value type
- Id for method names
- $\mathbb{N}, \mathbb{Z}$ as usual
- $\mathbb{B}$ for Boolean values
- Impl : (name : Id) $\times$ (var : V^*) $\times$ (nargs : $\mathbb{N}$) $\times$ (resT : T^*) $\times$ (blocks : B^*) $\times$ (clazz : Class) for method implementations
- Decl : (name : Id) $\times$ (opt : $\mathbb{B}$) $\times$ (argT : T^*) $\times$ (resT : T^*) for method declarations
- Class : (pub : Impl^*) $\times$ (priv : Impl^*) $\times$ (attr : F^*) $\times$ (child : Class^*) for classes and class types
- Iface : (methods : Decl^*) for interface types.
- $\mathcal{CA}$ for type cast actions
- RT for generic run-time values
  - Val for numeric run-time values
  - Obj : (methods : Impl^*) $\times$ (config : F^*) for objects
  - Mem : (obj : Obj) $\times$ (actions : $(\mathit{Id} \times \mathcal{CA}^*)^*$) for membranes
- Γ for the type environment

A *Frame* is defined as a tuple (obj : Obj) $\times$ (method : Impl) $\times$ (block : $\mathbb{N}$) $\times$ (pc : $\mathbb{N}$) $\times$ (var : V^*) $\times$ (res : L^*) $\times$ (resT : T^*) $\times$ (mem : f_A(obj)), where obj represents the current object, $method$ the currently executing method with the basic block number $block$ and its program counter pc. The current values of the method's variables are stored in var, and res is a list of operands to write back the method's results to the caller. $resT$ holds the result types expected by the caller. If the current method was called on a membrane, a reference to the membrane is stored in mem. For simplification we assume that the last instruction in each basic block is either a return statement or an unconditional jump into the following basic block, i.e., we do not need to model overflows of the program counter. A *state* S is a stack of frames. We write $s :: t$ to split S into the topmost frame s and the tail t.

$\mathit{BlockIndex}(m, b) = i$ holds, if there exists exactly one basic block in m's implementation that *is* b and this block is the i-th block in m.

$$\textsc{BlockIndex}\ \frac{m : \mathit{Impl} \quad b : B \quad \exists_1 i \geq 0 : m.\mathit{blocks}_i = b}{\mathit{BlockIndex}(m, b) = i}$$

To lookup a method implementation for a method named id inside an object o:

$$\textsc{LookUp}\ \frac{\begin{array}{c} o : \mathit{Obj} \\ \exists_1 m \in o.\mathit{methods} : m.\mathit{name} = id \end{array}}{\mathit{lookup}(id, o) = m}$$

Object creation follows the Instantiate rule:

$$\textsc{Instantiate}\ \frac{\begin{array}{c} \mathit{class} : \mathit{Class} \\ \langle \mathit{class.pub} \cup \mathit{class.priv}, \mathit{class.attr} \rangle = o \end{array}}{\mathit{Instantiate}(\mathit{class}) = o}$$

Figure 12. Semantic functions and auxiliary rules

All semantic rules are formulated as a state transition of one stack configuration to another. Possible effects of these transitions are:

1) the top-most frame is changed,
2) a new frame is pushed onto the stack,
3) the top-most frame is removed from the stack, or
4) any meaningful combination of that.

If a frame is changed, we only write down the differences in the transition. Everything remaining unchanged is not shown. If for example the program counter is increased, but everything else is not touched, we just write $s[pc \leftarrow pc + 1] :: t$. For new frames we assume that the program counter pc and the block number $block$ are set to 0 and refrain from writing this explicitly.

B. Constants, Branching and Arithmetic Operations

In this subsection we present the semantics for simple instructions, such as conditional and unconditional jumps, loading constants and arithmetic operations and relations.

$$\textsc{Jmp}\ \frac{\mathit{BlockIndex}(s.\mathit{method}, b) = \mathit{blk}}{s :: t \xrightarrow{\texttt{jmp b}} s[\mathit{block} \leftarrow \mathit{blk}, pc \leftarrow 0] :: t}$$

Jmp jumps to the given block and resets the program counter. It holds, iff the given block is part of the current method.

$$\textsc{Cjmp-T}\ \frac{\begin{array}{c} \mathit{BlockIndex}(s.\mathit{method}, b) = \mathit{blk} \\ v \rightarrow \mathit{val} : \mathit{Val} \\ (v \neq 0) = \mathit{nz} \end{array}}{s :: t \xrightarrow{\texttt{cjmp v nz b}} s[\mathit{block} \leftarrow \mathit{blk}, pc \leftarrow 0] :: t}$$

Cjmp-T jumps to the given block and resets the program counter. It holds, if:

1) the block is part of the current method,

2) v contains a value val (e.g., integer),
3) $val \neq 0$ and nz is set to true (jump on non-zero), or $val = 0$ and nz is false.

$$\textsc{Cjmp-F}\;\dfrac{\begin{array}{c} \mathit{BlockIndex}(s.method, b) = blk \\ v \to val : \mathit{Val} \\ (v \neq 0) \neq nz \end{array}}{s :: t \xrightarrow{\texttt{cjmp v nz b}} s[pc \leftarrow pc + 1] :: t}$$

CJMP-F increments the program counter. It holds, iff CJMP-T fails only on the third premise.

$$\textsc{Load}\;\dfrac{const : \mathbb{Z} \quad \Gamma \vdash dst : \tau_{ref} \Rightarrow const = 0}{s :: t \xrightarrow{\texttt{load const dst}} s[pc \leftarrow pc + 1, dst \leftarrow const] :: t}$$

LOAD loads the constant $const$ into the operand dst and increases the program counter. It holds for every dst having a value type. If dst has a reference type, $const$ must be 0 ("*null*").

$$\textsc{Op}\;\dfrac{\begin{array}{c} lhs \to val_1 : \mathit{Val} \\ rhs \to val_2 : \mathit{Val} \\ \Gamma \vdash dst : \tau_{val} \\ f \in \{\div, -, \texttt{mod}, \cdot, +\} \end{array}}{\begin{array}{c} s :: t \xrightarrow{\texttt{op lhs rhs f dst}} \\ s[pc \leftarrow pc + 1, dst \leftarrow val_1 \circ_f val_2] :: t \end{array}}$$

$$\textsc{Test}\;\dfrac{\begin{array}{c} lhs \to val_1 : \mathit{Val} \\ rhs \to val_2 : \mathit{Val} \\ \Gamma \vdash dst : \tau_{val} \\ f \in \{=, \neq, <, \geq, >, \leq\} \end{array}}{\begin{array}{c} s :: t \xrightarrow{\texttt{test lhs rhs f dst}} \\ s[pc \leftarrow pc + 1, dst \leftarrow val_1 \circ_f val_2] :: t \end{array}}$$

OP and TEST increase the program counter and calculate $val_1 \circ_f val_2$ to store it in the given operand dst. Both rules hold, if:

1) both source operands point to values,
2) the target operand has a value type and
3) the operator is valid.

C. Object Creation and Assignments

$$\textsc{New}\;\dfrac{\begin{array}{c} c : \mathit{Class} \\ \Gamma \vdash c : \tau_1 \\ \Gamma \vdash dst : \tau_2 \\ \mathit{Instantiate}(c) = o \\ f_{ca(\tau_1,\tau_2)}(o) = o\prime \end{array}}{s :: t \xrightarrow{\texttt{new c dst}} s[pc \leftarrow pc + 1, dst \leftarrow o\prime] :: t}$$

NEW increases the program counter and stores a reference to a newly created object (or to a membrane for that object). It holds, if:

1) the given class has type τ_1,
2) the target operand has type τ_2, and
3) an object o is created from the given class and
4) can be assigned to the target operand.

$$\textsc{Mov}\;\dfrac{\begin{array}{c} \Gamma \vdash src : \tau_1 \\ \Gamma \vdash dst : \tau_2 \\ src \to o : RT \\ f_{ca(\tau_1,\tau_2)}(o) = o\prime \end{array}}{s :: t \xrightarrow{\texttt{mov src dst}} s[pc \leftarrow pc + 1, dst \leftarrow o\prime] :: t}$$

MOV increases the program counter and assigns src to dst, performing the necessary cast actions. It holds, if src is initialized, and o is assignable to the target operand.

D. Calling Methods and Returning Results

$$\textsc{ChkType}\;\dfrac{\begin{array}{c} \Gamma \vdash r : \tau_{ref} \\ t : T \\ \tau_{ref} \prec: T = \text{assignable} : \mathbb{B} \end{array}}{\begin{array}{c} s :: t \xrightarrow{\texttt{chktype r t dst}} \\ s[pc \leftarrow pc + 1, dst \leftarrow \text{assignable}] :: t \end{array}}$$

CHKTYPE increases the program counter and writes **True** to dst, iff r is assignable to the given type t.

$$\textsc{Call}\;\dfrac{\begin{array}{c} r \to o : \mathit{Obj} \;\wedge\; \Gamma \vdash r : \tau \;\wedge\; \tau : \mathit{Iface} \\ m : \mathit{Decl} \\ m \in \tau.methods \\ \mathit{lookup}(m.name, o) = m\prime : \mathit{Impl} \\ \mathit{args} = (a_1, \ldots, a_n) \;\wedge\; \Gamma \vdash a_i : \tau_i \\ n = m\prime.nargs \\ (f_{ca(\tau_1, m.argT_1)}(a_1), \ldots, f_{ca(\tau_n, m.argT_n)}(a_n)) = \mathit{Var}\prime \\ m.resT : T^k \;\wedge\; res : L^k \end{array}}{\begin{array}{c} s :: t \xrightarrow{\texttt{call r m args res}} \\ [o, m\prime, \mathit{Var}\prime, res, m.resT, \bot] :: s :: t \end{array}}$$

CALL creates a new frame containing the target object, the called method implementation and the method's list of variables including the parameters. The frames also carries the result operands and the result types, as expected by the caller. The rule holds, if:

1) r points to an object o and has type τ,
2) m is a method declaration and available in τ,
3) o's class type has an implementation of m, and
4) the given arguments can be casted to the formal parameter types of m,
5) the number of formal result types matches the given write-back operands.

$$\textsc{Call-M}\;\dfrac{\begin{array}{c} r \rightarrow o' = f_A(o) \ \wedge \ o : \mathit{Obj} \ \wedge \ \Gamma \vdash r : \tau \\ m : \mathit{Decl} \\ m \in \tau.\mathit{methods} \\ \mathit{lookup}(m.\mathit{name}, o) = m' : \mathit{Impl} \\ \mathit{args} = (a_1, \dots, a_n) \ \wedge \ \Gamma \vdash a_i : \tau_i \\ n = m'.\mathit{nargs} \\ (f_{ca(\tau_1, m.\mathit{argT}_1)}(a_1), \dots, f_{ca(\tau_n, m.\mathit{argT}_n)}(a_n)) = \mathit{Var}' \\ m.\mathit{resT} : T^k \ \wedge \ \mathit{res} : L^k \\ (m.\mathit{name}, (A_1, \dots, A_n)) \in o'.\mathit{actions} \\ (f_{A_1}(\mathit{Var}'_1), \dots, f_{A_n}(\mathit{Var}'_n)) = \mathit{Var}'' \end{array}}{\begin{array}{c} s :: t \xrightarrow{\texttt{call r m args res}} \\ [o, m', \mathit{Var}'', \mathit{res}, m.\mathit{resT}, o'] :: s :: t \end{array}}$$

CALL-M does the same as CALL, but additionally adds the membrane to the new frame. The rule holds, if:

1) r points to a membrane o' for object o,
2) CALL would hold for $r \mapsto o$, and
3) o' holds method actions for m that can be applied to the casted parameters.

$$\textsc{Inv-M}\;\dfrac{\begin{array}{c} r \rightarrow o' = f_A(o) \ \wedge o : \mathit{Obj} \ \wedge \ \Gamma \vdash r : \mathit{Any} \\ \mathit{id} \rightarrow \mathit{name} : \mathit{Id} \\ \mathit{lookup}(\mathit{name}, o) = m : \mathit{Impl} \\ \mathit{args} = (a_1, \dots, a_n), \Gamma \vdash a_i : \mathit{Any} \\ n = m.\mathit{nargs} \\ m.\mathit{resT} : () \\ (m.\mathit{name}, (A_1, \dots, A_n)) \in o'.\mathit{actions} \\ (f_{A_1}(a_1), \dots, f_{A_n}(a_n)) = \mathit{Var}' \end{array}}{s :: t \xrightarrow{\texttt{inv r id args}} [o, m', \mathit{Var}', (), (), o'] :: s :: t}$$

INV-M creates a new frame containing the target object and the called method implementation, as well as the given arguments. Since `inv` is only allowed on references of type `Any`, the reference always points to a membrane, which is added to the new frame. Result operands and result types are not set, since the instruction does not support results. The rule holds, if

1) r r has type *Any* and points to a membrane o' for o,
2) o has an implementation for a method named *name*,
3) all arguments have type *Any*,
4) o' holds method actions for the called method, which can be applied to the arguments.

$$\textsc{Ret}\;\dfrac{\begin{array}{c} m : \mathit{Impl} \ \wedge \ m.\mathit{resT} : T^n \\ r : L^n \ \wedge \ \Gamma \vdash r_i : \tau_i \ \wedge \ r_i \mapsto o_i : \mathit{RT} \\ \mathit{res} : L^n \ \wedge \ \Gamma \vdash \mathit{res}_i : \eta_i \\ \mathit{resT} : T^n \\ f_{ca(\mathit{resT}_i, \eta_i)}(f_{ca(\tau_i, m.\mathit{resT}_i)}(r_i)) = \mathcal{V}_i \end{array}}{\begin{array}{c} [\dots, m, \dots, \mathit{res}, \mathit{resT}, \bot] :: s :: t \xrightarrow{\texttt{ret r}} \\ s[pc \leftarrow pc + 1, \mathit{res}_i \leftarrow \mathcal{V}_i] :: t \end{array}}$$

RET removes the top-most frame (the callee frame) from the stack and increases the program counter in the caller frame. The operand list *res* in the callee frame references storage locations available in the caller frame. Writing to these operands changes the caller frame, which is the intended behavior. The instruction performs two typecasts. First, the actual results r_i must be casted to the formal result type of the method implementation $m.\mathit{resT}_i$. Finally, we need to apply a cast action, casting from the interface's result type resT_i to the target operand's type η_i.

$$\textsc{Ret-M}\;\dfrac{\begin{array}{c} m : \mathit{Impl} \ \wedge \ m.\mathit{resT} : T^n \\ r : L^n \ \wedge \ \Gamma \vdash r_i : \tau_i \ \wedge \ r_i \mapsto o_i : \mathit{RT} \\ \mathit{res} : L^n \ \wedge \ \Gamma \vdash \mathit{res}_i : \eta_i \\ (m.\mathit{id} \times (A_1, \dots, A_n)) \in o'.\mathit{actions} \\ \mathit{resT} : T^n \\ f_{ca(\mathit{resT}_i, \eta_i)}(f_{A_i}(f_{ca(\tau_i, m.\mathit{resT}_i)}(r_i))) = \mathcal{V}_i \end{array}}{\begin{array}{c} [\dots, m, \dots, \mathit{res}, \mathit{resT}, o'] :: s :: t \xrightarrow{\texttt{ret r}} \\ s[pc \leftarrow pc + 1, \mathit{res}_i \leftarrow \mathcal{V}_i] :: t \end{array}}$$

In case the method was invoked through a membrane as in RET-M, an additional cast is needed between the two type casts mentioned in RET. This cast addresses differences between the method implementation's formal result types and interface's result types.

E. Proof Sketch: No Amplification of Rights

Assumption 1. *X is a context, and $o \in X$ is an object providing method m. X' is a different context ($X' \neq X$) holding references that point to o or a membrane for o. For all references $r \in X'$ pointing to o or a membrane for o, we assume that m is not callable.*

Notation: *For a method m and an interface type T we write $m \in T \Leftrightarrow (m : U, \eta) \in T$. For a reference r we write $m \in r \Leftrightarrow \Gamma \vdash r : T \wedge m \in T$. For a membrane mem we write $m \in \mathit{mem} \Leftrightarrow (m.\mathit{name}, \mathit{ca}) \in \mathit{mem}.\mathit{actions}$. For cast actions $\mathit{ca}(S, T)$ we write $m \in \mathit{ca}(S, T) \Leftrightarrow (m : A) \in \mathit{ca}(S, T)$.*

With this notation, the assumption can be formalized as

$$\forall r \in X' : \begin{cases} \Gamma \vdash r : T \ \wedge \ m \notin T & r \mapsto o \\ m \notin \mathit{membrane}(o) & r \mapsto \mathit{membrane}(o) \end{cases}$$

Observation 1. *For assignments from type $S = \{(k_j : S_j, \eta_j)^{j \in 1 \dots z}\}$ to type $T = \{(l_i : T_i, \sigma_i)^{i \in 1 \dots n}\}$ we have three cases:*

1) $S \not\prec: T$
   $\Rightarrow$ *assignment is not possible* $\Rightarrow$ *no need to consider*
2) $S \prec:_m T$
   $\Rightarrow \{{l_i}^{i \in 1 \dots n}\} \subseteq \{{k_j}^{j \in 1 \dots z}\} \ \wedge \ \mathit{ca}(S, T) = \top$
3) $S \prec: T \ \wedge \ S \not\prec:_m T$
   $\Rightarrow \mathit{ca}(S, T) = \{l_i : \mathit{ca}(S_j, T_i)^{\ i \in 1 \dots n,\ j \in 1 \dots z,\ k_j = l_i}\}$

Theorem 1. *There is no sequence of instructions executed in context X' involving any $r \in X'$ that results in successfully calling m on o, unless m is called by some method in a different context X'' or X'' returns a reference to o allowing m.*

Proof: COSMA supports 14 instructions of which four (`mov`, `call`, `ret`, `inv`) are able to create or copy object references. The new-instruction is of no interest. We perform an induction over the length ℓ of the instruction sequence. For $\ell = 0$, m is not callable on any reference $r \in X\prime$ by assumption (induction hypothesis).

Instruction: `mov r r′`
Let $r \in X\prime$ be a reference of type S and $r\prime$ a new reference of type T.

1) $S \prec:_m T$
   a) $r \mapsto o \overset{\text{Ass}}{\Rightarrow} m \notin S \Rightarrow m \notin T \overset{\text{Mov}}{\Rightarrow} r\prime = r \Rightarrow m \notin r\prime$
   b) $r \mapsto o\prime = f_{ca(S\prime,T\prime)}(o) \overset{\text{Ass}}{\Rightarrow} m \notin ca(S\prime,T\prime) \overset{\text{Mov}}{\Rightarrow} r\prime \mapsto o\prime \Rightarrow m \notin r\prime$
2) $S \prec: T \wedge S \not\prec:_m T$
   a) $r \mapsto o \overset{\text{Ass}}{\Rightarrow} m \notin S \Rightarrow m \notin ca(S,T) \Rightarrow m \notin f_{ca(S,T)}(o) \overset{\text{Mov}}{\Rightarrow} m \notin r\prime$
   b) $r \mapsto o\prime = f_{ca(S\prime,T\prime)}(o) \overset{\text{Ass}}{\Rightarrow} m \notin ca(S\prime,T\prime)$ $\overset{\text{Mov}}{\Rightarrow} r\prime = f_{ca(S,T)}(f_{ca(S\prime,T\prime)}(o)) = f_{\text{merge}(ca(S,T),ca(S\prime,T\prime))}(o) \overset{\text{merge}}{\Rightarrow} m \notin r\prime$

The second case needs more explanation: If r has type S and points directly to o, S has no m (induction hypothesis). Then, m is not part of $ca(S,T)$ following CA-RCD. By applying this cast action to o, m is not callable on the resulting membrane.

If r points to a membrane, following the induction hypothesis m is not callable on this membrane. But the membrane results from applying a type cast action $ca(S\prime,T\prime)$ to the object o, so m was not a part of that type cast action. When merging this cast action with any other type cast action, m cannot be part of the result. Therefore it is also not callable through $r\prime$.

Instruction: `call ref m args res`
Let method *foo* accept an argument of type U and $S \prec: U$. We assume that *foo* is callable through some reference *ref* and the actual implementation accepts arguments of type T, with $U \prec: T$. Let $r\prime$ be the name of a parameter in *foo*. The instruction pushes a new frame onto the stack so that the stack will look like this: $[\dots, Var\prime = (r\prime, \dots), \dots] :: s :: t$.

- $m = (foo, opt, argT = (\dots, U, \dots), resT = (\dots))$
- $args = (\dots, r, \dots),\quad res = (\dots),\quad ref \mapsto obj \vee ref \mapsto f_A(obj)$
- $ref \mapsto obj \Rightarrow U \prec:_m T \Rightarrow ca(U,T) = \top$

Again, we look at the different cases:

1) $S \prec:_m U \wedge U \prec:_m T \Rightarrow S \prec:_m T$
   a) $r \mapsto o \overset{\text{Ass}}{\Rightarrow} m \notin S \Rightarrow m \notin T \overset{\text{Call}}{\Rightarrow} r\prime \mapsto f_\top(o) = o \Rightarrow m \notin r\prime$
   b) $r \mapsto f_{ca(S\prime,T\prime)}(o) \overset{\text{Ass}}{\Rightarrow} m \notin ca(S\prime,T\prime) \overset{\text{Call}}{\Rightarrow} r\prime \mapsto f_\top(f_{ca(S\prime,T\prime)}(o)) = f_{ca(S\prime,T\prime)}(o) \Rightarrow m \notin r\prime$
2) $S \not\prec:_m U \vee U \not\prec:_m T \Rightarrow S \not\prec:_m T$
   a) $r \mapsto o \overset{\text{Ass}}{\Rightarrow} m \notin S \Rightarrow m \notin ca(S,U) \Rightarrow m \notin f_{ca(S,U)}(o)$ $\Rightarrow m \notin f_{ca(U,T)}(f_{ca(S,U)}(o)) \overset{\text{Call}}{\Rightarrow} m \notin r\prime$
   b) $r \mapsto o\prime = f_{ca(S\prime,T\prime)}(o) \overset{\text{Ass}}{\Rightarrow} m \notin ca(S\prime,T\prime)$ $\overset{\text{Call}}{\Rightarrow} r\prime = f_{ca(U,T)}(f_{ca(S,U)}(f_{ca(S\prime,T\prime)}(o)))$ $\overset{\text{merge}}{\Rightarrow} m \notin r\prime$

The proof for method calls performed on a membrane follows directly from Call-Mem and merge.

Instruction: `inv x mth r`
The inv-instruction accepts only arguments of type *Any*. In order to pass r as an argument to *mth* it needs to be casted it to this special type. This always introduces a membrane $f_{ca(S,Any)}(r)$ allowing exactly the methods that are callable through r, because $S \not\prec:_m Any$. This membrane can then be assigned to a reference $r\prime$ of the methods formal parameter type T with just a type check ($Any \prec:_m T \wedge Any \not\prec:_c T$).

$$m \notin r \overset{\text{Inv-M}}{\Rightarrow} m \notin r\prime$$

Instruction: `ret`
Let *bar* be the current method with formal return type $(\dots, U, \dots)$ that was invoked through an interface $bar : X \mapsto (\dots, V, \dots)$. Let $s\prime :: s :: t$ be a snapshot of the system's state right before the execution of the return-instruction.

$$s\prime = [\; obj,\; bar,\; Var\prime = (\dots, r, \dots),\; ResT\prime = (\dots, V, \dots),\; Res\prime = (\dots, r\prime \dots),\; \bot \;]$$

Let $\Gamma \vdash r :: S \wedge \Gamma \vdash r\prime :: T$ and $S \prec: U$.

- $S \prec: U \Rightarrow ca(S,U) \neq \bot$
- $V \prec: T \Rightarrow ca(V,T) \neq \bot$
- $U <: V \Rightarrow ca(U,V) = \top$ (follows directly from CM-Arrow)

Again, look at the different cases:

1) $S \prec:_m U \wedge V \prec:_m T \Rightarrow S \prec:_m T$
   a) $r \mapsto o \overset{\text{Ass}}{\Rightarrow} m \notin S \Rightarrow m \notin T \Rightarrow m \notin r\prime$
   b) $r \mapsto f_{ca(S\prime,T\prime)}(o) \overset{\text{Ass}}{\Rightarrow} m \notin ca(S\prime,T\prime) \overset{\text{Ret}}{\Rightarrow} r\prime \mapsto f_{ca(S\prime,T\prime)}(o) \Rightarrow m \notin r\prime$
2) $S \not\prec:_m U \vee V \not\prec:_m T \Rightarrow S \not\prec:_m T$
   a) $r \mapsto o \overset{\text{Ass}}{\Rightarrow} m \notin S$
   $\Rightarrow m \notin \text{merge}(\text{merge}(ca(S,U), \top), ca(V,T))$
   $\overset{\text{Ret}}{\Rightarrow} r\prime \mapsto f_{\text{merge}(\text{merge}(ca(S,U),\top),ca(V,T))}(o)$
   $\Rightarrow m \notin r\prime$
   b) $r \mapsto f_{M=ca(S\prime,T\prime)}(o) \overset{\text{Ass}}{\Rightarrow} m \notin r$
   $\Rightarrow m \notin ca(S\prime,T\prime)$
   $\overset{\text{Ret}}{\Rightarrow} r\prime = f_Y(f_M(o))$ with
   $Y = \text{merge}(\text{merge}(ca(S,U), \top), ca(V,T))$
   $\Rightarrow r\prime = f_{\text{merge}(Y,M)}(o)$
   $\Rightarrow m \notin r\prime$

The proof for returning from a method called on a membrane follows directly from Ret-M and merge.

■

Implementation and Performance Evaluation of Eavesdropping Protection Method over MPTCP Using Data Scrambling and Path Dispersion

Toshihiko Kato1)2), Shihan Cheng1), Ryo Yamamoto1), Satoshi Ohzahata1) and Nobuo Suzuki2)
1) Graduate School of Informatics and Engineering
University of Electro-Communications
Tokyo, Japan
2) Adaptive Communications Research Laboratories
Advanced Telecommunication Research Institute International
Kyoto, Japan
kato@net.lab.uec.ac.jp, chengshihan@net.lab.uec.ac.jp, ryo_yamamotog@net.lab.uec.ac.jp,
ohzahata@net.lab.uec.ac.jp, nu-suzuki@atr.jp

***Abstract*—In order to utilize multiple communication interfaces installed mobile terminals, Multipath Transmission Control Protocol (MPTCP) has been introduced recently. It can establish an MPTCP connection that transmits data segments over the multiple interfaces, such as 4G and Wireless Local Area Network (WLAN), in parallel. However, it is possible that some interfaces are connected to untrusted networks and that data transferred over them is observed in an unauthorized way. In order to avoid this situation, we proposed a method to improve privacy against eavesdropping using the data dispersion by exploiting the multipath nature of MPTCP in our previous papers. The proposed method takes an approach that, if an attacker cannot observe the data on *every* path, he/she cannot observe the traffic on *any* path. The fundamental techniques of this method is a per-byte data scrambling and path dispersion. In this paper, we present the result of implementing the proposed method within the Linux operating system and its performance evaluation in more detail than our former papers.**

Keywords- Multipath TCP; Eavesdropping; Data Dispersion; Data Scrambling.

I. INTRODUCTION

This paper is an extension of our previous paper [1], which is presented in an IARIA conference.

Recent mobile terminals are equipped with multiple interfaces. For example, most smart phones have interfaces for 4G Long Term Evolution (LTE) and WLAN. In the next generation (5G) mobile network, it is expected that multiple communication paths provided by multiple network operators are commonly involved [2]. In this case, mobile terminals will have more than two interfaces.

However, the conventional TCP establishes a connection between single IP addresses at individual ends, and so it cannot utilize multiple interfaces in one end at the same time. In order to cope with this issue, MPTCP [3] is being introduced in several operating systems, such as Linux, Apple OS/iOS [4] and Android [5]. MPTCP is an extension of the conventional TCP. It combines multiple TCP flows into one data stream called an MPTCP connection, and provides the same programing interface with the socket interface. So, existing TCP applications can use MPTCP as if they were working over conventional TCP.

MPTCP is defined by three Request for Comments (RFC) documents by the Internet Engineering Task Force. RFC 6182 [6] outlines architecture guidelines. RFC 6824 [7] presents the details of extensions to support multipath operation, including the maintenance of an MPTCP connection and subflows (TCP connections associated with an MPTCP connection), and the data transfer over an MPTCP connection. RFC 6356 [8] presents a congestion control algorithm that couples the congestion control algorithms running on different subflows.

When a mobile terminal uses multiple paths, some of them may be unsafe such that an attacker is able to observe data over them in an unauthorized way. For example, a WLAN interface is connected to a public WLAN access point, data transferred over this WLAN may be disposed to other nodes connected to it. One way to prevent the eavesdropping is the Transport Layer Security (TLS). Although TLS can be applied to various applications including web access, e-mail, and ftp, however, it generally requires at least one end to maintain a public key certificate, and so it will not be used in some kind of communication, such as private server access and peer to peer communication.

As an alternative scheme, we proposed a method to improve confidentiality against eavesdropping by exploiting the multipath nature of MPTCP [9][10]. Even if an unsafe WLAN path is used, another path may be safe, such as LTE supported by a trusted network operator. So, the proposed method is based on an idea that, if an attacker cannot observe the data on *every* path, he/she cannot observe the traffic on *any* path [11]. In order to realize this idea, we adopted a byte based data scrambling for data segments sent over multiple subflows. This mixes up data to avoid its recognition through illegal monitoring over an unsafe path. Although there are some proposals to use multiple TCP connections to protect eavesdropping [12]-[15], all of them depend on the encryption techniques. The proposed method is dependent on the exclusive OR (XOR) calculation that is much lighter in terms of processing overhead.

In our previous paper [1] that is the origin of this paper, we showed how to implement the proposed method over the Linux operating system. We used the kernel debugging mechanism called *JProbe*, in order to avoid the modification of the Linux kernel as much as possible. The previous paper also showed the results of implementation focusing on how the proposed method works over off-the shelf personal computers and access point, but the descriptions on performance evaluation was limited.

In this paper, we describe the proposed method and its implementation in more detail. We also show another behavior of the proposed method and the results of performance evaluation in detail.

The rest of this paper is organized as follows. Section II explains the overview and the security issue of MPTCP [10]. Section III describes the proposed method. Section IV shows how to implement the proposed method within the MPTCP software in the Linux operating system. Section V gives the behavior of the proposed method and the results of the performance evaluation. In the end, Section VI concludes this paper.

II. Overview and Security Issues of MPTCP

A. MPTCP connections and subflows

As described in Figure 1, the MPTCP module is located on top of TCP. As described above, MPTCP is designed so that the conventional applications do not need to care about the existence of MPTCP. MPTCP establishes an *MPTCP connection* associated with two or more regular TCP connections called *subflows*. The management and data transfer over an MPTCP connection is done by newly introduced TCP options for MPTCP operation.

Figure 2 shows an example of MPTCP connection establishment where host A with two network interfaces invokes this sequence for host B with one network interface. In the beginning, host A sends a SYN segment to host B with a *Multipath Capable (MP_CAPABLE)* TCP option. This option indicates that an initiator supports the MPTCP functions and requests to use them in this TCP connection. It contains host A's *Key* (64 bits) used by this MPTCP connection. Then, host B replies a SYN+ACK segment with MP_CAPABLE option with host B's Key. This reply means that host B accepts the use of MPTCP functions. In the end, host A sends an ACK segment with MP_CAPABLE option including both A's and B's Keys. Through this three-way handshake procedure, the first subflow and the MPTCP connection are established. Here, it should be mentioned that these "Keys" are not keys in a cryptographic sense. They are a sort of random numbers assigned for individual MPTCP connections. As described below, they are used for generating the Hash-based Message Authentication Code (HMAC), but MPTCP does not provide any mechanisms to protect them from attackers' accessing while transfer.

Next, host A tries to establish the second subflow through another network interface. In the first SYN segment in this try, another TCP option called a *Join Connection (MP_JOIN)* option is used. An MP_JOIN option contains the receiver's *Token* (32 bits) and the sender's *Nonce* (random number, 32 bit). A Token is an information to identify the MPTCP connection to be joined. It is obtained by taking the most significant 32 bits from the SHA-1 hash value for the receiver's Key (host B's Key in this example). Then, host B replies a SYN+ACK segment with MP_JOIN option. In this case, MP_JOIN option contains the random number of host B and the most significant 64 bits of the HMAC value. An HMAC value is calculated for the nonces generated by hosts A and B using the Keys of A and B. In the third ACK segment, host A sends an MP_JOIN option containing host A's full HMAC value (160 bits). In the end, host B acknowledges the third ACK segment. Using these sequence, the newly established subflow is associated with the MPTCP connection.

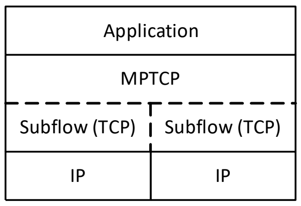


Figure 1. Layer structure of MPTCP.

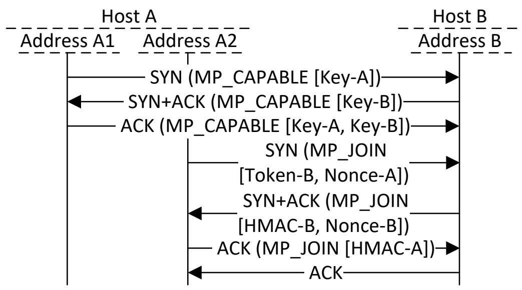

Host A
Address A1
Address A2
Host B
Address B
SYN (MP_CAPABLE [Key-A])
SYN+ACK (MP_CAPABLE [Key-B])
ACK (MP_CAPABLE [Key-A, Key-B])
SYN (MP_JOIN [Token-B, Nonce-A])
SYN+ACK (MP_JOIN [HMAC-B, Nonce-B])
ACK (MP_JOIN [HMAC-A])
ACK

Figure 2. Example of MPTCP connection establishment.

B. Data transfer

An MPTCP implementation will take one input data stream from an application, and split it into one or more subflows, with sufficient control information to allow it to be reassembled and delivered to the receiver side application reliably and in order. The MPTCP connection maintains the *data sequence number* independent of the subflow level sequence numbers. The data and ACK segments may contain a *Data Sequence Signal (DSS)* option depicted in Figure 3.

The data sequence number and data ACK is 4 or 8 byte long, depending on the flags in the option. The number is assigned on a byte-by-byte basis similarly with the TCP sequence number. The value of data sequence number is the number assigned to the first byte conveyed in that TCP segment. The data sequence number, subflow sequence number (relative value) and data-level length define the mapping between the MPTCP connection level and the subflow level. The data ACK is analogous to the behavior of the standard TCP cumulative ACK. It specifies the next data sequence number a receiver expects to receive.

Kind (= 30)	Length	Subtype (= 2)	Flags
Data ACK (4 or 8 octets, depending on flags)			
Data sequence number (4 or 8 octets, depending on flags)			
Subflow sequence number (4 octets)			
Data-level length (2 octets)		Checksum (2 octets)	

Figure 3. Data Sequence Signal option.

C. *Eavesdropping protection over multiple TCP connections*

As we mentioned in Section I, there are several proposals on the data dispersion over multiple paths. Yang and Papavassiliou [12] showed a method to analyze the security performance when a virtual connection takes multiple disjoint paths to the destination, and proposed a traffic dispersion scheme to minimize the information leakage when some of the intermediate routers are attacked. Nacher et al. [13] tried to determine the optimal trade-off between traffic dispersion and TCP performance over mobile ad-hoc networks to reduce the chances of successful eavesdropping while maintaining acceptable throughput. These two studies use multiple TCP connections by their own coordination methods instead of MPTCP. Gurtov and Polishchuk [14] used host identity protocol (HIP), which locates between IP and TCP to provide multiple paths, and proposed how to spread traffic over them. Apiecionek et al. [15] proposed a way to use MPTCP for more secure data transfer. After data are encrypted, they are divided into blocks, mixed in the predetermined random sequence, and then transferred through multiple MPTCP subflows. A receiver rearranges received blocks in right order and decrypts them.

All of those proposals aim at just spreading data packets over multiple paths, and do not consider the coordination over multiple paths. If the transferred data are encrypted before dispersion, it can be said that they are coordinated by the encryption procedure, but the coordination is not realized by the dispersion schemes. In contrast with them, our proposal adopts an approach to improve privacy by coordinating data over multiple paths through data scrambling not encryption.

III. Proposed Method

A. *Motivation*

The first point we considered in designing our proposal was utilize multiple paths supported by MPTCP. So we picked up the secret sharing method [16], which produces some number of pieces from data in a way that a specific number of pieces are required for reconstructing the original data. By using the secret sharing, it is possible to divide data into multiple subflows. However, in this approach, it is required to duplicate the original data or at least increase the amount of sending data. Besides, this approach requires some cryptographic calculation that uses a lot of CPU power.

Next approach we considered is the network coding approach [17], where a packet is XORed with the following packet and the first packet and the XOR result are sent via different paths [18]. This approach does not require any cryptographic calculation and so the processing overhead is low. However, if the packet length is different, unnecessary padding may be necessary.

The third approach is applying the mode of operation, such as Cipher Block Chaining (CBC) and Output Feedback (OFB), used in block ciphering [19]. The block cipher defines only how to encrypt or decrypt a fixed length bits (block). The mode of operation defines how to apply this operation to data longer than a block. CBR and OFB introduce a chaining between blocks such that a block is combined with the preceding block by XOR calculation. The application of the mode of operation without any encryption to data dispersion is considered as a block level data scrambling. So, if the length of packet is not integral multiple of block length, unnecessary padding will be required again.

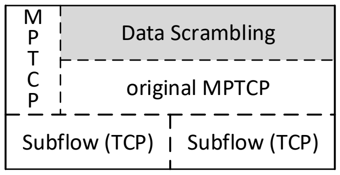
Based on these considerations, we picked up a byte level scrambling which is described in the following subsection.

B. *Detailed procedure*

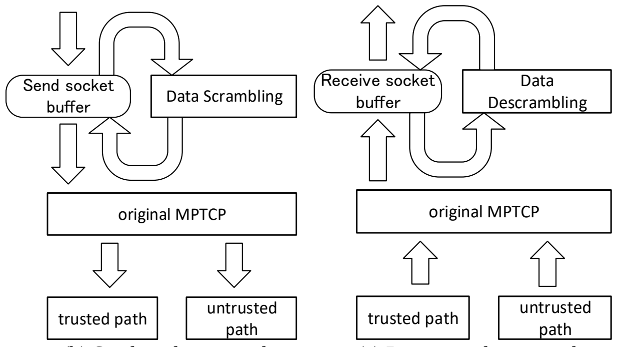
Figure 4 shows the overview of the proposed method. As shown in Figure 4(a), we introduce a data scrambling function within MPTCP and on top of the original MPTCP. When an MPTCP communication is started, the use of data scrambling is negotiated. It may be done using a flag bit in MP_CAPABLE TCP option.

When an application sends data, it is stored in the send socket buffer in the beginning. The proposed method scrambles the data by calculating XOR of a byte with its preceding 64 bytes in the sending byte stream. Then, the scrambled data is sent through multiple subflows associated with the MPTCP connection. Since some data segments are transmitted through trusted subflows, an attacker monitoring only a part of data segments cannot obtain all of sent data and so cannot descramble any of them. When receiving data segments, they are reordered in the receive socket buffer by MPTCP. The proposed method descrambles them in a byte-by-byte basis just before an application reads the received data.

Figure 5 shows the details of data scrambling. In order to realize this scrambling, the data scrambling module maintains the *send scrambling buffer*, whose length is 64 bytes. It is a shift buffer and its initial value is HMAC of the key of this side, with higher bytes set to zero. The key used here is one of the MPTCP parameters, exchanged in the first stage of MPTCP connection establishment. When a data comes from an application, each byte (b_i in the figure) is XORed with the result of XOR of all the bytes in the send scrambling buffer.



(a) Layer structure of proposed method



(b) Sending data procedure (c) Receiving data procedure

Figure 4. Overview of proposed method [9].

The obtained byte (B_i) is the corresponding sending byte. After calculating the sending byte, the original byte (b_i) is added to the send scramble buffer, forcing out the oldest (highest) byte from the buffer. The send scrambling buffer holds recent 64 original bytes given from an application. By using 64 byte buffer, the access to the original data is protected even if there are well-known byte patterns (up to 63 bytes) in application protocol data.

Figure 6 shows the details of data descrambling, which is similar with data scrambling. The data scrambling module also maintains the receive scramble buffer whose length is 64 bytes. Its initial value is HMAC of the key of the remote side. When an in-sequence data is stored in the receive socket buffer, a byte (B_i that is scrambled) is applied to XOR calculation with the XOR result of all the bytes in the receive scramble buffer. The result is the descrambled byte (b_i), which is added to the receive scramble buffer.

By using the byte-wise scrambling and descrambling, the proposed method does not increase the length of exchanged data at all. The separate send and receive control enables two way data exchanges to be handled independently. Moreover, the proposed method introduces only a few modification to the original MPTCP.

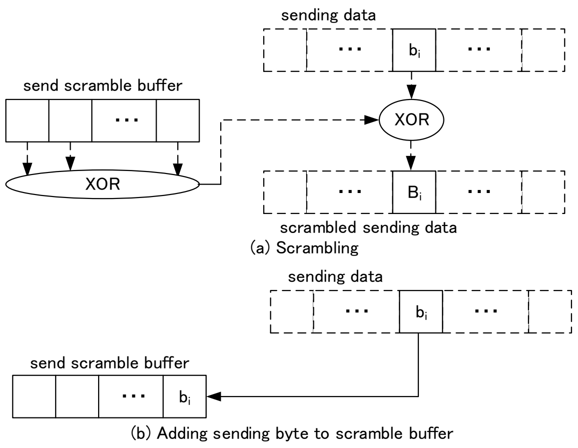


Figure 5. Processing of data scrambling [9].

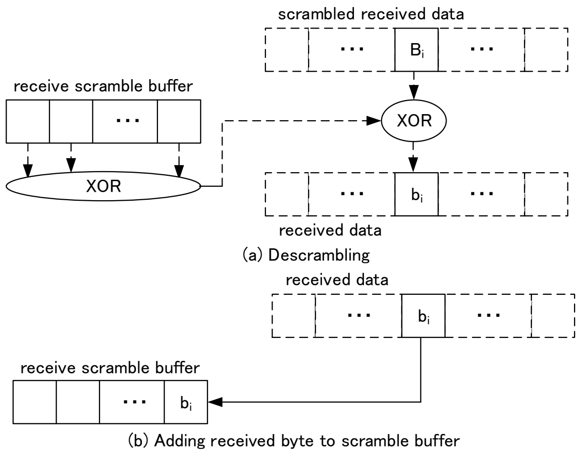


Figure 6. Processing of data descrambling [9].

IV. Implementation

A. Use of Kernel Probes

Since MPTCP is implemented inside the Linux operating system, the proposed method also needs to be realized by modifying operating system kernel. However, modifying an operating system kernel is a hard task, and so we decided to use a debugging mechanism for the Linux kernel, called kernel probes [20].

Among kernel probes methods, we use a way called "JProbe" [21]. JProbe is used to get access to a kernel function's arguments at runtime. It introduces a JProbe handler with the same prototype as that of the function whose arguments are to be accessed. When the probed function is executed, the control is first transferred to the user-defined JProbe handler. After the user-defined handler returns, the control is transferred to the original function [20].

In order to make this mechanism work, a user needs to prepare the following;

- registering the entry by `struct jprobe` and
- defining the init and exit modules by functions `register_jprobe()` and `unregister_jprobe()` [21].

In the Linux kernel, function `tcp_sendmsg()` is called when an application sends data to MPTPCP (actually TCP, too) [22]. As stated in Section II, the scrambling will be done at the beginning of this function. So, we define a JProbe handler for function `tcp_sendmsg()` for scrambling data to be transferred.

In order for an application to read received data, it calls function `tcp_recvmsg()` in MPTCP. In contrast to data scrambling, the descrambling procedure needs to be done at the end of this function. So, we introduce a dummy kernel function and export its symbol just before the returning points of function `tcp_recvmsg()`. We then define a JProbe handler for descrambling in this dummy function.

By adopting this approach, we can program and debug scrambling/descrambling independently of the Linux kernel itself.

B. Modification of Linux opeating system

We modified the source code of the Linux operating system in the following way. We believe that this is a very slight modification that requires to us to rebuild the kernel only once.

- *Introduce a dummy function in* `tcp_recvmsg()`.

As described above, we defined a dummy function named `dummy_recvmsg()`. It is defined in the source file "`net/ipv4/tcp.c`" as shown in Figure 7. It is a function just returning and inserted before function `tcp_recvmsg()` releases the socket control. Since this function is very simple, "`noinline`" indication pragma needs to be specified. The prototype declaration is done in the source file "`include/net/tcp.h`".

- *Maintain control variables within socket data structure.*

In order to perform the scrambling/descrambling, the control variables, such as a scramble buffer, need to be installed within the Linux kernel. The TCP software in the kernel uses a socket data structure to maintain internal control data on an individual TCP / MPTCP connection [22]. This is controlled by the following variable, as shown in Figure 4.

```
    struct tcp_sock *tp = tcp_sk(sk);
```

This structure includes the MPTCP related parameters, such as keys and tokens. The parameters are packed in an element given below.

```
            struct mptcp_cb *mpcb;
```

So, we added the control variables for data scrambling in this data structure. Figure 8 shows the control variables. The details of those variables are given in the following.

- `sScrBuf[64]` and `rScrBuf[64]`: the send and receive scramble buffers, used as ring buffers.
- `sXor` and `rXor`: the results of calculation of XOR for all the bytes in the send and receive scramble buffers.
- `sIndex` and `rIndex`: the index of the last (newest) element in `sScrBuf[64]` and `rScrBuf[64]`.
- `sNotFirst` and `rNotFirst`: the flags indicating whether the scrambling and descrambling are invoked for the first time in the MPTCP connection, or not.

C. Implementation of scrambling

(1) Framework of JProbe handler

Figure 9 shows the framework of JProbe hander defined for `tcp_sendmsg()`. Function `jtcp_sendmsg()` is a main body of the JProbe hander. The arguments need to be exactly the same with the hooked kernel function `tcp_sendmsg()`, and it calls `jprobe_return()` just before its returning. Data structure `struct jprobe mptcp_jprobe` specifies its details.

Function `mptcp_scramble_init()` is the initialization function invoked when the relevant kernel module is inserted. In the beginning, it confirm that the hander has the same prototype with the hooked function. Then it defines the entry point and registers the JProbe handler. Function `mptcp_scramble_exit()` is called when the relevant kernel module is removed. It removes the entry point and unregisters the hander from the kernel.

```
int tcp_recvmsg(struct sock *sk, struct msghdr *msg,
    size_t len, int nonblock,int flags, int *addr_len) {
  struct tcp_sock *tp = tcp_sk(sk);
  . . . .
  dummy_recvmsg(sk, msg, len, nonblack, flags, addr_len, copied);
  release_sock(sk);
  return copied;
  . . . .
} // dummy_recvmsg() inserted
EXPORT_SYMBOL(tcp_recvmsg);

void noinline dummy_recvmsg(struct sock *sk, struct msghdr *msg,
     size_t len, int nonblock, int flags, int *addr_len,
     int copied)
{
  return;
} // Defining dummy_recvmsg()
EXPORT_SYMBOL(dummy_recvmsg);
```

Figure 7. Dummy function in `tcp_recvmsg()`.

```
struct mptcp_cb {
 . . . .
  unsigned char sScrBuf[64], rScrBuf[64];
  unsigned char sXor, rXor;
  int sIndex, rIndex, sNotFirst, rNotFirst;
};
```

Figure 8. Control variables for data scrambling/descrambling.

```
static const char procname[] = "mptcp_scramble"
int jtcp_sendmsg(struct sock *sk, struct msghdr *msg,
       size_t size) {
  struct tcp_sock *tp = tcp_sk(sk);
  . . .
  jprobe_return();
  return 0;
} // (i) JProbe handler

static struct jprobe mptcp_jprobe = {
  .kp = {.symbol_name = "tcp_sendmsg",},
        .entry = jtcp_sendmsg,
}; // (ii) Register entry

static __init int mptcp_scramble_init(void) {
  int ret = -ENOMEM;
  BUILD_BUG_ON(__same_type(tcp_sendmsg, jtcp_sendmsg) == 0);
  if(!proc_create(procname, S_IRUSR, init_net.proc_net, 0))
    return ret;
  ret = register_jprobe(&mptcp_jprobe);
  if (ret) {
    remove_proc_entry(procname, init_net.proc._net);
    retrun ret;
  }
  return 0;
}  // (iii) Init function
module_init(mptcp_scramble_init);

static __exit void mptcp_scramble_exit(void) {
  remove_proc_entry(procname, init_net.proc._net);
  unregister_jprobe(&mptcp_jprobe);
}  // (iv) Exit function
module_exit(mptcp_scramble_exit);
```

Figure 9. JProbe hander definition for `tcp_sendmsg()`.

(2) Flowchart of data scrambling

The data scrambling procedure is implemented in `jtcp_sendmsg()`. Figure 10 shows the flowchart for this procedure. When `jtcp_sendmsg()` is called, it is checked whether this function is invoked for the first time or not. If it is the first invocation over a specific MPTCP connection, `sScrBuf[]` is initialized to the value of the local key maintained in the `struct mptcp_cb` structure. Then, XOR of all the bytes in `sScrBuf[]` is calculated and saved in `sXor`, and `sIndex` is set to 63.

The argument containing data (`msg`) is a list of data blocks, and so individual blocks are handled sequentially. For each data block, a byte-by-byte basis calculation is performed in the following way. First, the XOR of the focused byte and `sXor` is saved in temporal variable `x`. Then, `sIndex` is advanced by one under modulo 64. Thirdly, the XOR of `sXor`, `sScrBuf[sIndex]` and the original byte are calculated and saved in sXor. It should be noted that the value in `sScrBuf[sIndex]` at this stage is the oldest value in the send scramble buffer. Fourthly, the original byte is stored in `sScrBuf[sIndex]`,which means that the send scramble buffer is updated. At last, the byte in the message block is replaced by the value of `x`.

D. Implementation of descrambling

The data descrambling is implemented similarly with scrambling. We developed the JProbe handler for function `dummy_recvmsg()` in the same way with the approach given in Figure 9. The flowchart of descrambling procedure is shown in Figure 11. This is similar with the flowchart shown in Figure 10. In the first part of the flowchart, it should

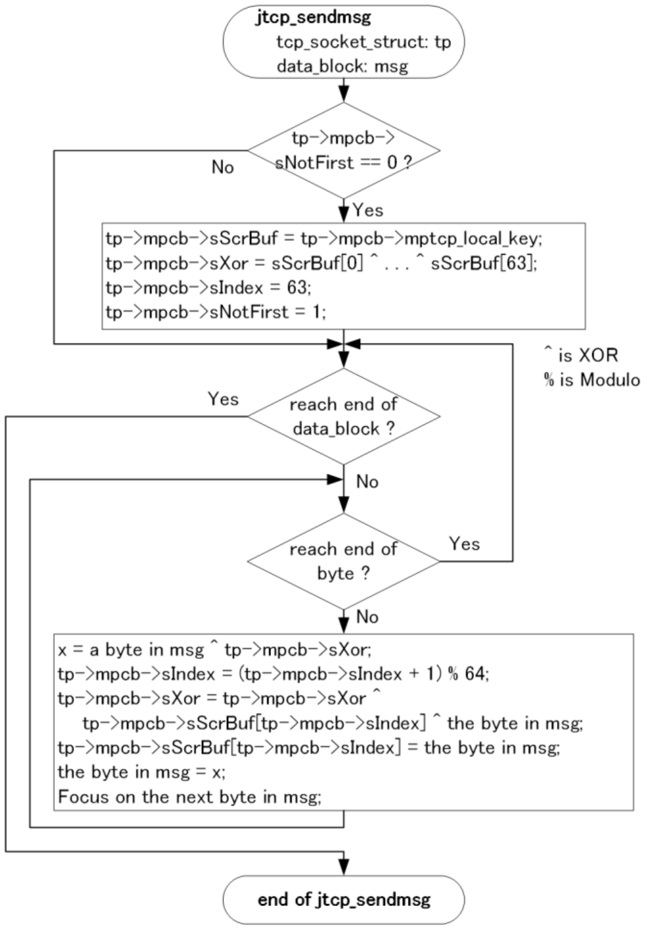

jtcp_sendmsg
tcp_socket_struct: tp
data_block: msg
tp->mpcb->sNotFirst == 0 ?
No
Yes
tp->mpcb->sScrBuf = tp->mpcb->mptcp_local_key;
tp->mpcb->sXor = sScrBuf[0] ^ . . . ^ sScrBuf[63];
tp->mpcb->sIndex = 63;
tp->mpcb->sNotFirst = 1;
^ is XOR
% is Modulo
reach end of data_block ?
Yes
No
reach end of byte ?
Yes
No
x = a byte in msg ^ tp->mpcb->sXor;
tp->mpcb->sIndex = (tp->mpcb->sIndex + 1) % 64;
tp->mpcb->sXor = tp->mpcb->sXor ^ tp->mpcb->sScrBuf[tp->mpcb->sIndex] ^ the byte in msg;
tp->mpcb->sScrBuf[tp->mpcb->sIndex] = the byte in msg;
the byte in msg = x;
Focus on the next byte in msg;
end of jtcp_sendmsg

Figure 10. Flowchart of data scrambling.

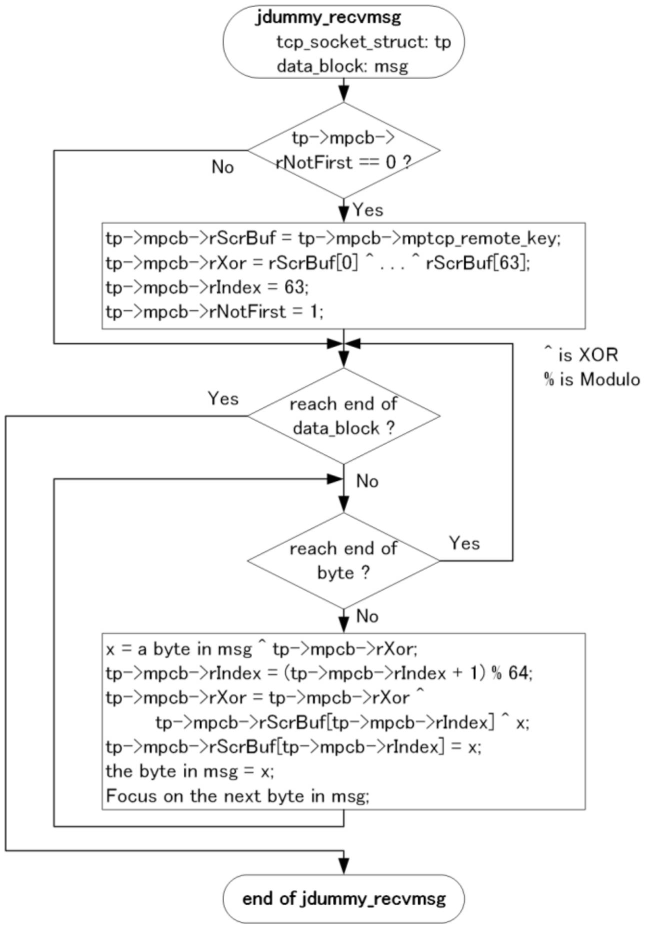

jdummy_recvmsg
tcp_socket_struct: tp
data_block: msg
tp->mpcb->rNotFirst == 0 ?
No
Yes
tp->mpcb->rScrBuf = tp->mpcb->mptcp_remote_key;
tp->mpcb->rXor = rScrBuf[0] ^ . . . ^ rScrBuf[63];
tp->mpcb->rIndex = 63;
tp->mpcb->rNotFirst = 1;
^ is XOR
% is Modulo
reach end of data_block ?
Yes
No
reach end of byte ?
Yes
No
x = a byte in msg ^ tp->mpcb->rXor;
tp->mpcb->rIndex = (tp->mpcb->rIndex + 1) % 64;
tp->mpcb->rXor = tp->mpcb->rXor ^ tp->mpcb->rScrBuf[tp->mpcb->rIndex] ^ x;
tp->mpcb->rScrBuf[tp->mpcb->rIndex] = x;
the byte in msg = x;
Focus on the next byte in msg;
end of jdummy_recvmsg

Figure 11. Flowchart of data descrambling.

be noted that `rScrBuf[]` is set to the remote key, which is the local key in the sender side. In this case, the data block is a descrambled data. Therefore, in the byte-by-byte basis part, the original value (`x` in the figure) is used to calculate `rXor` and is stored in `rSrcBuf[rIndex]`.

V. EXPERIMENT AND PERFORMANCE EVALUATION

A. *Experimental settings*

We implemented the proposed method over the Linux operating system (Ubuntu 16.04 LTS/14.04 LTS with MPTCP support). We evaluated it in the experimental configuration shown in Figure 12. Two note PCs are used as a client and a server. They are connected with each other via a hub/access point using Ethernet and WLAN. Ethernet is 100base-T and WLAN is IEEE 802.11g with 2.4 GHz. The client uses both Ethernet and WLAN, and the server uses only Ethernet. The WLAN interface does not use any encryption. We suppose that the Ethernet link is a trusted network and the WLAN link without any encryption is an untrusted network. Table I shows the specification of nodes. It should be noted that the model for the client and server nodes is a little old and the processing power of CPU is low. The model of the access point is Buffalo Air Station G54. The proposed method is implemented in the Linux operating system running over the client and server nodes. In the attacker node, Wireshark (and tshark) is executed in order to monitor data transferred over WLAN.

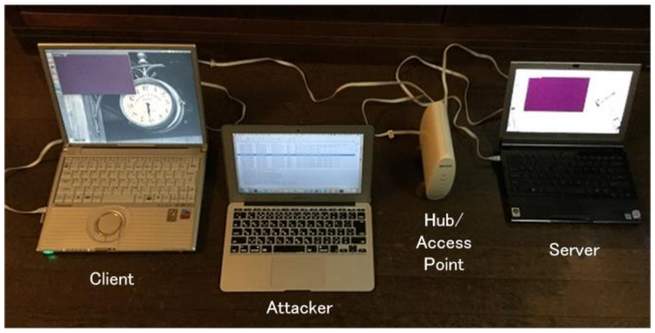


Figure 12. Experiment configuration.

TABLE I. SPECIFICATION OF NODES USED IN EXPERIMENT.

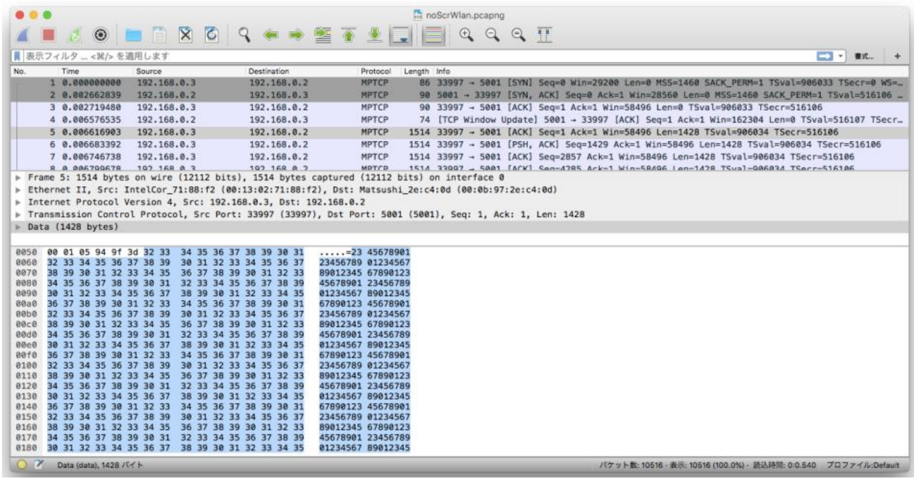
Node	Model	CPU	Operating system
Client	Panasonic Let's note CF-Y4	Intel Pentium M 1.50 GHz	ubuntu 16.04 LTS
Server	Sony Vaio PCG-4P7N	Intel Core 2 Duo U7700 1.33 GHz x 2	ubuntu 14.04 LTS
Attacker	Apple MacBook Air	Intel Core i5 1.7 GHz	macOS High Sierra

The network setting is as follows.

- Since the access point works as a bridge, the client and the server are connected to the same subnetwork, 192.168.0.0/24.
- The Ethernet and WLAN interfaces in the client are assigned with IP addresses 192.160.0.1 and 192.168.0.3, respectively. The Ethernet interface in the server is assigned with IP address 192.168.0.2. The ESSID of the WLAN is "MPTCP-AP."
- In order to use two interfaces at the client, the IP routing tables are set for individual interfaces, by use of the ip command in the following way (for the Ethernet interface `enp4s1`).
  - ➢ `ip rule add from 192.168.0.1 table 1`
  - ➢ `ip route add 192.168.0.0/24 dev enp4s1 scope link table 1`
- The JProbe handlers for `jtcp_sendmsg()` and `jdummy_recvmsg()` are built as kernel modules. They are inserted and removed using `insmod` and `rmmod` Linux commands without rebooting the system.
- In the experiment, we used `iperf` for sending data from the client to the server, using Ethernet and WLAN. In another evaluation, we used a simple file transfer, which we implemented, where a specified file is transferred from the server to the client.
- In the attacker node, the Wireshark network analyzer is invoked for monitoring a WLAN interface with both the promiscuous mode the monitor mode set to effective. When we use tshark at the attacker node, which is a command line interface version of Wireshark, the "`-I`" option is used for capturing all WLAN frames and the "`-Y`" option is used for applying a display filter defined similarly with Wireshark.

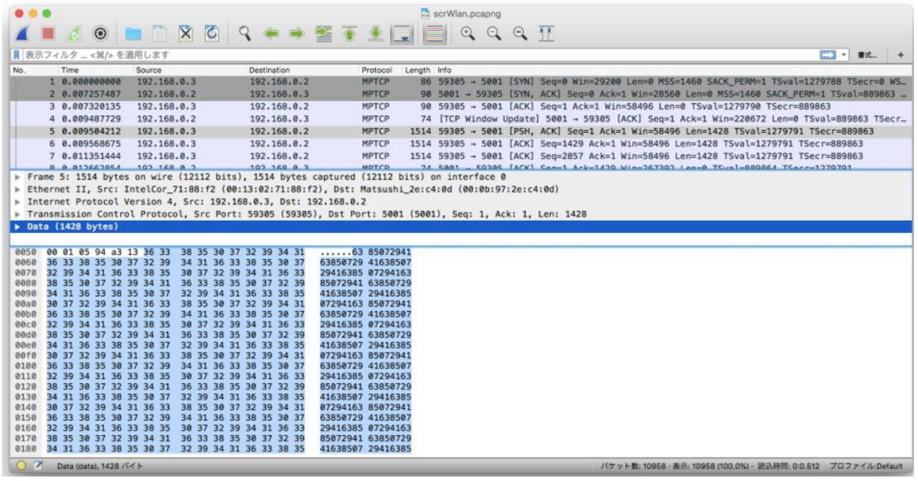
B. *Behaviors of proposed method*

Figure 13 shows a result of the attacker's monitoring of iperf communication over WLAN in the conventional communication. In the iperf communication, an ASCII digit sequence "0123456789" is sent repeatedly. If the attacker can monitor the WLAN, the content is disposed as shown in this figure. Figure 14 shows a monitoring result by the attacker over the WLAN link when the data scrambling is performed.



```
No.  Time         Source        Destination   Protocol Length Info
  1 0.000000000   192.168.0.3   192.168.0.2   MPTCP      86 33997 → 5001 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=906033 TSecr=0 WS=…
  2 0.002662839   192.168.0.2   192.168.0.3   MPTCP      90 5001 → 33997 [SYN, ACK] Seq=0 Ack=1 Win=28560 Len=0 MSS=1460 SACK_PERM=1 TSval=516106 …
  3 0.002719480   192.168.0.3   192.168.0.2   MPTCP      90 33997 → 5001 [ACK] Seq=1 Ack=1 Win=58496 Len=0 TSval=906033 TSecr=516106
  4 0.006576535   192.168.0.2   192.168.0.3   MPTCP      74 [TCP Window Update] 5001 → 33997 [ACK] Seq=1 Ack=1 Win=162304 Len=0 TSval=516107 TSecr…
  5 0.006616903   192.168.0.3   192.168.0.2   MPTCP    1514 33997 → 5001 [ACK] Seq=1 Ack=1 Win=58496 Len=1428 TSval=906034 TSecr=516106
  6 0.006683392   192.168.0.3   192.168.0.2   MPTCP    1514 33997 → 5001 [PSH, ACK] Seq=1429 Ack=1 Win=58496 Len=1428 TSval=906034 TSecr=516106
  7 0.006746738   192.168.0.3   192.168.0.2   MPTCP    1514 33997 → 5001 [ACK] Seq=2857 Ack=1 Win=58496 Len=1428 TSval=906034 TSecr=516106
  8 0.006799678   192.168.0.3   192.168.0.2   MPTCP    1514 33997 → 5001 [ACK] Seq=4285 Ack=1 Win=58496 Len=1428 TSval=906034 TSecr=516106
▶ Frame 5: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
▶ Ethernet II, Src: IntelCor_71:88:f2 (00:13:02:71:88:f2), Dst: Matsushi_2e:c4:0d (00:0b:97:2e:c4:0d)
▶ Internet Protocol Version 4, Src: 192.168.0.3, Dst: 192.168.0.2
▶ Transmission Control Protocol, Src Port: 33997 (33997), Dst Port: 5001 (5001), Seq: 1, Ack: 1, Len: 1428
▶ Data (1428 bytes)

0050  00 01 05 94 9f 3d 32 33  34 35 36 37 38 39 30 31   .....=23 45678901
0060  32 33 34 35 36 37 38 39  30 31 32 33 34 35 36 37   23456789 01234567
0070  38 39 30 31 32 33 34 35  36 37 38 39 30 31 32 33   89012345 67890123
0080  34 35 36 37 38 39 30 31  32 33 34 35 36 37 38 39   45678901 23456789
0090  30 31 32 33 34 35 36 37  38 39 30 31 32 33 34 35   01234567 89012345
00a0  36 37 38 39 30 31 32 33  34 35 36 37 38 39 30 31   67890123 45678901
00b0  32 33 34 35 36 37 38 39  30 31 32 33 34 35 36 37   23456789 01234567
00c0  38 39 30 31 32 33 34 35  36 37 38 39 30 31 32 33   89012345 67890123
00d0  34 35 36 37 38 39 30 31  32 33 34 35 36 37 38 39   45678901 23456789
00e0  30 31 32 33 34 35 36 37  38 39 30 31 32 33 34 35   01234567 89012345
00f0  36 37 38 39 30 31 32 33  34 35 36 37 38 39 30 31   67890123 45678901
0100  32 33 34 35 36 37 38 39  30 31 32 33 34 35 36 37   23456789 01234567
0110  38 39 30 31 32 33 34 35  36 37 38 39 30 31 32 33   89012345 67890123
0120  34 35 36 37 38 39 30 31  32 33 34 35 36 37 38 39   45678901 23456789
0130  30 31 32 33 34 35 36 37  38 39 30 31 32 33 34 35   01234567 89012345
0140  36 37 38 39 30 31 32 33  34 35 36 37 38 39 30 31   67890123 45678901
0150  32 33 34 35 36 37 38 39  30 31 32 33 34 35 36 37   23456789 01234567
0160  38 39 30 31 32 33 34 35  36 37 38 39 30 31 32 33   89012345 67890123
0170  34 35 36 37 38 39 30 31  32 33 34 35 36 37 38 39   45678901 23456789
0180  30 31 32 33 34 35 36 37  38 39 30 31 32 33 34 35   01234567 89012345

Data (data), 1428 バイト          パケット数: 10516 · 表示: 10516 (100.0%) · 読込時間: 0:0.540   プロファイル:Default
```

Figure 13. Capturing result of iperf when no scrambling is performed.



```
No.  Time         Source        Destination   Protocol Length Info
  1 0.000000000   192.168.0.3   192.168.0.2   MPTCP      86 59305 → 5001 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1279788 TSecr=0 WS…
  2 0.007257487   192.168.0.2   192.168.0.3   MPTCP      90 5001 → 59305 [SYN, ACK] Seq=0 Ack=1 Win=28560 Len=0 MSS=1460 SACK_PERM=1 TSval=889863 …
  3 0.007320135   192.168.0.3   192.168.0.2   MPTCP      90 59305 → 5001 [ACK] Seq=1 Ack=1 Win=58496 Len=0 TSval=1279790 TSecr=889863
  4 0.009487729   192.168.0.2   192.168.0.3   MPTCP      74 [TCP Window Update] 5001 → 59305 [ACK] Seq=1 Ack=1 Win=220672 Len=0 TSval=889863 TSecr…
  5 0.009504212   192.168.0.3   192.168.0.2   MPTCP    1514 59305 → 5001 [PSH, ACK] Seq=1 Ack=1 Win=58496 Len=1428 TSval=1279791 TSecr=889863
  6 0.009568675   192.168.0.3   192.168.0.2   MPTCP    1514 59305 → 5001 [ACK] Seq=1429 Ack=1 Win=58496 Len=1428 TSval=1279791 TSecr=889863
  7 0.011351444   192.168.0.3   192.168.0.2   MPTCP    1514 59305 → 5001 [ACK] Seq=2857 Ack=1 Win=58496 Len=1428 TSval=1279791 TSecr=889863
  8 0.012662854   192.168.0.2   192.168.0.3   MPTCP      74 5001 → 59305 [ACK] Seq=1 Ack=1429 Win=267392 Len=0 TSval=889864 TSecr=1279791
▶ Frame 5: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
▶ Ethernet II, Src: IntelCor_71:88:f2 (00:13:02:71:88:f2), Dst: Matsushi_2e:c4:0d (00:0b:97:2e:c4:0d)
▶ Internet Protocol Version 4, Src: 192.168.0.3, Dst: 192.168.0.2
▶ Transmission Control Protocol, Src Port: 59305 (59305), Dst Port: 5001 (5001), Seq: 1, Ack: 1, Len: 1428
▶ Data (1428 bytes)

0050  00 01 05 94 a3 13 36 33  38 35 30 37 32 39 34 31   ......63 85072941
0060  36 33 38 35 30 37 32 39  34 31 36 33 38 35 30 37   63850729 41638507
0070  32 39 34 31 36 33 38 35  30 37 32 39 34 31 36 33   29416385 07294163
0080  38 35 30 37 32 39 34 31  36 33 38 35 30 37 32 39   85072941 63850729
0090  34 31 36 33 38 35 30 37  32 39 34 31 36 33 38 35   41638507 29416385
00a0  30 37 32 39 34 31 36 33  38 35 30 37 32 39 34 31   07294163 85072941
00b0  36 33 38 35 30 37 32 39  34 31 36 33 38 35 30 37   63850729 41638507
00c0  32 39 34 31 36 33 38 35  30 37 32 39 34 31 36 33   29416385 07294163
00d0  38 35 30 37 32 39 34 31  36 33 38 35 30 37 32 39   85072941 63850729
00e0  34 31 36 33 38 35 30 37  32 39 34 31 36 33 38 35   41638507 29416385
00f0  30 37 32 39 34 31 36 33  38 35 30 37 32 39 34 31   07294163 85072941
0100  36 33 38 35 30 37 32 39  34 31 36 33 38 35 30 37   63850729 41638507
0110  32 39 34 31 36 33 38 35  30 37 32 39 34 31 36 33   29416385 07294163
0120  38 35 30 37 32 39 34 31  36 33 38 35 30 37 32 39   85072941 63850729
0130  34 31 36 33 38 35 30 37  32 39 34 31 36 33 38 35   41638507 29416385
0140  30 37 32 39 34 31 36 33  38 35 30 37 32 39 34 31   07294163 85072941
0150  36 33 38 35 30 37 32 39  34 31 36 33 38 35 30 37   63850729 41638507
0160  32 39 34 31 36 33 38 35  30 37 32 39 34 31 36 33   29416385 07294163
0170  38 35 30 37 32 39 34 31  36 33 38 35 30 37 32 39   85072941 63850729
0180  34 31 36 33 38 35 30 37  32 39 34 31 36 33 38 35   41638507 29416385

Data (data), 1428 バイト          パケット数: 10958 · 表示: 10958 (100.0%) · 読込時間: 0:0.512   プロファイル:Default
```

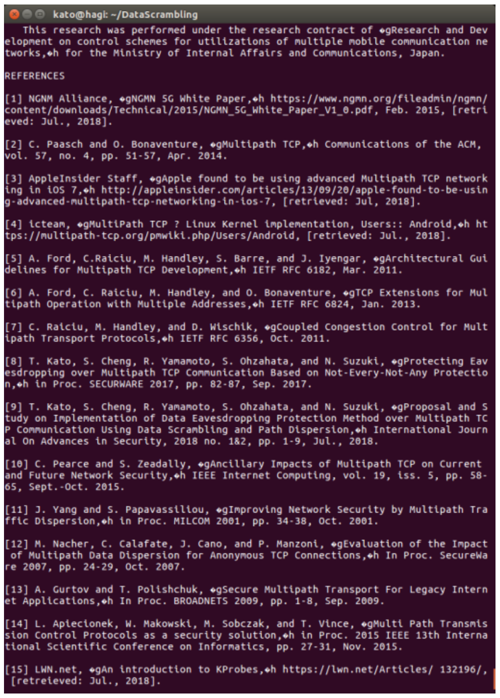
Figure 14. Capturing result of iperf when scrambling is performed.

This figure shows the monitoring result for the first data segment over the WLAN link, which is the same with Figure 13. The original data is a repetition of "0123456789" but the data is scrambled in the result here. So, it can be said that the attacker cannot understand the content, even the WLAN link is not encrypted.

Figures 15 through 17 show result of the attacker's monitoring of file transfer over WLAN. Figure 15 is a display image at the client node when it is receiving a text file containing the text of our previous paper. It is the part of references in the paper. Figure 16 shows a monitoring result by the attacker when no data scrambling is performed. This figure is a result with tshark command with "−x" option that requests to display of data part in TCP segments. As shown in this figure, the content of file can be obtained by the attacker. Figure 17 shows the monitoring result by the attacker for the same part of file given in Figure 16. Since the content is scrambled, the attacker cannot recognize the content of the file.

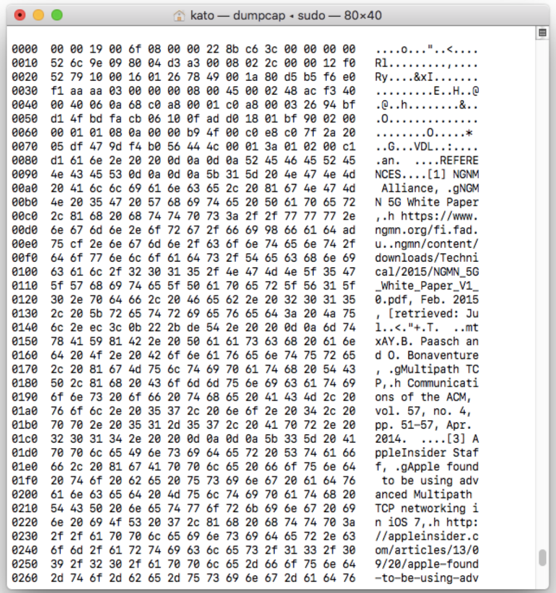
C. *Throughput evaluation*

In order to evaluation the performance of the proposed method, we measured file transfer throughput for the original MPTCP (without data scrambling nor encryption), the proposed method, and the MPTCP data transfer with encryption and decryption using Advanced Encryption Standard (AES) [23]. AES is a block based ciphering algorithm standardized by NIST in 2001. It is a symmetric block cipher that can process data blocks of 128 bits (16 bytes), using cipher keys with lengths of 128, 192, and 256 bits (16 bytes, 24 bytes, and 32 bytes, respectively). In this experiment, we used 16 byte key with the CBC mode. For the implementation of AES based file transfer, we used a publicly available source programs for AES [24] distributed by PJC, a



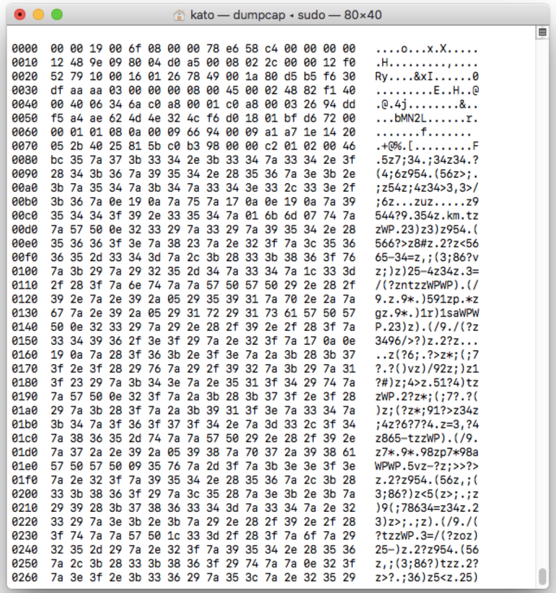
```
  This research was performed under the research contract of ◆gResearch and Dev
elopment on control schemes for utilizations of multiple mobile communication ne
tworks,◆h for the Ministry of Internal Affairs and Communications, Japan.

REFERENCES

[1] NGNM Alliance, ◆gNGMN 5G White Paper,◆h https://www.ngmn.org/fileadmin/ngmn/
content/downloads/Technical/2015/NGMN_5G_White_Paper_V1_0.pdf, Feb. 2015, [retri
eved: Jul., 2018].

[2] C. Paasch and O. Bonaventure, ◆gMultipath TCP,◆h Communications of the ACM,
vol. 57, no. 4, pp. 51-57, Apr. 2014.

[3] AppleInsider Staff, ◆gApple found to be using advanced Multipath TCP network
ing in iOS 7,◆h http://appleinsider.com/articles/13/09/20/apple-found-to-be-usin
g-advanced-multipath-tcp-networking-in-ios-7, [retrieved: Jul, 2018].

[4] icteam, ◆gMultiPath TCP ? Linux Kernel implementation, Users:: Android,◆h ht
tps://multipath-tcp.org/pmwiki.php/Users/Android, [retrieved: Jul., 2018].

[5] A. Ford, C.Raiciu, M. Handley, S. Barre, and J. Iyengar, ◆gArchitectural Gui
delines for Multipath TCP Development,◆h IETF RFC 6182, Mar. 2011.

[6] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, ◆gTCP Extensions for Mul
tipath Operation with Multiple Addresses,◆h IETF RFC 6824, Jan. 2013.

[7] C. Raiciu, M. Handley, and D. Wischik, ◆gCoupled Congestion Control for Mult
ipath Transport Protocols,◆h IETF RFC 6356, Oct. 2011.

[8] T. Kato, S. Cheng, R. Yamamoto, S. Ohzahata, and N. Suzuki, ◆gProtecting Eav
esdropping over Multipath TCP Communication Based on Not-Every-Not-Any Protectio
n,◆h in Proc. SECURWARE 2017, pp. 82-87, Sep. 2017.

[9] T. Kato, S. Cheng, R. Yamamoto, S. Ohzahata, and N. Suzuki, ◆gProposal and S
tudy on Implementation of Data Eavesdropping Protection Method over Multipath TC
P Communication Using Data Scrambling and Path Dispersion,◆h International Journ
al On Advances in Security, 2018 no. 1&2, pp. 1-9, Jul., 2018.

[10] C. Pearce and S. Zeadally, ◆gAncillary Impacts of Multipath TCP on Current
and Future Network Security,◆h IEEE Internet Computing, vol. 19, iss. 5, pp. 58-
65, Sept.-Oct. 2015.

[11] J. Yang and S. Papavassiliou, ◆gImproving Network Security by Multipath Tra
ffic Dispersion,◆h in Proc. MILCOM 2001, pp. 34-38, Oct. 2001.

[12] M. Nacher, C. Calafate, J. Cano, and P. Manzoni, ◆gEvaluation of the Impact
 of Multipath Data Dispersion for Anonymous TCP Connections,◆h In Proc. SecureWa
re 2007, pp. 24-29, Oct. 2007.

[13] A. Gurtov and T. Polishchuk, ◆gSecure Multipath Transport For Legacy Intern
et Applications,◆h In Proc. BROADNETS 2009, pp. 1-8, Sep. 2009.

[14] L. Apiecionek, W. Makowski, M. Sobczak, and T. Vince, ◆gMulti Path Transmis
sion Control Protocols as a security solution,◆h in Proc. 2015 IEEE 13th Interna
tional Scientific Conference on Informatics, pp. 27-31, Nov. 2015.

[15] LWN.net, ◆gAn introduction to KProbes,◆h https://lwn.net/Articles/ 132196/,
 [retreieved: Jul., 2018].
```

Figure 15. Client display image of file receiving.



```
0000  00 00 19 00 6f 08 00 00 22 8b c6 3c 00 00 00 00   ....o..."..<....
0010  52 6c 9e 09 80 04 d3 a3 00 08 02 2c 00 00 12 f0   Rl.........,....
0020  52 79 10 00 16 01 26 78 49 00 1a 80 d5 b5 f6 e0   Ry....&xI.......
0030  f1 aa aa 03 00 00 00 08 00 45 00 02 48 ac f3 40   .........E..H..@
0040  00 40 06 0a 68 c0 a8 00 01 c0 a8 00 03 26 94 bf   .@..h........&..
0050  d1 4f bd fa cb 06 10 0f ad d0 18 01 bf 90 02 00   .O..............
0060  00 01 01 08 0a 00 00 b9 4f 00 c0 e8 c0 7f 2a 20   ........O.....*
0070  05 df 47 9d f4 b0 56 44 4c 00 01 3a 01 02 00 c1   ..G...VDL..:....
0080  d1 61 6e 2e 20 20 0d 0a 0d 0a 52 45 46 45 52 45   .an.  ....REFERE
0090  4e 43 45 53 0d 0a 0d 0a 5b 31 5d 20 4e 47 4e 4d   NCES....[1] NGNM
00a0  20 41 6c 6c 69 61 6e 63 65 2c 20 81 67 4e 47 4d    Alliance, .gNGM
00b0  4e 20 35 47 20 57 68 69 74 65 20 50 61 70 65 72   N 5G White Paper
00c0  2c 81 68 20 68 74 74 70 73 3a 2f 2f 77 77 77 2e   ,.h https://www.
00d0  6e 67 6d 6e 2e 6f 72 67 2f 66 69 98 66 61 64 ad   ngmn.org/fi.fad.
00e0  75 cf 2e 6e 67 6d 6e 2f 63 6f 6e 74 65 6e 74 2f   u..ngmn/content/
00f0  64 6f 77 6e 6c 6f 61 64 73 2f 54 65 63 68 6e 69   downloads/Techni
0100  63 61 6c 2f 32 30 31 35 2f 4e 47 4d 4e 5f 35 47   cal/2015/NGMN_5G
0110  5f 57 68 69 74 65 5f 50 61 70 65 72 5f 56 31 5f   _White_Paper_V1_
0120  30 2e 70 64 66 2c 20 46 65 62 2e 20 32 30 31 35   0.pdf, Feb. 2015
0130  2c 20 5b 72 65 74 72 69 65 76 65 64 3a 20 4a 75   , [retrieved: Ju
0140  6c 2e ec 3c 0b 22 2b de 54 2e 20 20 0d 0a 6d 74   l..<."+.T.  ..mt
0150  78 41 59 81 42 2e 20 50 61 61 73 63 68 20 61 6e   xAY.B. Paasch an
0160  64 20 4f 2e 20 42 6f 6e 61 76 65 6e 74 75 72 65   d O. Bonaventure
0170  2c 20 81 67 4d 75 6c 74 69 70 61 74 68 20 54 43   , .gMultipath TC
0180  50 2c 81 68 20 43 6f 6d 6d 75 6e 69 63 61 74 69   P,.h Communicati
0190  6f 6e 73 20 6f 66 20 74 68 65 20 41 43 4d 2c 20   ons of the ACM,
01a0  76 6f 6c 2e 20 35 37 2c 20 6e 6f 2e 20 34 2c 20   vol. 57, no. 4,
01b0  70 70 2e 20 35 31 2d 35 37 2c 20 41 70 72 2e 20   pp. 51-57, Apr.
01c0  32 30 31 34 2e 20 20 0d 0a 0d 0a 5b 33 5d 20 41   2014.  ....[3] A
01d0  70 70 6c 65 49 6e 73 69 64 65 72 20 53 74 61 66   ppleInsider Staf
01e0  66 2c 20 81 67 41 70 70 6c 65 20 66 6f 75 6e 64   f, .gApple found
01f0  20 74 6f 20 62 65 20 75 73 69 6e 67 20 61 64 76    to be using adv
0200  61 6e 63 65 64 20 4d 75 6c 74 69 70 61 74 68 20   anced Multipath
0210  54 43 50 20 6e 65 74 77 6f 72 6b 69 6e 67 20 69   TCP networking i
0220  6e 20 69 4f 53 20 37 2c 81 68 20 68 74 74 70 3a   n iOS 7,.h http:
0230  2f 2f 61 70 70 6c 65 69 6e 73 69 64 65 72 2e 63   //appleinsider.c
0240  6f 6d 2f 61 72 74 69 63 6c 65 73 2f 31 33 2f 30   om/articles/13/0
0250  39 2f 32 30 2f 61 70 70 6c 65 2d 66 6f 75 6e 64   9/20/apple-found
0260  2d 74 6f 2d 62 65 2d 75 73 69 6e 67 2d 61 64 76   -to-be-using-adv
```

Figure 16. Capturing result of file transfer without scrambling.



```
0000  00 00 19 00 6f 08 00 00 78 e6 58 c4 00 00 00 00   ....o...x.X.....
0010  12 48 9e 09 80 04 d0 a5 00 08 02 2c 00 00 12 f0   .H.........,....
0020  52 79 10 00 16 01 26 78 49 00 1a 80 d5 b5 f6 30   Ry....&xI......0
0030  df aa aa 03 00 00 00 08 00 45 00 02 48 82 f1 40   .........E..H..@
0040  00 40 06 34 6a c0 a8 00 01 c0 a8 00 03 26 94 dd   .@.4j........&..
0050  f5 a4 ae 62 4d 4e 32 4c f6 d0 18 01 bf d6 72 00   ...bMN2L......r.
0060  00 01 01 08 0a 00 09 66 94 00 09 a1 a7 1e 14 20   .......f.......
0070  05 2b 40 25 81 5b c0 b3 98 00 00 c2 01 02 00 46   .+@%.[.........F
0080  bc 35 7a 37 3b 33 34 2e 3b 33 34 7a 33 34 2e 3f   .5z7;34.;34z34.?
0090  28 34 3b 36 7a 39 35 34 2e 28 35 36 7a 3e 3b 2e   (4;6z954.(56z>;.
00a0  3b 7a 35 34 7a 3b 34 7a 33 34 3e 33 2c 33 3e 2f   ;z54z;4z34>3,3>/
00b0  3b 36 7a 0e 19 0a 7a 75 7a 17 0a 0e 19 0a 7a 39   ;6z...zuz.....z9
00c0  35 34 34 3f 39 2e 33 35 34 7a 01 6b 6d 07 74 7a   544?9.354z.km.tz
00d0  7a 57 50 0e 32 33 29 7a 33 29 7a 39 35 34 2e 28   zWP.23)z3)z954.(
00e0  35 36 36 3f 3e 7a 38 23 7a 2e 32 3f 7a 3c 35 36   566?>z8#z.2?z<56
00f0  36 35 2d 33 34 3d 7a 2c 3b 28 33 3b 38 36 3f 76   65-34=z,;(3;86?v
0100  7a 3b 29 7a 29 32 35 2d 34 7a 33 34 7a 1c 33 3d   z;)z)25-4z34z.3=
0110  2f 28 3f 7a 6e 74 7a 7a 57 50 57 50 29 2e 28 2f   /(?zntzzWPWP).(/
0120  39 2e 7a 2e 39 2a 05 29 35 39 31 7a 70 2e 2a 7a   9.z.9*.)591zp.*z
0130  67 7a 2e 39 2a 05 29 31 72 29 31 73 61 57 50 57   gz.9*.)1r)1saWPW
0140  50 0e 32 33 29 7a 29 2e 28 2f 39 2e 2f 28 3f 7a   P.23)z).(/9./(?z
0150  33 34 39 36 2f 3e 3f 29 7a 2e 32 3f 7a 17 0a 0e   3496/>?)z.2?z...
0160  19 0a 7a 28 3f 36 3b 2e 3f 3e 7a 2a 3b 28 3b 37   ..z(?6;.?>z*;(;7
0170  3f 2e 3f 28 29 76 7a 29 2f 39 32 7a 3b 29 7a 31   ?.?()vz)/92z;)z1
0180  3f 23 29 7a 3b 34 3e 7a 2e 35 31 3f 34 29 74 7a   ?#)z;4>z.51?4)tz
0190  7a 57 50 0e 32 3f 7a 2a 3b 28 3b 37 3f 2e 3f 28   zWP.2?z*;(;7?.?(
01a0  29 7a 3b 28 3f 7a 2a 3b 39 31 3f 3e 7a 33 34 7a   )z;(?z*;91?>z34z
01b0  3b 34 7a 3f 36 3f 37 3f 34 2e 7a 3d 33 2c 3f 34   ;4z?6?7?4.z=3,?4
01c0  7a 38 36 35 2d 74 7a 7a 57 50 29 2e 28 2f 39 2e   z865-tzzWP).(/9.
01d0  7a 37 2a 2e 39 2a 05 39 38 7a 70 37 2a 39 38 61   z7*.9*.98zp7*98a
01e0  57 50 57 50 09 35 76 7a 2d 3f 7a 3b 3e 3e 3f 3e   WPWP.5vz-?z;>>?>
01f0  7a 2e 32 3f 7a 39 35 34 2e 28 35 36 7a 2c 3b 28   z.2?z954.(56z,;(
0200  33 3b 38 36 3f 29 7a 3c 35 28 7a 3e 3b 2e 3b 7a   3;86?)z<5(z>;.;z
0210  29 39 28 3b 37 38 36 33 34 3d 7a 33 34 7a 2e 32   )9(;78634=z34z.2
0220  33 29 7a 3e 3b 2e 3b 7a 29 2e 28 2f 39 2e 2f 28   3)z>;.;z).(/9./(
0230  3f 74 7a 7a 57 50 1c 33 3d 2f 28 3f 7a 6f 7a 29   ?tzzWP.3=/(?zoz)
0240  32 35 2d 29 7a 2e 32 3f 7a 39 35 34 2e 28 35 36   25-)z.2?z954.(56
0250  7a 2c 3b 28 33 3b 38 36 3f 29 74 7a 7a 0e 32 3f   z,;(3;86?)tzz.2?
0260  7a 3e 3f 2e 3b 33 36 29 7a 35 3c 7a 2e 32 35 29   z>?.;36)z5<z.25)
```

Figure 17. Capturing result of file transfer with scrambling.

Japanese software company. In the throughput measurement, data is transferred from the server node to the client node in the configuration given in Figure 12. We measured the throughput by changing the size of file transferred.

Figure 18 shows the measured throughput. When changing the transferred file size from 20 MB through 80 MB, the results are similar for all the sizes in every case. The

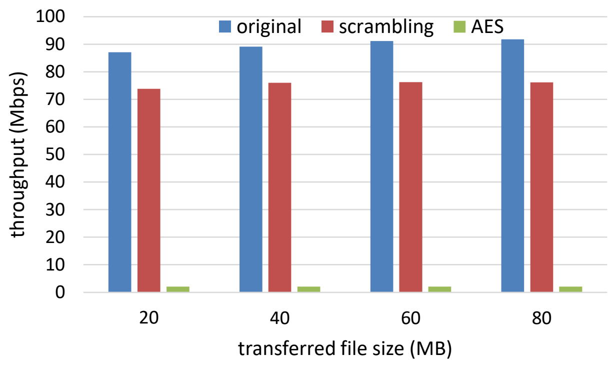


Figure 18. Measured throughput.

throughput of the original MPTCP is the highest among three and the result is around 90 Mbps. In the case of the proposed method, the measured throughput is around 75 Mbps, which is lower than the original MPTCP but high enough. In the case of AES encryption, the throughput goes down to 2 Mbps. Comparing the encryption and decryption, the decryption has higher overhead in our experiment, and it limits the performance of file transfer. It should be noted that the software we used for AES encryption may not be optimized and so the sophisticated AES program may improve the throughput. But, it can be said that the proposed method will require much less overhead than the encryption based method. Another thing to be mentioned is that the equipment we used in this experiment is rather old and so the processing power of CPU is not high. This is one factor that makes the throughput of the AES based method worse.

VI. Conclusions

This paper described the results of implementation and performance evaluation of a method to improve privacy against eavesdropping over MPTCP communications, which we proposed in the previous papers. The proposed method here is based on the not-every-not-any protection principle, that is, *if an attacker cannot observe the data over trusted path such as an LTE network, he/she cannot observe the traffic on any path*. Specifically, the proposed method uses the byte oriented data scrambling and the data dispersion over multiple paths.

In the implementation of the proposed method, we took an approach to avoid the modification of the Linux kernel as much as possible. The modification is as follows. The control parameters are inserted in the socket data structure, and the dummy function for the last part of `tcp_recvmsg()` function. The main part of scrambling and descrambling is implemented by use of the kernel debugging routine called JProbe handler, which is independent of the kernel.

Through the experiment, we confirmed that the data transferred over unencrypted WLAN link cannot be recognized when the data scrambling is performed. As for the performance, the throughput of the scrambled communication is just a little smaller than the original MPTCP communication exposed to unauthorized access. Moreover, the throughput of cryptographic method is degraded largely compared with the original MPTCP and the proposed method. For the terminals with low power CPU, the cryptographic approach will decrease the throughput and so the proposed method will be effective.

In the last of this paper, we need to say that the proposed method is a practical approach based on the assumption that the trusted path, for example, a path via a trusted network operator, is safe enough. That means that, if the trusted path is accessed in an unauthorized way, the data will be observed thoroughly. By owing to the safety in the trusted networks, the proposed method provides low overhead in the data protection.

Acknowledgment

This research was performed under the research contract of "Research and Development on control schemes for utilizations of multiple mobile communication networks," for the Ministry of Internal Affairs and Communications, Japan.

References

[1] T. Kato, S. Cheng, R. Yamamoto, S. Ohzahata, and N. Suzuki, "Implementation of Eavesdropping Protection Method over MPTCP Using Data Scrambling and Path Dispersion," in Proc. SECURWARE 2018, pp. 108-113. Sep. 2018.

[2] NGNM Alliance, "NGMN 5G White Paper," https://www.ngmn.org/fileadmin/ngmn/content/downloads/Technical/2015/NGMN_5G_White_Paper_V1_0.pdf, Feb. 2015, [retrieved: Feb., 2019].

[3] C. Paasch and O. Bonaventure, "Multipath TCP," Communications of the ACM, vol. 57, no. 4, pp. 51-57, Apr. 2014.

[4] AppleInsider Staff, "Apple found to be using advanced Multipath TCP networking in iOS 7," http://appleinsider.com/articles/13/09/20/apple-found-to-be-using-advanced-multipath-tcp-networking-in-ios-7, [retrieved: Feb., 2019].

[5] icteam, "MultiPath TCP – Linux Kernel implementation, Users:: Android," https://multipath-tcp.org/pmwiki.php/Users/Android, [retrieved: Feb., 2019].

[6] A. Ford, C.Raiciu, M. Handley, S. Barre, and J. Iyengar, "Architectural Guidelines for Multipath TCP Development," IETF RFC 6182, Mar. 2011.

[7] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses," IETF RFC 6824, Jan. 2013.

[8] C. Raiciu, M. Handley, and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols," IETF RFC 6356, Oct. 2011.

[9] T. Kato, S. Cheng, R. Yamamoto, S. Ohzahata, and N. Suzuki, "Protecting Eavesdropping over Multipath TCP Communication Based on Not-Every-Not-Any Protection," in Proc. SECURWARE 2017, pp. 82-87, Sep. 2017.

[10] T. Kato, S. Cheng, R. Yamamoto, S. Ohzahata, and N. Suzuki, "Proposal and Study on Implementation of Data Eavesdropping Protection Method over Multipath TCP Communication Using Data Scrambling and Path Dispersion," International Journal On Advances in Security, 2018 no. 1&2, pp. 1-9, Jul. 2018.

[11] C. Pearce and S. Zeadally, "Ancillary Impacts of Multipath TCP on Current and Future Network Security," IEEE Internet Computing, vol. 19, iss. 5, pp. 58-65, Sept.-Oct. 2015.

[12] J. Yang and S. Papavassiliou, "Improving Network Security by Multipath Traffic Dispersion," in Proc. MILCOM 2001, pp. 34-38, Oct. 2001.

[13] M. Nacher, C. Calafate, J. Cano, and P. Manzoni, "Evaluation of the Impact of Multipath Data Dispersion for Anonymous TCP Connections," In Proc. SecureWare 2007, pp. 24-29, Oct. 2007.

[14] A. Gurtov and T. Polishchuk, “Secure Multipath Transport For Legacy Internet Applications,” In Proc. BROADNETS 2009, pp. 1-8, Sep. 2009.

[15] L. Apiecionek, W. Makowski, M. Sobczak, and T. Vince, “Multi Path Transmission Control Protocols as a security solution,” in Proc. 2015 IEEE 13th International Scientific Conference on Informatics, pp. 27-31, Nov. 2015.

[16] A. Shamir, “How to share a secret,” Communications of the ACM, vol. 22, no. 11, pp. 612-613, Nov. 1979.

[17] R. Ahlswede, N. Cai, S. Li, and R. Yeung, “Network Information Flow,” IEEE Trans. Information Theory, vol. 46, no. 4, pp. 1204-1216, Jul. 2000.

[18] M. Li, A. Lukyanenko, and Y. Cui, “Network Coding Based Multipath TCP,” in Proc. Global Internet Symposium 2012, pp. 25-30, Mar. 2012.

[19] ISO JTC 1/SC27, “ISO/IEC 10116: 2006 – Information technology – Security techniques – Modes of operation for an n-bit cipher,” ISO Standards, 2006.

[20] LWN.net, “An introduction to KProbes,” https://lwn.net/Articles/132196/, [retreieved: Feb., 2019].

[21] GitHubGist, “jprobes example: dzeban / jprobe_etn_io.c,” https://gist.github.com/dzeban/a19c711d6b6b1d72e594, [retreieved: Feb., 2019].

[22] S. Seth and M. Venkatesulu, “TCP/IP Architecture, Desgn, and Implementation in Linux,” John Wiley & Sons, 2009.

[23] Federal Information Processing Standards Publication 197, “Anouncing the Advanced Encryption Standard (AES),” Nov. 2001.

[24] PJC, “Distribution of Sample Program / Source / Software (in Japanese),” http://free.pjc.co.jp/index.html, [retrieved: Feb., 2019].

Protecting Deployment Models in Collaborative Cloud Application Development

Vladimir Yussupov, Ghareeb Falazi, Michael Falkenthal, and Frank Leymann
Institute of Architecture of Application Systems
University of Stuttgart, Stuttgart, Germany
email: [firstname.lastname]@iaas.uni-stuttgart.de

Abstract—**Profitability of industrial processes today depends on well-timed utilization of new technologies. Development of cloud applications combining cross-domain knowledge from multiple collaborating parties is one common way to enhance manufacturing. Often, such collaborations are not centralized due to outsourcing or rearrangements in organizational structures. Moreover, manual deployment inefficiency and intellectual property issues further tangle the development process of such applications. While the development of deployment models obviates the necessity to manually deploy applications, a way to protect sensitive data in exchanged deployment models is still needed. In this work, we describe the specifics of modeling and enforcement of security requirements for deployment models in the context of decentralized collaborative cloud application development. We provide a stepwise demonstration of how security requirements can be specified and enforced in a collaborative development scenario based on the TOSCA cloud standard. Furthermore, we conceptualize the system architecture, provide details about the implementation of certain approach-specific operations, and discuss the limitations of the approach. Finally, we show the feasibility of the presented concepts via an open-source prototype.**

Keywords–Collaboration; Security Policy; Confidentiality; Integrity; Deployment Model; Deployment Automation; TOSCA.

I. Introduction

In the recent years, processes and technologies fostering manufacturing automation gained a lot of attention from both, industry and academia. Often, an intricacy of industrial processes leads to the fact that desired automation goals can only be achieved using custom-tailored software solutions. Frequently, such software is the result of a teamwork involving multiple independent parties, e.g., representing different participating organizations. As a prerequisite for collective software development to be successful, often, various functional and non-functional system requirements have to be satisfied. Security and privacy of the data exchanged among involved parties are critically important requirements that have to be properly documented and enforced during the development lifecycle [1].

Numerous modern computing paradigms have great potential for accelerating the 4^{th} industrial revolution, often referred to as Industry 4.0 [2]. One notable example is the rapidly evolving field of cloud computing [3], which allows on-demand access to potentially unbounded number of computing resources. Combined together with ubiquitous sensors usage in the context of the Internet of Things (IoT) [4], cloud computing facilitates the development of composite, cross-domain applications tailored specifically for automation and optimization of manufacturing. Along with the clear advantages, such emerging technologies introduce additional challenges that need to be tackled. For instance, the overall complexity of development processes might become a significant obstacle for industries willing to benefit from cloud applications.

A typical cloud application today has a composite structure consisting of multiple interconnected and heterogeneous components [5]. Deploying such complexly-structured applications in a manual fashion is error-prone and inefficient [6]. Therefore, various deployment automation approaches exist. One well-established automation technique relies on the concept of *deployment models* that specify application structure along with the necessary deployment information. Automated processing of such models considerably reduces the deployment's complexity and minimizes required efforts. Another significant benefit, which improves portability and reusability aspects of the application development process, is that instead of separate application components, standardized models can be exchanged [7].

Complexity and heterogeneity of application's components are among the reasons why a common cloud application development scenario in the context of Industry 4.0 is a collaboration [8] involving several multidisciplinary partners responsible for separate parts of the application [5]. The final goal of this collaboration is to combine all parts into a complete and deployable cloud application. Collaborative development can significantly benefit from the portability and reusability properties of deployment models. However, since not all parties are known in advance, e.g., due to task outsourcing or changes in organizational structure, the issues of intellectual property protection in decentralized settings arise. For instance, confidential information like sensor measurements and proprietary algorithms might be subject to various *security requirements*, including protection from unauthorized access and verification of its integrity. Therefore, modeling and enforcement of such requirements aimed at specific parts of deployment models, have to be supported.

In our previous work [1], we introduced a method for modeling and enforcement of security requirements in deployment models that combines the ideas of sticky policies [9], policy-based cryptography [10], and Cryptographic Access Control (CAC) [11]. In this paper, we build upon our previous work and discuss in more details, how security requirements aimed at data protection in modeled cloud applications can be expressed using security policies and which parts of deployment models need to support the attachment of security policies. Focusing more on the practical aspects, we provide a stepwise demonstration of how the introduced approach can be applied to a collaborative and standardized process of deployment model development. To have a uniform way of deployment modeling, we use the existing OASIS standard, Topology and Orchestration Specification for Cloud Applications (TOSCA) [12], [13], which specifies an extensible, provider-agnostic cloud modeling language [14]. To validate our concepts, we implement them in OpenTOSCA [15], an opensource ecosystem that allows modeling and execution of TOSCA-compliant deployment models. Moreover, we include a detailed description of the system architecture and elaborate on the process of security requirements enforcement during import and export of deployment models using our prototype. The remainder of this paper is structured as follows. As in our previous work [1], we first describe the fundamentals underlying this work in Section II and discuss a motivational scenario in Section III. In Section IV, we present concepts for modeling and enforcement of security requirements in collaborative deployment models development. In Section V, we apply the concepts to a TOSCA-based deployment modeling process and provide a demonstration of a TOSCA-based collaborative development scenario using the example collaboration described in the motivational scenario. The details about the prototypical implementation in OpenTOSCA are discussed in Section VI. In addition, we discuss the system's architecture and describe the specifics of import and export processes of deployment models. Finally, in Section VII, we describe related work and conclude this paper in Section VIII.

II. Fundamentals

In this section, we provide an overview of several important concepts that serve as a basis for our work, namely: (i) deployment automation of cloud applications by means of deployment modeling approaches, (ii) usage of policies as means to specify non-functional system requirements, (iii) and a brief coverage of access control mechanisms.

A. Deployment Modeling

The compound application structure and increased integration complexity make it non-trivial to automate the deployment of modern cloud applications [6]. The concept of deployment modeling aims to tackle the automation problem, and there are several known approaches including imperative and declarative modeling [6], [16], [17]. Both paradigms are based on the idea of creating a description, or deployment model, sufficient enough for deploying a chosen application in an automated fashion. What makes these modeling approaches different is the way how corresponding deployment models are implemented.

In case of the declarative modeling [16], a deployment model is a structural model that conveys the desired state and structure of the application. Essential parts of the declarative deployment model include a specification of application's components with respective dependencies and necessary connectivity details. As a result, the model might contain binaries or scripts responsible for running some application's components, e.g., a specific version of Apache Tomcat, or a predefined Shell script for running a set of configuration commands. In addition, a description of non-functional system requirements in some form can be included into the model. Some examples supporting this type of modeling include Chef [18] and Juju [19] automation tools, as well as TOSCA. This type of models relies on the concept of deployment engines, which are able to interpret a provided description and infer a sequence of steps required for successful deployment of the modeled application.

Compared to the declarative approach, the imperative modeling [16] focuses on a procedure, which leads to automatic application deployment. More specifically, an imperative model describes (i) a set of activities corresponding to the required deployment tasks that need to be executed, and (ii) the control and data flow between those activities. One robust technique for this modeling style is to use a process engine, e.g., supporting standards like Business Process Execution Language (BPEL) [20] or Business Process Model and Notation (BPMN) [21], which can execute provided imperative models in an automated fashion.

A combination of declarative and imperative approaches is also possible. In general, creating both types of models requires efforts from the modeler. However, the imperative modeling approach is generally more time-consuming and error-prone, since multiple heterogeneous components need to be properly orchestrated. Moreover, the structure of the application might change frequently, which requires to modify imperative models. To minimize required modeling efforts, imperative models might be derived from the provided declarative models [6].

One important aspect of deployment models is that apart from valid descriptions they also need to include various files related to described software components and other parts of the application, e.g., scripts, binaries, documentation and license details. As a result, the term deployment model usually refers to a combination of all the corresponding metadata and application files required for automatically deploying a target application.

B. Policies

One well-known approach for separating non-functional requirements from the actual functionalities of a target system relies on the usage of policies [22]. Essentially, a *policy* is a semi-structured representation of a certain management goal [23]. The term management here is rather broad, as it might refer to different aspects of management, e.g., high-level corporate goals or more low-level, technology-oriented management goals. For instance, from the system's perspective, performance, configuration, and security are among the classes of non-functional requirements that can be described using policies. Additionally, various policy specification languages exist in order to simplify the process of describing such requirements in a standardized manner [22]. From the high-level view, policies only declare the requirements, which then have to be enforced using dedicated enforcement mechanisms [24].

The idea to specify security requirements in policies dates back to at least the 1970s [22]. Depending on the level of details security policies might specify, e.g., privacy requirements for the whole system or for particular data objects. In information exchange scenarios, security policies specified on the level of data objects have to be ensured during the whole exchange process [25]. For this reason, all receivers have to be aware of specified policies and enforcement must happen, e.g., by means of globally-available security mechanisms. Similarly, deployment models in collaborative application development are constantly exchanged and parts of them might be subject to security policies. So-called *sticky policies* [25] is an approach to propagate policies with the data they target. This approach can be combined with cryptography in order to ensure that data is accessed only when requirements specified in policies are satisfied. Multiple approaches to combine sticky policies with different cryptographic techniques such as public key encryption or Attribute-Based Encryption (ABE) exist [26].

C. Access Control

A secure information system must prevent disclosure (confidentiality) or modification (integrity) of sensitive data to an unauthorized party and ensure that data are accessible (availability) [24]. These requirements can be enforced by assuring only authorized access to the system and its resources. Commonly, this process is referred to as *access control* and there exist multiple well-established access control mechanisms. For example, in Discretionary Access Control (DAC) mechanism, the access is defined based on the user's identity. This results in access rules that are specified specifically for this identity, e.g., in the form of an access control matrix [27]. Another well-known access control mechanism is called Role-Based Access Control (RBAC) where access is granted or denied based on the user roles and access rules defined for these roles.

One disadvantage of the aforementioned access control mechanisms is that they commonly rely on some centralized trusted authority, making it difficult to implement them in large scale and open systems [11]. The idea of CAC is based on well-known cryptographic mechanisms and regulates access permissions based on the possession of encryption keys. In CAC, the stored data are encrypted and can only be accessed by those users who have the corresponding keys. One positive advantage of this approach is that the data owner can grant keys to other involved parties of his choice using established key distribution mechanisms, thus enforcing the access control without relying on the trusted third party.

III. Motivational Scenario

Developing distributed cloud applications and analytics applications in the context of Industry 4.0 typically requires combining numerous heterogeneous software components [28], [29]. Commonly, this process implies a collaboration among experts from various domains, such as data scientists, infrastructure integrators, and application providers. Furthermore, resulting applications are often required to be deployable on demand and, thus, are expected to be in the form of deployment models that allow automating application provisioning [5], [30].

An example of a collaborative cloud application development depicted in Figure 1 involves four participants responsible for distinct parts of the application. When joined together, all developed parts of the application, e.g., software components, datasets, and connectivity information, comprise a complete and provisioning-ready deployment model. In this scenario, the main beneficiary who orders the application from a set of partners and has exclusive rights on the resulting deployment model is called the *Application Owner*. The *Infrastructure Modeler* is responsible for integrating different components, such as analytics runtime environments, databases, or application servers. Moreover, two additional co-modelers are involved in the development process, namely a *Data Scientist* and a *Dataset Provider*. The former develops a certain proprietary algorithm, whereas the latter provides a private dataset, e.g., comprised of sensor measurements obtained from a combination of various cyber-physical systems used in production processes.

While the Application Owner has full rights on the resulting deployment model, other participants might be subject to security restrictions. For example, access to the dataset provided by the Dataset Provider might need to be restricted to some of the parties. Similarly, the Data Scientist might want to specify security requirements on the provided algorithm. Since the final infrastructure must include all corresponding sub-parts that were provided directly or indirectly by participants, the Infrastructure Modeler is responsible for preparation and shipping of the finalized deployment model to the Application Owner who is then able to create new instances of the application on demand.

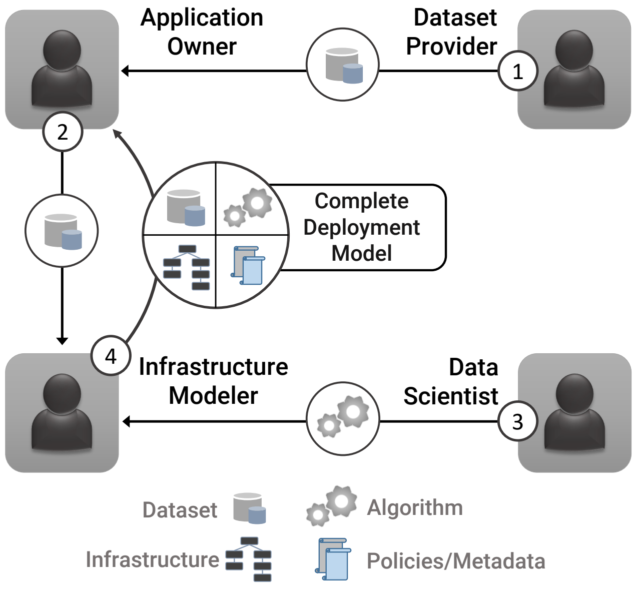


Figure 1. A collaborative application development scenario.

Generally, collaborative processes from various fields share some common characteristics. For instance, according to Wang et al. [31] such issues as (i) *dynamically changing sets of participants*, (ii) the *lack of centralization*, (iii) *intellectual property and trust management issues*, and (iv) *heterogeneity of exchanged data* are important in collaborative development of computer-aided design models. Likewise, the lack of knowledge about all participants involved in collaborative cloud application development makes it difficult to establish a centralized interaction among them. Possible reasons include outsourcing of development tasks and introduction of additional participants due to rearrangements in organizational structures. Since no strict centralization is possible, communication with known participants happens in a peer-to-peer manner. Another important aspect of collaborative cloud application development is its iterative nature. Since exchanged deployment models might be impartial or require several rounds of refinement, a potentially complicated sequence of exchange steps is possible for obtaining a final result. Therefore, deployment models need to be exchanged in collaborations in a way that simplifies the overall process and enforces potential security requirements.

A deployment model, generally, can be exchanged either in a self-contained form or on a per-participant basis. In the former case, the deployment model is self-contained and its content is the same for all participants, whereas in the latter case its content is fragmented according to some rules separately for each participant. Sometimes, however, exchanging deployment models on a per-participant basis interferes with the actual goals of the collaboration. For example, in the exchange sequence shown in Figure 1 the dataset is firstly passed directly to the Application Owner by the Dataset Provider. For the integration of the dataset into the final model, the Infrastructure Modeler needs to model the required infrastructure, e.g., a Database Management System (DBMS) and related tooling. As only the Application Owner has full rights on all parts of the application, the provided dataset has to be protected from unauthorized access. Intellectual property issues become even more complex in highly-dynamic scenarios when multiple parties continuously exchange partially-completed deployment models. Unfortunately, encrypting an entire deployment model does not solve the problem since models might be intended to remain partially-accessible by parties with limited access rights. Apart from confidentiality problems, the authenticity and integrity of passed deployment models and their parts might be subject to verification requirements. For instance, the Application Owner might need to check if an algorithm was actually provided by the Data Scientist and no changes were made by other parties. In such case, signing the hash value of an entire deployment model is not suitable as integrity of individual model's parts have to be verified. Hence, it should be possible to verify distinct parts of deployment models independently.

The aforementioned scenario highlights several important issues in collaborative development of deployment models, which need to be solved, namely (i) confidentiality, authenticity, and integrity requirements of each involved participant have to be reflected in the model, (ii) various levels of granularity for these requirements need to be considered, i.e., from full models to their separate parts, and (iii) a method to enforce modeled requirements in a peer-to-peer model exchange is needed.

IV. Modeling and Enforcement of Security Requirements

Intellectual property in collaborations has to be protected from both, external and internal adversaries with respect to their relation to the process. The former describes any attacker from outside of the collaboration, i.e., who is not participating and is not reflected in any kind of agreements, e.g., Service Level Agreements (SLAs). Conversely, the latter refers to a dishonest party involved in the process. We focus on internal adversaries and data protection issues involving known parties.

This section presents an approach to ensure the fulfillment of security requirements in the collaborative development of deployment models. Our approach relies on the well-established concept of representing non-functional requirements via policies [32], [33], [34], [35]. The semantics of security requirements is analyzed to derive a set of action and grouping policies. The former type represents cryptographic operations allowing to enforce confidentiality and integrity requirements, inspired by the idea of policy-based cryptography [10]. The latter type simplifies grouping parts of models that are subjects to action policies. Both policy types are data-centric and attachment happens with respect to a certain entity or a group of entities in the manner of sticky policies [25] to preserve the self-containment property of deployment models. The access control enforcement is inspired by the idea of CAC [11].

A. Assumptions

To focus on internal adversaries, we assume that participants establish bidirectional secure communication channels for data exchange and that the modeling environment of every involved participant is secure. By modeling environment we mean any software that simplifies the process of deployment modeling, e.g., by providing functionalities like loading (import) and packaging (export) of deployment models of different formats or performing various kinds of model validation. We employ an "honest but curious" [36], [37], [38] adversary model in which adversaries are interested in reading the data, but avoid modifications to remain undetected. Despite the absence of modifications made by adversaries, authenticity and integrity requirements still need to be modeled and enforced. For instance, participants might want to track changes or verify the origin of some specific part in the model.

When describing how data encryption can be modeled, we assume that no double encryption is needed for distinct parts of deployment models. We do not distinguish between read and write rights when discussing access control based on cryptographic key possession. Therefore, a participant with the required key is assumed to have full access rights on the corresponding entity. For efficiency reasons, we adopt symmetric encryption for ensuring the confidentiality of data.

B. Security Policies in Collaborative Deployment Models

An assumption that data is exchanged in a secure manner among the participants does not guarantee that all involved parties can be trusted. Therefore, security requirements are important even under the secure communication channels assumption. Security requirements we focus on are: (i) protection of data confidentiality in deployment models, and (ii) verification of data integrity and authenticity of deployment models. On the conceptual level, two distinct types of policies, namely *encryption policy* and *signing policy*, can be distinguished. The former is aimed to solve the confidentiality problem, whereas the latter targets integrity-related requirements. However, having a completely encrypted deployment model does not solve the confidentiality problem, since a party with limited rights will not be able to access the parts of the deployment model that were intended to remain accessible. A similar problem might arise for a signature of the complete packaged deployment model, e.g., in a form of an archive, since it will not be possible to check what exactly was changed unless all files are also signed separately as a part of the process. More specifically, if only the hash of an entire deployment model was signed, there will be no way to distinguish, which specific part of the model is invalid. Therefore, we need to model security policies on the level of atomic entities in deployment models to support collaborations similar to the scenario described in Section III.

Naturally, if only parts of deployment models are subject to confidentiality requirements, enforcement of encryption and signing policies must affect only respective entities. In our approach, an encryption policy attached to a certain entity of the deployment model signals that it has to be encrypted. In a similar manner, if a certain entity of the deployment model needs to be signed, a corresponding signing policy needs to be linked with it. In both cases, policies represent actual keys that are going to be used for encryption or signing. Since not all collaborations can rely on a centralized way to manage policies, the deployment model has to be transferred together with corresponding policies attached to its entities. The keys bound to policies, however, cannot be embedded, as deployment models will no longer remain suitable for sharing with all possible participants in a self-contained fashion. In such cases, either participants with proper access control rights can receive such models, or the models have to be split on a per-participant basis. Since not all scenarios favor participant-wise model splitting, a policy needs to be linked with a specific key in a decoupled manner to preserve self-containment of a deployment model. As a side effect of decoupling keys from policies, existing key distribution channels can be utilized independently from deployment model exchange channels.

For linking policies with particular keys, we need to maintain unique identifiers for every key involved in the collaboration. Since not all participants know each other, one simple solution is to compute a digest of the key and use it as an identifier or additionally combine it with several other parameters such as algorithm details, participant identifier, etc. Another option is to use identifiers, which include some partner-specific parts so that policies can be easily identified. Several important points have to be mentioned here. Linking the policy only with the unique key identifier is not enough for decryption since the modeler needs to know the algorithm details to perform decryption. Such information can be provided either as properties of a policy itself or be a part of the key exchange. Additionally, specifically for encryption there is no obvious way to distinguish if the policy was already applied and the data is in encrypted state when a deployment model is received. Although the data format after encryption will not be identical to the original entity's format, checking this difference for every modeled entity is not efficient. For this reason, a policy needs to have an attribute stating that it was applied. Due to the usage of symmetric encryption, generating a respective *decryption policy* is unnecessary as it is identical to the encryption policy.

Conversely, the verification of signing policies differs from the encryption process since private keys are used for signing and certificate chains of one or more certificates containing the public key and identity information are used for verification. As a result, there are two options: to follow the encryption approach and decouple certificates from policies, or, to embed certificates into policies to simplify the verification process. While certificates are meant for distribution, there is one caveat in the embedding of certificates approach, however. Certificates commonly have a validity period and verification must be able to deal with the cases when certificates embedded into policies are no longer valid. Since such verification is more an issue of a proper tooling, the certificates are embedded into policies.

Unlike file artifacts, e.g., software components or datasets, which are referenced from deployment models and supplied alongside with them, some sensitive information, e.g., model's properties, might be directly embedded into models. For instance, if user credentials for a third-party service have to be passed from one modeler to another and no other participant is allowed to see them, then these properties must be encrypted. Sometimes such properties also need to be verified, e.g., the Service Owner might want to check if the endpoint information for a third-party service was actually modeled by the Infrastructure Modeler. Therefore, an additional caveat one has to consider is that not only distinct artifacts, but also separate parts of artifacts might require encryption or signing. The corresponding artifact in this case has to store these properties with the modeled security requirements being enforced, e.g., encrypted or signed.

Hence, we need two more policy types: *encryption grouping policy* and *signing grouping policy*, which contain lists of properties within an artifact that have to be encrypted or signed, respectively. From the conceptual point of view, the discussed policies can be classified as *action* and *grouping* policies. The former includes policies representing an action, i.e., encryption or signing, whereas the latter identifies groups of entities that require the action. As a result, the corresponding grouping policies are linked with the desired action policies, i.e., with actual keys that will be applied to selected properties.

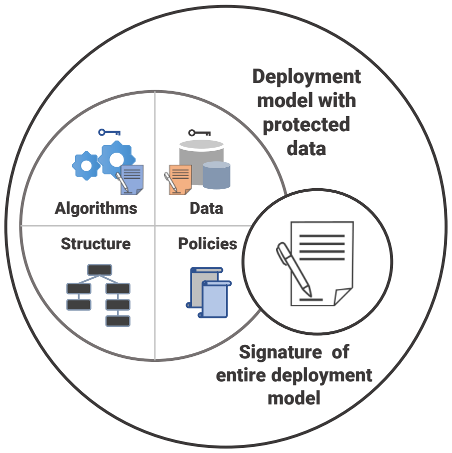

Deployment model with protected data
Algorithms
Data
Structure
Policies
Signature of entire deployment model

Figure 2. A conceptual model of the signed deployment model.

C. *Integrity and Self-Containment of Deployment Models*

When security policies are modeled and enforced, the resulting deployment model contains a combination of encrypted and signed artifacts and properties. Integrity check at this point allows to verify the state of modifications and authenticity of entities modeled by other participants. However, verification of the entire deployment model's integrity including modeled security policies and other attached metadata requires an additional signature on the level of deployment model.

For this purpose we adopt a technique analogous to signing of Java archives (JARs) [39]. Essentially, a packaged deployment model is some sort of an archive containing grouped artifacts. It is then possible to assume the presence of a meta file similar to manifest in JARs, which provides the list of all contents plus some additional information. In situations when such a *manifest* file does not exist, it can easily be generated by traversing the contents of a corresponding deployment model.

As both, integrity of the model's parts that are targeted by security requirements and integrity of the entire deployment model have to be considered, an enhanced packaging format is needed. The enhanced structure of a deployment model consists of its original content as well as the content's signature files. The latter is achieved via a combination of: (i) a manifest file with digests for every file, (ii) a signature file consisting of digests for every digest given in the manifest file plus the digest of the manifest file itself, and (iii) a signature block file consisting of a signature generated by the modeler and the certificate details. The resulting conceptual model is shown in Figure 2. To make a signed deployment model distinguishable from regular deployment models, the signature has to be generated in a standardized fashion, e.g., it can be stored in a predefined folder inside the package or entire deployment models can be archived along with the generated signature information.

One important issue is that, technically, there is no fixed concept of a deployment model in collaboration. Since parts of cloud applications might be exchanged separately or merged together, the definition of the exchanged deployment model is changing throughout the process. Thus, it is mandatory to preserve the self-containment of modeled security requirements on the level of atomic entities. Firstly, security policies are always included into the deployment model since they are tightly-coupled with target entities. With respect to actual

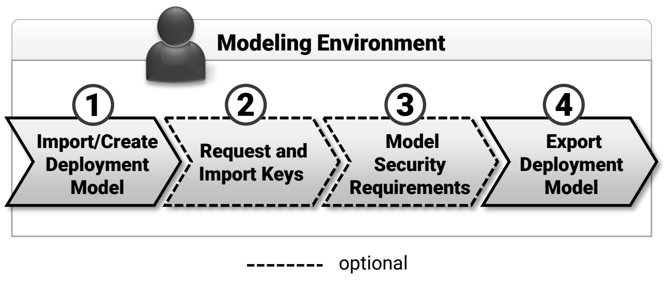


Figure 3. Actions of a collaborating participant.

entities, the problem is trivial in case of encryption since locations of files or properties remain unchanged and only their state changes. In other words, whether the encrypted entity is exported from or imported into the modeling environment, the information about encryption is always available. Conversely, signatures of modeled entities have to be created as separate files since embedding them might not always work. For instance, embedding a signature into the application's source code might result in an incorrect behavior at runtime. This leads to a requirement of generating and storing signatures in a self-contained manner when signing policies enforcement happens.

In contrast, the signature of an entire deployment model reflects a snapshot of its state at a particular point in time, e.g., when the deployment model was packaged by a certain participant. Semantically, this signature does not mean that all content of the deployment model belongs to a signing party, but only captures the state of the deployment model at export time. In our approach, we use this external signature only for integrity verification at import time, but do not explicitly store it if verification was successful. However, if stored in a centralized or decentralized manner, this type of signature might form an expressive log of all export states, which can later be utilized for audit and compliance checking purposes.

D. Enforcement of Security Policies

As participants of a collaborative development process might not know all involved parties, every side has to maintain a set of permissions for known participants, e.g., in a form similar to the access matrix model [24]. In our case, permissions have to reflect, which policies are available to which participant and are, therefore, used for export and distribution of keys. One caveat is that in long sequences of steps there will be cases when a party does not know which rights with respect to the specific key have to be defined for some of the involved parties. The rules in such collaborations rely on various types of agreements, such as SLAs, which define the lists of trusted parties. Hence, we handle only explicitly mentioned access rights defined by participants and forbid transitive trust [40] propagation.

To enforce security policies in collaborations, participants have to follow a set of actions shown in Figure 3. A new or existing deployment model can be imported into the participant's modeling environment. Signatures are verified for an existing deployment model before import. An entire model's signature is verified first and if verification is successful, all signed entities are verified next. If certificate chains are embedded, all certificates must be valid. The import is aborted in case some signatures or certificates are invalid. A participant might request keys needed for encrypted entities and if access is granted by the key owner, keys can be imported into the modeling environment and used for decryption. The policy enforcement at export time happens transparently for participants as entities always get encrypted if the respective keys are present. Since decryption is only possible when the key is available, the encryption at export is ensured by the modeling environment.

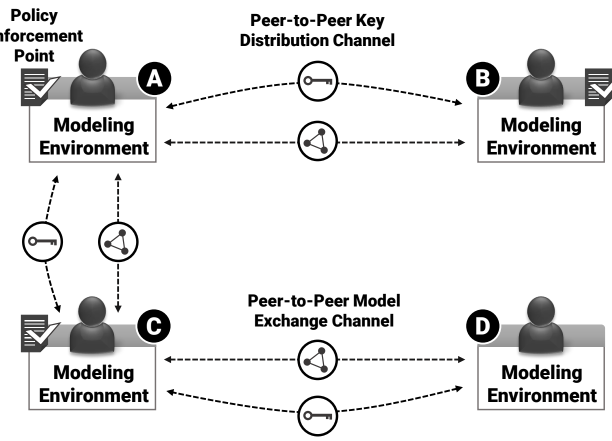


Figure 4. Model and key exchange in collaborations.

Afterwards, participants can model additional security requirements and export a modified deployment model. One issue related to signatures and mutual modifications of the same entity is whether to keep the obsolete signature information. Since the original content of the entity has to be modified, we consider it being a new entity, which can be modeled separately eliminating the problem of handling several signatures altogether. At export time, all modeled requirements are enforced with respect to the keys available in the modeling environment. The decrypted data get encrypted again, in case the corresponding key is present and the entity was decrypted previously. Only signatures modeled by the participant who performs the export are generated. All entities that were signed by others remain in a self-contained state after import and thus are exported in a regular fashion. One important point here is that all enforced signing policies have to be verified before export. In case a violation is found, the export can no longer proceed in the regular way.

Generated signatures must be linked with corresponding modeling constructs. For instance, for every signed file the corresponding signature files must be added as additional linked references, e.g., following a predefined name format "filename#sigtype.sig". Signing properties requires a slightly different approach. Since properties are parts of artifacts and are subject to certain policies, their signatures have to be grouped with respect to the policy. This results in generation of the combined signature file and linking it with the artifact that holds the signed properties. Signature of this file is, again, generated similar to JAR files signing, but in this case the generated artifact contains the details about signed properties.

Figure 4 shows communication infrastructure for collaboration described in Section III. As key distribution is decoupled from the model exchange, two peer-to-peer channel types are distinguished. Generally, not all participants need to communicate with each other. For example, in an outsourcing scenario, a contractor might grant rights to the ordering party based on the contract rules and does not need to communicate with others. Therefore, access permissions of the ordering party have to also reflect access rules for the part of the deployment model provided by the contractor. The access to encrypted data is inquired by requesting a key using the corresponding policy identifier. Without having a centralized Policy Enforcement Point (PEP) [41], [42], every participant's modeling environment acts as a separate PEP, which regulates access control permissions based on inter-participant agreements. Participants are responsible for maintaining proper access control permissions including transitive cases.

V. Standards-based Secure Collaborative Development of Deployment Models

In this section we discuss the specifics of collaborative development of deployment models using TOSCA. We first start with a basic overview of TOSCA concepts and proceed with an analysis of which modeling constructs in TOSCA might require protection. As a next step, we describe how our concepts can be applied to this cloud standard and present a stepwise example demonstrating how collaborative TOSCA development process might look like.

A. *TOSCA Application Model*

TOSCA [12] is a cloud application modeling standard that allows to automate the deployment and management of applications. The structure of a TOSCA application is characterized by descriptions of its components with corresponding connectivity information, modeled as a directed, attributed, and not necessarily connected graph. In TOSCA terminology the entire application model is called a *Service Template*, whereas the connectivity information is a subpart of it and referred to as a *Topology Template*. The management information in TOSCA terms is called *Management Plans*. This information is necessary for execution and management of applications throughout their lifecycle and can be represented, e.g., as BPEL [20] or BPMN [21] models. A simplified TOSCA topology of a Python cloud service [5] is shown in Figure 5. It consists of several *nodes* representing software components that are connected with directed edges describing the *relationships* among them.

TOSCA differentiates between *entity types* and *entity templates*, where the term entity might refer to distinct TOSCA entities such as nodes, relationships, artifacts, or policies. Such separation eases reusing modeled TOSCA entities, since the semantics is always defined in the corresponding type. For instance, the node representing a vSphere hypervisor in Figure 5 is a template *of a certain type*, or, more specifically, a certain *Node Type*. The term "vSphere" written in braces in Figure 5 is the name of this node type. It describes a generic setup of a vSphere virtualization platform and defines all required configuration properties. Correspondingly, the term "vSphere-Hypervisor" written without braces represents a particular instance of the "vSphere" *Node Type* and in TOSCA terms is referred to as a *Node Template*. Apart from defining common properties, any Node Type might provide definitions of interface operations required for managing its instances. For example, a virtual machine node might need to implement management operations such as "start", "stop", and "restart" that allow controlling the state of the virtual machine. Implementations

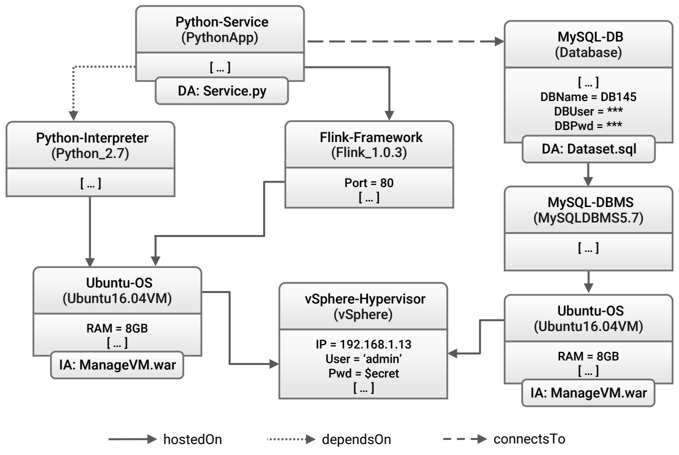


Figure 5. A simplified TOSCA model of a cloud application.

of the operations can be provided in various ways, e.g., in a form of Java web applications or shell scripts.

For deployment and management of the cloud service, all required artifacts have to be modeled, e.g., the application files and implementations of management interface operations. The artifact entity in TOSCA can be of two types, namely deployment artifacts (DA) and implementation artifacts (IA). The former defines an executable required for materialization of a node instance. The latter is a representation of an executable, which implements a certain interface management operation.

One of the main goals of deployment models is to make cloud applications portable and reusable. For this reason TOSCA introduces a self-contained packaging format called Cloud Service Archive (CSAR). Essentially, it is an archive containing all application-related data necessary for automated deployment and management, including, e.g., the model definitions, artifact files, policies and other metadata. In addition, it contains a TOSCA.meta file, which describes files in the archive similarly to a manifest file in JARs.

B. *Security Requirements for TOSCA Entities*

Several TOSCA modeling constructs can be associated with confidential information or be subject to integrity checks. Modeled application files, i.e., artifacts in TOSCA terms, are an obvious example. All artifacts are always modeled as Artifact Templates of desired Artifact Types in TOSCA, e.g., a Java web application artifact is a template of the Web application Archive (WAR) Artifact Type. While Artifact Types are generic entities, which do not store any sensitive data, Artifact Templates include actual application files. However, in the TOSCA specification there is no standard way to describe security requirements using policies for Artifact Templates. To provide such modeling capabilities, an extension to TOSCA is needed. Since properties are defined at the level of Types in TOSCA, e.g., Node Types, it is useful to have a mechanism allowing to enforce security requirements at this level. Semantically, this would mean that encryption or signing policies have to be applied to all Node Templates of a certain Node Type. TOSCA does not offer a standard way of attaching policies to specific properties, thus a proper way to enforce protection of properties at the level of Node Types is needed as well. In both cases, some form of extension to TOSCA is needed. In the next subsection, we demonstrate how such extensions can be made without introducing non-compliant changes to the standard.

C. TOSCA Policy Extensions

Due to the highly-extensible nature of TOSCA, we introduce several extension points to support the attachment of security policies to the aforementioned TOSCA entities. All policies are defined in a dedicated extension element, which belongs to a chosen entity. A simplified XML snippet in Figure 6 shows extension policies for Artifact Templates and Node Types from Figure 5. One important point here is to use a separate namespace for newly-introduced extensible elements, but for the sake of brevity we omit namespaces in demonstrated XML snippets. For Artifact Templates, a security policy is attached in a separate element directly to the Artifact Template. Essentially, an Artifact Template is a container grouping related files in a form of file references. We treat Artifact Templates as atomic entities meaning that policies are applied to all referenced files, which makes the semantics of modeled security requirements clearer. In cases when some referenced files need to be distributed without enforcement of policies, they can be modeled as separate Artifact Templates.

A combination of two policy types has to be defined in a dedicated extension element for encryption and signing of properties. A modeler has to specify a list of property names that must be encrypted or signed as well as to attach a corresponding action policy. These extensions allow participants to model desired security requirements for parts of the CSAR.

The introduced extensions, however, do not offer modeling capabilities for signing the entire CSAR. These two notions of integrity might contradict with each other, since a party having parts of the cloud service belonging to other parties is required to sign them as well. Hence, we separate the integrity check for a specific part of the model from an integrity check of the entire CSAR leaving the latter outside of TOSCA modeling.

Essentially, Policy Types and Templates representing action and grouping policies do not require significant modeling efforts. The Encryption Policy Type defines a key's hash value, an algorithm, and a key's size as its properties. In the corresponding Policy Template, these properties are populated using the respective key's data. Similarly, the Signing Policy Type has public key's hash and related certificate chain as its properties, filled in using the given key. Certificate chain can be embedded, e.g., in a form of a Privacy Enhanced Mail (PEM) encoded string in case of X509 [43] certificates. The only property defined in grouping policies is a space-separated list of property names. This Policy Type is abstract and is not directly bound to any specific entity. Therefore, the tooling is responsible for verification of the consistency of specified property names in attached policies.

D. Ensuring the Self-Containement Property of CSARs

Enforcement of modeled encryption requirements does not produce additional entities, but only modifies the existing ones. However, in case of enforcing signing requirements, new files are generated, i.e., the corresponding signature files for properties or files of Artifact Templates. These generated files have to be associated with the deployment model to ensure that the resulting CSAR contains all files and the deployment model represents these newly-generated files properly. Preservation of a CSAR's self-containment property after enforcement of modeled policies requires embedding the signature information for artifacts and properties into the corresponding entities. More specifically, when a signature for an artifact is created, it has to be placed along with other files referenced in the artifact. For the signature of properties, one artifact containing all properties' signatures needs to be generated and attached to the corresponding Node Template. Following this approach, modeled entities remain self-contained even in case they are being reused in other Service Templates.

```
<ArtifactTemplate name="Python-Service" ...>
  <Policies>
    <Policy applied="false" name="encryption"
         policyType="csar:EncryptionPolicyType"
         policyRef="csar1:c0e9a0e7".../>
  </Policies>
  <ArtifactReferences>
    <ArtifactReference ref=".../Service.py"/>
  </ArtifactReferences>
</ArtifactTemplate>
...
<NodeType name="vSphere" ...>
  <PropertiesDefinition>...</PropertiesDefinition>
  <Policies>
    <Policy ... name="signing" .../>
    <Policy ... name="signedprops" .../>
  </Policies>
</NodeType>
```

Figure 6. Example of TOSCA extension policies specification in XML.

E. Step by Step Collaborative Development Example in TOSCA

For a stepwise demonstration, we consider that the application topology shown in Figure 5 is being collaboratively developed following the same scenario and exchange sequence among the four participants as depicted in Figure 1. This example is not meant to be a thorough guideline for a given collaboration scenario, but rather it aims to demonstrate how certain security requirements can be modeled and enforced throughout the collective TOSCA-based model development.

As discussed in Section III, the four different participants involved in this scenario are: (i) an Application Owner who orders an application, (ii) a Dataset Provider who provides a private dataset, (iii) a Data Scientist who provides a proprietary algorithm, (iv) and an Infrastructure Modeler who defines the infrastructure. The output expected from this collaboration is a complete application topology, which is depicted in Figure 5. In the motivational scenario, essentially both the dataset provider and the data scientist are only responsible for single nodes in the topology that represent a private dataset and a proprietary algorithm, respectively. The participant-specific and infrastructure-specific nodes are combined by the infrastructure modeler in a complete and deployable application topology, which can afterwards be used by the application owner.

Prior to the beginning of the exchange sequence shown in Figure 1, an application deployment model does not exist yet, hence, one of the participants needs to instantiate it. Since the application owner is the main driver for this collaboration, we assume that the deployment model is first instantiated by the application owner. For the sake of brevity, we omit the discussion on how an empty deployment model is instantiated.

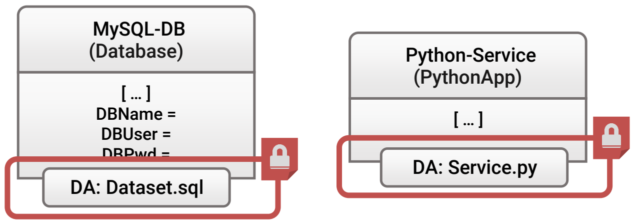


Figure 7. Application topology consisting of two decoupled Node Templates that represent the dataset and algorithm with encryption requirements.

In the *first part* of this collaboration, the dataset provider enriches the deployment model with the dataset-specific information and sends it back to the application owner, who then passes this part of the model to the infrastructure modeler. Since the dataset is private and must only be accessed by the application owner, the dataset provider must provide it in a secured form. Thus, to protect the dataset from unauthorized access, it must be encrypted. In addition, the dataset provider might want to sign the dataset to ease its integrity verification. The MySQL-DB Node Template shown in Figure 5 represents a MySQL database node that is hosted on the MySQL Database Management System (DBMS). In our example, the private dataset is provided as a SQL dump file and is attached as a DA to the node. Note that apart from the DA, the MySQL database node has properties such as a database name or access credentials, some of which might be set by different participants. For instance, a name for the database might be present in the dump file, therefore, the dataset provider might specify this property and, optionally, decide to protect it. We assume that only the artifact corresponding to the dataset has to be protected for the MySQL-DB node as depicted in Figure 7.

In the *second part* of collaboration, the data scientist models and enforces the encryption requirements for the Deployment Artifact of the Node Template, which represents the algorithm, and passes it to the infrastructure modeler. As the application owner is the only authorized user of the algorithm, the encryption requirements are similar to the ones of the dataset. At this point, the application topology consists of two decoupled Node Templates, namely *MySQL-DB* and *Python-Service* as shown in Figure 7. Since both steps are about enforcing encryption requirements for corresponding Deployment Artifacts, the process of modeling and enforcing these requirements is similar. A participant, i.e., the dataset provider or the data scientist, needs to: (i) create a TOSCA Policy Template of type “Encryption Policy”, (ii) attach a TOSCA Policy, i.e., an instance of the corresponding Policy Template, to the corresponding Artifact Template, (iii) optionally include other information, and (iv) export the resulting CSAR and pass it to the next participant. As encryption Policy Templates are uniquely associated with symmetric keys, the respective participants must generate keys first and afterwards generate corresponding Policy Templates. Moreover, the rules for sharing keys need to be stored and maintained, e.g., in the form of access control lists. In our example, though, the key sharing rules are trivial, as only the application owner is able to access the private contents of the application’s topology. Simplified examples of Encryption Policy Type and Template are shown in Figure 8. While the Encryption Policy Type only specifies the schema for properties, e.g., a property *keyHash* is of type

```
<PolicyType name="EncryptionPolicyType" ...>
  <PropertiesDefinition>
    <properties>
      <key>keyHash</key>
      <type>xsd:string</type>
    </properties>
    <properties>
      <key>algorithm</key>
      <type>xsd:string</type>
    </properties>
    ...
  </PropertiesDefinition>
</PolicyType>

<PolicyTemplate name="4def0020237351e59...“
                type="EncryptionPolicyType"...>
  <Properties>
    <keyHash>4def0020237351e59...</keyHash>
    <algorithm>AES</algorithm>
    ...
  </Properties>
</PolicyTemplate>
```

Figure 8. Simplified example of TOSCA Encryption Policy Type and Template specification in XML.

```
<PolicyType name="SignedPropertiesPolicyType“ ...>
  ...
  <PropertiesDefinition>
    <properties>
      <key>propertyNames</key>
      <type>xsd:string</type>
    </properties>
  </PropertiesDefinition>
</PolicyType>

<PolicyTemplate name=“vSphereSignedProperties“
                type="SignedPropertiesPolicyType“...>
  <Properties>
    <propertyNames>IP User Pwd</propertyNames>
  </Properties>
</PolicyTemplate>
```

Figure 9. Example of TOSCA Signing Grouping Policy Type and its Template specification for vSphere-Hypervisor Node Template’s properties in XML.

xsd:string, the corresponding Policy Template references an actual key. Therefore, the properties in the Policy Template shown in Figure 8 contain the details about the key, i.e., its hash value and the algorithm name. Depending on the algorithm used and the desired level of details, the number of properties might be increased, e.g., to include the key size in bits. Moreover, the attachment of policies to respective Artifact Templates looks similar to the example shown in Figure 6. The modeled encryption requirements are enforced at export time, which results in encryption of the Artifact Template’s files using the referenced key and changing the value of the corresponding policy’s *applied* attribute to *true*.

In the *final part* of the collaboration scenario, the infrastructure modeler enriches the application topology with infrastructure-related nodes and sets up all the necessary properties required for deploying this application in an automated way. One important point here is that the model can be further refined in case additional requirements arise afterwards. This means that any participant might need to add or remove something

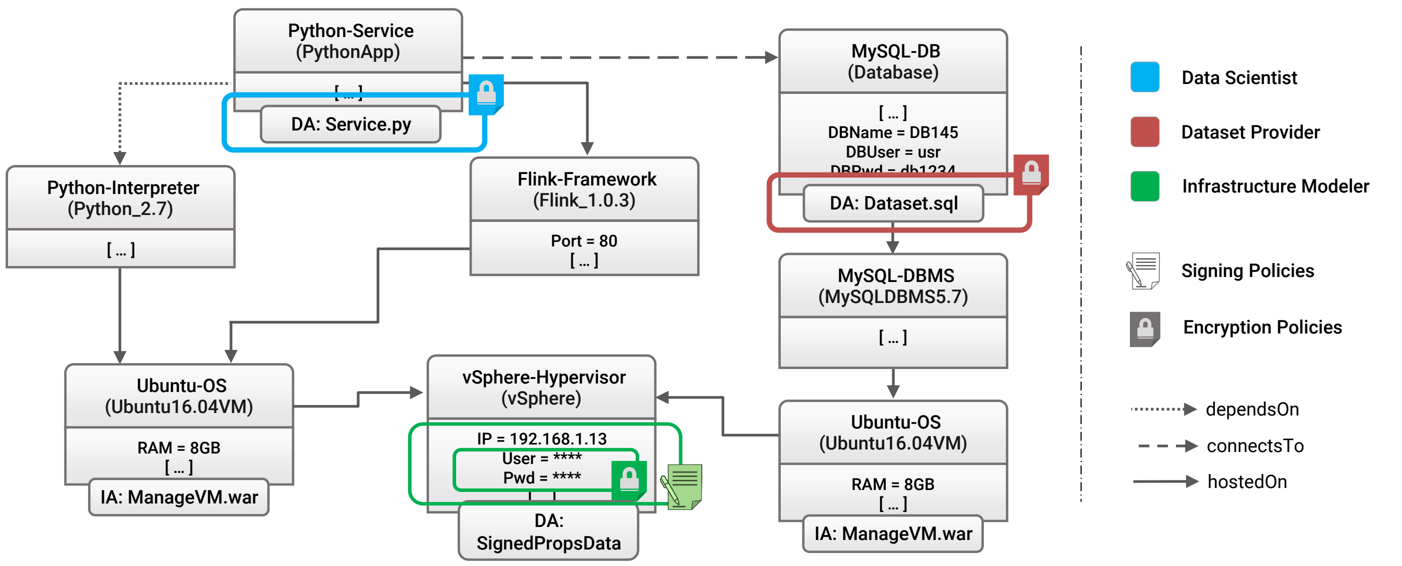


Figure 10. A complete deployment model with enforced security requirements.

from the model *after* the original exchange sequence. Therefore, all sensitive information, such as access credentials or endpoints, might require to be encrypted and signed by the infrastructure modeler to allow the application owner to ensure that the application will be deployed according to the SLAs.

To demonstrate how encryption and signing requirements can be combined, we assume that the infrastructure modeler must encrypt and sign the properties for the vSphere-Hypervisor Node Template, thus, ensuring that the application will be deployed to the correct location. For instance, the user credentials data must be encrypted and both, user credentials and IP address information need to be signed. With these requirements, the resulting set of policies related to the vSphere-Hypervisor Node Template consists of: (i) encryption policy, (ii) encryption grouping policy, (iii) signing policy, and (iv) signing grouping policy. This set of policies, as discussed in Section V-B, is attached to the vSphere Node Type to guarantee that all templates of this type will have these requirements enforced. The attachment of policies happens similar to the example Node Type policy specification snippet shown in Figure 6. Two attached policies represent respective keys, namely encryption and signing policies. The former is linked with a symmetric key used for encryption, and the latter is linked with a key-pair. The purpose of grouping policies is to combine sets of properties targeted for the same action, i.e., encryption or signing. Figure 9 depicts a simple specification of a signing grouping Policy Type and the corresponding Policy Template for grouping the desired properties of vSphere-Hypervisor Node Template. Chosen property names are listed in the form of a space-separated list; therefore, the Policy Type contains only one property of type *xsd:string*. The Policy Template instantiates this property with a list of property names related to the vSphere-Hypervisor Node Template. Note that separate grouping policies are specified for encryption and signing requirements to allow distinguishing them and looking for overlaps. In cases when property lists overlap, the signing operations need to be performed twice, before and after encryption. This allows keeping the information about plain properties' hash values. This is one technical nuance that can simplify verification of properties at export time for parties authorized to access their values, since values can be checked prior to encryption. After enforcing modeled requirements, the infrastructure modeler possesses the complete version of the deployment model, which is now ready to be passed to the application owner. Figure 10 depicts the complete deployment model with enforced encryption and signing requirements from all involved participants. Generated signature files related to properties are attached to the model to keep it self-contained. Files that are required to be encrypted keep their original names, but are not accessible anymore without being decrypted.

VI. Prototype

In this section we describe the prototypical implementation of the presented concepts. The prototype is implemented during the course of SePiA.Pro [44], a project that tackles the issues of optimizing industrial automation processes in the context of Industry 4.0. Our prototype is based on the OpenTOSCA ecosystem, an open source toolchain for development and execution of TOSCA-compliant cloud applications. The OpenTOSCA ecosystem consists of such tools as Winery [45], [46], OpenTOSCA Container [15], and Vinothek [47]. Winery is a TOSCA-compliant modeling environment that supports graphical and text-based modeling of deployment models and offers additional functionalities including, e.g., GUI-based plan modeling capabilities. OpenTOSCA Container is an execution environment for TOSCA-based deployment models, which offers a rich set of functionalities including, e.g., deployment model execution, and a GUI for managing and monitoring the application instances that is based on Vinothek concepts.

A. *Overview*

Winery is the core part for implementation of the presented concepts, as most of them are coupled with the modeling process. Winery is a feature-rich modeling environment for TOSCA-compliant applications. The prototypical implementation adds extensions to both the backend and frontend of Winery, does not require further configuration, and can be used as an integral part of it. It is open source and available via Github [48]. As discussed in Section IV, in our approach every modeler is required to use a local Winery instance due to the absence of a centralized environment. Since keys are used for the enforcement of policies, Winery is extended to support key management functionalities. This includes storing, deletion, and generation of symmetric and asymmetric keys. For key storage we rely on the usage of Java's Java Cryptography Extension KeyStore (JCEKS) for storing all imported keys together. Assuming that Winery runs in a local and secure environment of a distinct party, registering keys in it is not problematic since keys never leave the modeler's environment. This approach, however, has to be extended to support multiple-owner Winery instances. Corresponding policies are generated based on the selected keys. For key distribution, a partner-wise specification of access control lists for security policies is added to Winery. Every participant needs to maintain a list of partner-specific rules negotiated by means of agreements in collaborations. Therefore, whenever a key is requested by some party, the key access rights are defined based on the local rules in Winery. All functionalities are accessible via the corresponding REST over HTTP endpoints.

The resulting prototype supports modeling of security requirements using Winery's built-in XML editors for the corresponding TOSCA entities. Winery stores the modeled TOSCA entities in a decoupled manner, which makes a concept of CSAR important only at export or import time. More specifically, at import time CSARs are disassembled into distinct entities to prevent storing duplicates. In a similar fashion, at export time CSARs are assembled from all the entities that are included in the chosen Service Template. This results in an issue that TOSCA meta files are not explicitly stored and are generated on-the-fly. As described in Section IV, the enforcement of modeled security policies at export time for selected TOSCA entities, e.g., Service Templates or Artifact Templates, happens in case specified keys are present in the system. Signatures for files in Artifact Templates are generated, and then referenced as additional files in the same Artifact Templates. If the files of Artifact Templates are subject to both, encryption and signing requirements, then the signatures of plain and encrypted files are attached. This allows verifying the integrity of target files to any involved participant independently of whether the files are encrypted or not. Signatures for properties are grouped as a separate Artifact Template of type "Signature", which is attached to the respective Node Template. This way of referencing newly-generated files ensures the self-containment property of deployment models. In addition, if policies were applied, the corresponding attribute is set to signify this fact. After encryption and signing requirements are enforced, an external signature of a CSAR is generated using a so-called *master key*, which is specified by the modeler for the whole environment as discussed in Section IV. The corresponding certificate or chain of certificates for this external signature is embedded into the CSAR and is used for verification at import time. The external signature is verified first at import time and is not stored if verification succeeds, since the CSAR is decomposed into distinct separately-stored entities. Furthermore, the import is aborted in case if the integrity check was not successful. In situations when keys requested by a modeler were provided, they can be imported and used for decryption of entities. Finally, only the participant who has an entire set of keys is able to decrypt and deploy the final application. Afterwards, the deployment and execution in the OpenTOSCA Container happens in a regular manner, since the CSAR contains the original deployment model.

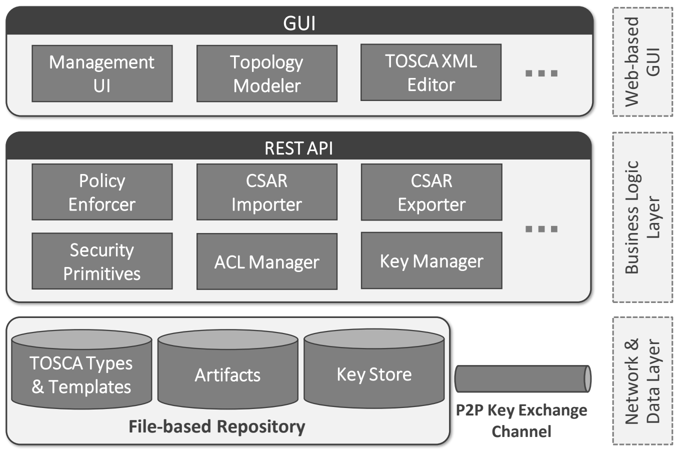


Figure 11. Architectural overview of the prototypical implementation.

To provide readers with a better view, in the following subsection we discuss an architecture of the prototype and explain several specific details related to the described concepts.

B. System Architecture

Figure 11 shows an architectural overview of Winery focusing on the components relevant to the prototype. At the topmost layer, Winery provides a Web-based graphical user interface (GUI) that consists of multiple sub-modules among which the following are of interest to the prototype: (i) The *Topology Modeler* is a sub-module that allows users to visually design various nodes and their interconnections as parts of a desired TOSCA deployment model. Using this component one is able to instantiate Node Types into Node Templates and populate their properties appropriately. Furthermore, this sub-module allows attaching various artifacts to these nodes and form a topology out of them using suitable Relationship Templates, all with simple UI operations such as Drag and Drop (c.f. Section V-A). On the other hand, (ii) the *TOSCA XML Editor* allows manually altering declarations of various TOSCA constructs. This editor is used, for example, to define the necessary Encryption and Signing Policy Types and attach instances of them to Node Types and Artifact Templates (c.f Section V-C). Finally, (iii) the *Management UI* allows users, among other things, to request the export of CSARs, which would enforce the corresponding policies, and the import of CSARs, which would trigger the process of verifying signed entities against the embedded certificate chains.

The GUI is merely an interface for the actual operations residing in the Winery backend, which are accessible via a REST API. In Figure 11, these operations are grouped into the Business Logic Layer, which contains an extensive set of sub-modules: (i) The *CSAR Exporter* is responsible for assembling the resulting portable CSAR out of the corresponding TOSCA templates and the artifacts attached to them. It is also responsible for enforcing the policies attached to these constructs. To this end, the CSAR Exporter utilizes another sub-module, namely (ii) the *Policy Enforcer*, which applies policies by

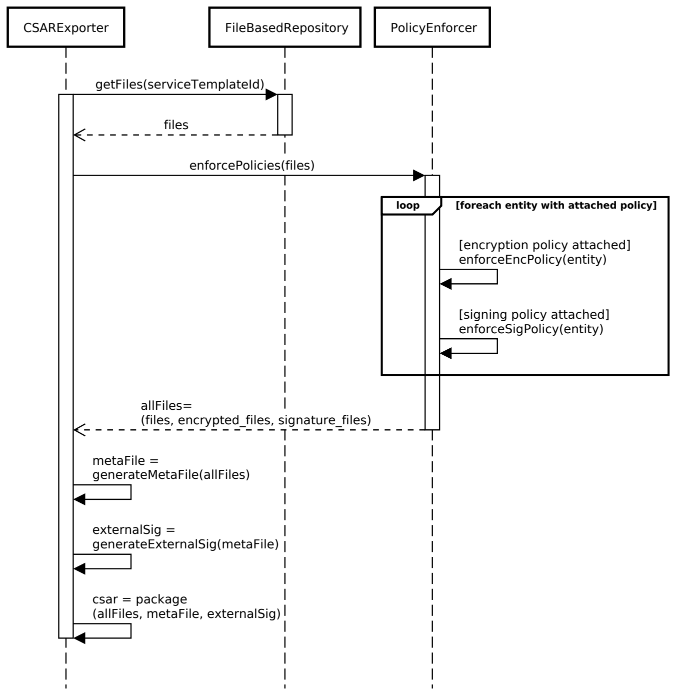

CSARExporter
FileBasedRepository
PolicyEnforcer
getFiles(serviceTemplateId)
files
enforcePolicies(files)
loop
[foreach entity with attached policy]
[encryption policy attached]
enforceEncPolicy(entity)
[signing policy attached]
enforceSigPolicy(entity)
allFiles=
(files, encrypted_files, signature_files)
metaFile =
generateMetaFile(allFiles)
externalSig =
generateExternalSig(metaFile)
csar = package
(allFiles, metaFile, externalSig)

Figure 12. A simplified process of securely exporting a CSAR.

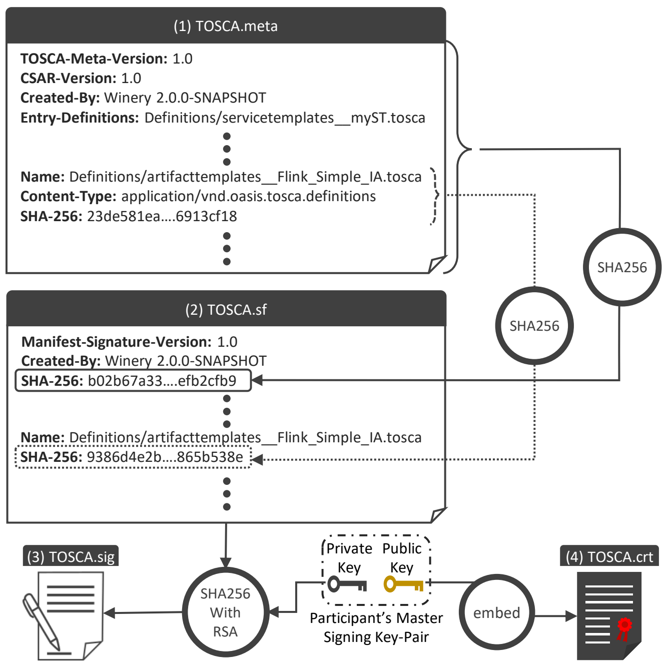

(1) TOSCA.meta
TOSCA-Meta-Version: 1.0
CSAR-Version: 1.0
Created-By: Winery 2.0.0-SNAPSHOT
Entry-Definitions: Definitions/servicetemplates__myST.tosca
Name: Definitions/artifacttemplates__Flink_Simple_IA.tosca
Content-Type: application/vnd.oasis.tosca.definitions
SHA-256: 23de581ea....6913cf18
(2) TOSCA.sf
Manifest-Signature-Version: 1.0
Created-By: Winery 2.0.0-SNAPSHOT
SHA-256: b02b67a33....efb2cfb9
Name: Definitions/artifacttemplates__Flink_Simple_IA.tosca
SHA-256: 9386d4e2b....865b538e
SHA256
SHA256
(3) TOSCA.sig
SHA256 With RSA
Private Key
Public Key
Participant's Master Signing Key-Pair
embed
(4) TOSCA.crt

Figure 13. The process of generating the external CSAR signature files at export time.

encrypting the properties and artifacts that have an Encryption Policy attached, and generating a signature for them if a Signing Policy is attached. On the other hand, the provisioning of cryptographic keys used for encryption and signing is done by (iii) the *Key Manager*. Moreover, (iv) the *CSAR Importer* is responsible for disassembling a CSAR file and importing its constituent components into the local repository. It is also responsible for verifying the integrity and authenticity of the imported CSAR by comparing the contents of the signed entities with the associated signatures and only performing the import if no violations are detected. As a supporter for the other components, (v) the *Security Primitives* sub-module utilizes the Bouncy Castle Crypto APIs [49] to provide various necessary cryptographic operations, such as symmetric and asymmetric encryption, key and certificate generation and content signing and verification. Finally, as mentioned earlier, (vi) the *Access Control List (ACL) Manager* specifies the rules that govern the relationships the local participant has with the other participants. This is necessary to determine who is authorized to obtain secret keys used to encrypt sensitive parts of the deployment model.

As discussed earlier, the actual exchange of keys happens at a lower level using a secure *P2P Channel*, which is not part of the actual prototype. Besides key exchange, the bottom layer of Figure 11 is responsible for the permanent storage of the modeled and imported TOSCA entities, as well as the associated implementation and deployment artifacts. The JCE KeyStore also resides at this layer. Storage of these objects happens using a *File-based Repository*.

In the following subsection, we see how the introduced sub-modules interact together to achieve the operations of exporting and importing a CSAR that utilizes the described TOSCA Policy Extension (c.f. Section V-C).

C. Secure Export of CSARs

As we have discussed earlier, the enforcement of the introduced policies happens during the creation of a CSAR, i.e., the CSAR export operation. Moreover, during this operation the CSAR's external signature is also generated. Figure 12 shows a simplified UML sequence diagram that explains how involved modules from the introduced system architecture interact together in order to perform the export operation: after receiving a request from the *Management UI* (not shown in this figure) the *CSAR Exporter* starts the process via traversing the TOSCA Topology specified by the Service Template and requesting pointers to the corresponding TOSCA definition files, as well as the files associated with them, from the *File-based Repository*. Afterwards, a request is sent to the *Policy Enforcer* to perform the enforcement. It does so by searching the set of TOSCA definition files for Node Types or Artifact Templates with attached policies, then encryption and signing policies are enforced as discussed earlier. To this end, the *Policy Enforcer* utilizes the functionalities provided by the *Security Primitives* sub-module (not shown in the figure). The result of policy enforcement is a changed set of files that has some encrypted content and/or additional signature files. In the next step, the *CSAR Exporter* populates the TOSCA.meta file by traversing the set of files and creating an entry for each of them. Finally, the external signature is generated starting from the TOSCA.meta file and the resulting CSAR file is created by packaging all files in a single compressed archive.

Figure 13 demonstrates the overall procedure of generating the external signature of the CSAR, which starts from processing the TOSCA.meta file and which, as mentioned earlier, follows the JAR signing process. The TOSCA.meta file is used to describe the contents of a CSAR and consists of: (i) a header section, which provides general information about the

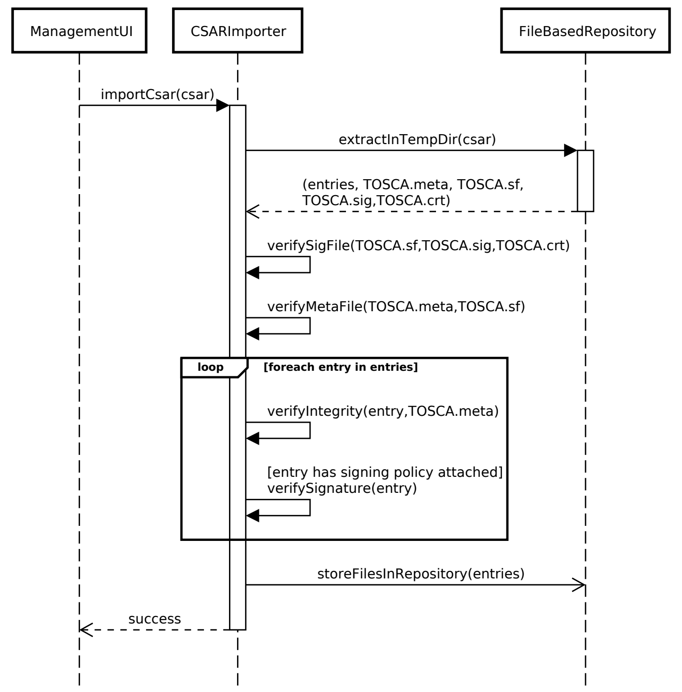

ManagementUI
CSARImporter
FileBasedRepository
importCsar(csar)
extractInTempDir(csar)
(entries, TOSCA.meta, TOSCA.sf, TOSCA.sig,TOSCA.crt)
verifySigFile(TOSCA.sf,TOSCA.sig,TOSCA.crt)
verifyMetaFile(TOSCA.meta,TOSCA.sf)
loop
[foreach entry in entries]
verifyIntegrity(entry,TOSCA.meta)
[entry has signing policy attached] verifySignature(entry)
storeFilesInRepository(entries)
success

Figure 14. A simplified process of securely importing a CSAR showing only the successful control flow.

deployment model such as who created it and the path of the TOSCA definition file associated with the Service Template, and (ii) a body section that lists all files contained in the CSAR along with their exact location and MIME file type. We enhance the body section of the TOSCA.meta file with a digest subsection for each contained file that helps in guaranteeing its integrity. In this prototype, we use the SHA-256 [50] function to calculate digests. In the next step, a signature file (TOSCA.sf) is generated using the TOSCA.meta file. While the purpose of the TOSCA.meta file is to guarantee the integrity of the CSAR, the purpose of TOSCA.sf file is guaranteeing the integrity of the TOSCA.meta file itself. To this end, the TOSCA.sf file has a header section that includes a digest of the whole TOSCA.meta file, and a body section with subsections corresponding to the ones in the TOSCA.meta file. However, in this case each subsection contains a digest of the whole matching subsection in the TOSCA.meta file. Furthermore, to ensure the overall integrity, the TOSCA.sf file is signed using the master signing private key of the participant who issued the CSAR export. As a result, a block signature file (TOSCA.sig) is generated. Finally, a certificate (TOSCA.crt), which contains the master signing public key of the participant, is also included in the archive to provide future importers with the possibility to verify the signature. The participant's master signing key-pair resides, like other keys, in the *Key Store* repository that can be accessed by other components via the *Key Manager* sub-module.

D. Secure Import of CSARs

The import of a protected CSAR into the modeling environment comprises two steps: (i) firstly, it must trigger verifications of CSAR's integrity and authenticity, and afterwards (ii) the regular task of storing constituent TOSCA entities and other artifacts in the local repository must be accomplished. This is depicted in Figure 14: upon a request from the *Management UI* the *CSAR Importer* module begins the process of importing the CSAR by asking the *File-based Repository* to create a temporary directory in which the archive content is extracted. As we have seen in the previous section, in addition to the regular entries, the contents of the CSAR include the TOSCA.meta file as well as the external signature files, i.e., TOSCA.sf, TOSCA.sig, and TOSCA.crt. The importer then performs a check to verify the integrity and authenticity of the signature file (TOSCA.sf) by performing the digital signature verification process using the TOSCA.sig and TOSCA.crt files. Next, the importer verifies the integrity of the TOSCA.meta file by comparing its overall digest as well as the digests of its subsections to the corresponding entries in the TOSCA.sf file. Afterwards, the regular CSAR entries are enumerated, and an integrity check is performed for each one of them by calculating its digest and comparing it to the corresponding digest listed in the TOSCA.meta file. A further authenticity check is performed if the entry is a Node Type, or an Artifact Template with an associated Signing Policy in which case the included signature is verified. If all previous validity checks pass, the contents of the CSAR are imported into the *File-based Repository* and the *Management UI* is notified about the success of the operation. Otherwise, the import is aborted, and a proper error message is returned to the *Management UI*. Similar to what we have seen in the export process, the *Security Primitives* sub-module is utilized by the *CSAR Importer* to perform required cryptographic tasks.

VII. Related Work

The problem of data protection in outsourcing and collaboration scenarios appears in works related to different domains. Multiple works attempt to tackle security-related problems using centralized approaches. Wang et al. [31] present a method for protecting the models in collaborative computer-aided design (CAD), which extends RBAC mechanism by adding notions of scheduling and value-adding activity to roles. Authors propose to selectively share data to prevent reverse engineering. However, no clear description how to enforce the proposed model is given. Cera et al. [51] introduce another RBAC-based data protection approach in collaborative design of 3D CAD models. Models are split into separate parts based on specified role-based security requirements to provide personalized views using a centralized access control mechanism. Li et al. [32] propose a security policy meta-model and the framework for securing big data using sticky policies concept. Policies are loosely-coupled with the data and the framework relies on a trusted party, which combines policy and key management functionalities, and enforces the access control. Huang et al. [52] introduce a set of measures allowing to protect patients data in portable electronic health records (EHRs). Authors propose a centralized system, which combines de-identification, encryption, and digital signatures as means to achieve data privacy. Li et al. [36] describe an approach based on the Attribute-Based Encryption that helps to protect patient's personal health records in the cloud. In this approach, data is encrypted using keys that are generated based on the owner-selected set of attributes and then published to the cloud. Users can only access the data in case they possess corresponding attributes, e.g., profession or organization. More specifically, users are divided into several security domains and the attributes for these domains are managed by corresponding attribute authorities. Decryption keys, therefore, can be generated independently from data owners by the respective attribute authorities. Essentially, the described approaches are domain-specific and rely on a trusted party, which makes it problematic to apply them to the problem of protecting collaborative development of deployment models discussed in Section III.

A number of approaches focus on the data encryption in outsourcing scenarios. Miklau and Suciu [53] introduce an encryption framework for protecting XML data published on the Internet. Contributions of the work include a policy specification language available in the form of queries and a model allowing to encrypt single XML documents. Access control is enforced based on key possession. Vimercati and Foresti [54] discuss fragmentation-based approaches for protecting outsourced relational data. The authors elaborate on several techniques allowing to split up the given data based on some constraints into one or more fragments and store them in a way to protect confidentiality and privacy. For instance, data can be split into two parts and stored on non-communicating servers. Whenever constraints cannot be satisfied for some attributes, the encryption is used. In the follow-up work, Vimercati et al. [55] present a way to enforce selective access control using the cryptography-based policies. Authors propose to use key derivation mechanisms to simplify the distribution of keys.

To the best of our knowledge, there is no approach that successfully tackles our problem of deployment models protection in collaborative application development scenarios. While the described related work provides useful insights on the usage of well-established concepts such as the usage of sticky policies and access control enforcement by means of key possession, it is not applicable to our problem due to several reasons. One of the main issues is that most of the discussed approaches rely on the idea of a trusted party, which can regulate the access control. While it is desirable to have a central authority, in many cases it is unrealistic, leading to a need for peer-to-peer solutions. Moreover, having focus only on separate security requirements like encryption or strong assumptions about the underlying data makes these approaches not suitable for the described problems.

VIII. Conclusion and Future Work

In this work, we showed how security requirements can be modeled and enforced in collaborative development of deployment models. We identified sensitive parts in deployment models and proposed a method, which allows protecting them based on a combination of existing research work. For validation of the presented concepts, we applied them to TOSCA, an existing OASIS standard, which specifies a provider-agnostic cloud modeling language. The resulting prototypical implementation is based on Winery, the modeling environment, which is a part of the OpenTOSCA ecosystem, an open source collection of applications supporting TOSCA.

One issue in our approach that has to be optimized is the way keys are distributed. We rely on the fact that not all participants need to exchange keys, which, however, does not solve the scalability problem. If N keys were used for encryption, eventually all of them will be used in key distribution. For improving the efficiency, the key derivation techniques, e.g., described by Vimercati et al. [55], can be used to reduce the number of keys that need to be exchanged. Another problem for future work is the generalization of the adversary model. Since deployment models can be intentionally corrupted by an adversary, there is a strong need to store the provenance information, which describes deployment model's states at every export with respect to a certain collaboration. Having such provenance information stored in some accessible form makes it possible to track the entire collaboration history with all the deployment model states that were existing throughout the process. For this reason, one might employ a centralized system, which will also simplify the policy enforcement and key distribution processes, or store the provenance in a decentralized fashion, e.g., by utilizing the blockchain technology [56]. In addition, we plan to analyze the existing deployment technologies and identify the optimal mapping constructs for enabling our approach to them. For instance, not every technology allows using policies as explicit modeling constructs, which requires attaching security and privacy requirements using other constructs, e.g., by means of annotations or descriptors. Moreover, the modeling-related tooling varies for different deployment technologies, which imposes additional hurdles on enforcing modeled requirements for the chosen technology.

There are several issues in our prototype that can be optimized. Currently, the grouping policies in the prototype cannot be linked with the corresponding encryption or signing policies, hence it is not possible to model several groups of properties signed or encrypted using different keys. In future work, we plan to extend the implementation to support more complex scenarios and increase the overall usability of the modeling process using Winery and its respective components. Finally, there is a pitfall for cases when files are modeled in the form of references, e.g., if they reside on a remote server. Encrypting and signing such files completely changes the verification semantics as only the references are checked. This is not safe since the actual content behind the reference can be changed multiple times by the data owner without changing the reference itself. Moreover, the usage of references invalidates the self-containment property of deployment models. In future work, referenced files need to be materialized at export time, which solves this problem and preserves deployment models in a self-contained state.

In our previous work [7], we tackled the issue of guaranteeing accountability of collaborative development of deployment models by registering fingerprints of the various states a deployment model goes through while being developed in the blockchain. These fingerprints serve as a guarantee for the integrity and authenticity of the model when being passed from one participant to the next. Furthermore, we stored the actual contents of these models in a decentralized file storage addressable using the aforementioned fingerprints. This creates an immutable history of verifiable deployment model states. A problem with that approach is the public nature of the decentralized storage where the contents of deployment models are stored. This makes the sensitive information that might exist within these models accessible to the public. The obvious solution to this is encryption, thus in a future work, we plan to combine both approaches in order to have a collaboration process, which guarantees both security and accountability.

Acknowledgment

This work is partially funded by the BMWi project *SePiA.Pro* (01MD16013F) and the European Union's Horizon 2020 research and innovation project *RADON* (825040).

References

[1] V. Yussupov, M. Falkenthal, O. Kopp, F. Leymann, and M. Zimmermann, "Secure Collaborative Development of Cloud Application Deployment Models," in Proceedings of The 12th International Conference on Emerging Security Information, Systems and Technologies (SECURWARE 2018). Xpert Publishing Services, September 2018, pp. 48–57.

[2] M. Hermann, T. Pentek, and B. Otto, "Design principles for industrie 4.0 scenarios," in 2016 49th Hawaii International Conference on System Sciences (HICSS). IEEE, 2016, pp. 3928–3937.

[3] P. M. Mell and T. Grance, "Sp 800-145. the NIST definition of cloud computing," Gaithersburg, MD, United States, Tech. Rep., 2011.

[4] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," Computer networks, vol. 54, no. 15, 2010, pp. 2787–2805.

[5] M. Zimmermann, U. Breitenbücher, M. Falkenthal, F. Leymann, and K. Saatkamp, "Standards-based function shipping – how to use tosca for shipping and executing data analytics software in remote manufacturing environments," in Proceedings of the 2017 IEEE 21st International Enterprise Distributed Object Computing Conference (EDOC 2017). IEEE Computer Society, 2017, pp. 50–60.

[6] U. Breitenbücher et al., "Combining declarative and imperative cloud application provisioning based on tosca," in Proceedings of the IEEE International Conference on Cloud Engineering (IEEE IC2E 2014). IEEE Computer Society, March 2014, pp. 87–96.

[7] G. Falazi, U. Breitenbücher, M. Falkenthal, L. Harzenetter, F. Leymann, and V. Yussupov, "Blockchain-based Collaborative Development of Application Deployment Models," in On the Move to Meaningful Internet Systems. OTM 2018 Conferences (CoopIS 2018), ser. Lecture Notes in Computer Science, vol. 11229. Springer, 2018, pp. 40–60.

[8] T. Kvan, "Collaborative design: what is it?" Automation in construction, vol. 9, no. 4, 2000, pp. 409–415.

[9] G. Karjoth, M. Schunter, and M. Waidner, "Platform for enterprise privacy practices: Privacy-enabled management of customer data," in International Workshop on Privacy Enhancing Technologies. Springer, 2002, pp. 69–84.

[10] W. Bagga and R. Molva, "Policy-based cryptography and applications," in International Conference on Financial Cryptography and Data Security. Springer, 2005, pp. 72–87.

[11] A. Harrington and C. Jensen, "Cryptographic access control in a distributed file system," in Proceedings of the 8th ACM symposium on Access control models and technologies. ACM, 2003, pp. 158–165.

[12] OASIS, Topology and Orchestration Specification for Cloud Applications (TOSCA) Version 1.0, Organization for the Advancement of Structured Information Standards (OASIS), 2013.

[13] OASIS, Topology and Orchestration Specification for Cloud Applications (TOSCA) Primer Version 1.0, Organization for the Advancement of Structured Information Standards (OASIS), 2013.

[14] A. Bergmayr et al., "A systematic review of cloud modeling languages," ACM Comput. Surv., vol. 51, no. 1, Feb. 2018, pp. 22:1–22:38.

[15] T. Binz et al., "Opentosca – a runtime for tosca-based cloud applications," in Service-Oriented Computing. Berlin, Heidelberg: Springer, 2013, pp. 692–695.

[16] C. Endres et al., "Declarative vs. imperative: Two modeling patterns for the automated deployment of applications," in Proceedings of the 9th International Conference on Pervasive Patterns and Applications. Xpert Publishing Services (XPS), Feb. 2017, pp. 22–27.

[17] U. Breitenbücher, K. Képes, F. Leymann, and M. Wurster, "Declarative vs. imperative: How to model the automated deployment of iot applications?" in Proceedings of the 11th Advanced Summer School on Service Oriented Computing. IBM Research Division, Nov. 2017, pp. 18–27.

[18] Chef. [Online]. Available: https://www.chef.io/ [retrieved: July, 2018]

[19] Juju. [Online]. Available: https://jujucharms.com/ [retrieved: July, 2018]

[20] OASIS, Web Services Business Process Execution Language (WS-BPEL) Version 2.0, Organization for the Advancement of Structured Information Standards (OASIS), 2007.

[21] OMG, Business Process Model and Notation (BPMN) Version 2.0, Object Management Group (OMG), 2011.

[22] R. Boutaba and I. Aib, "Policy-based management: A historical perspective," Journal of Network and Systems Management, vol. 15, no. 4, Dec 2007, pp. 447–480.

[23] R. Wies, "Using a classification of management policies for policy specification and policy transformation," in Integrated Network Management IV. Springer, 1995, pp. 44–56.

[24] P. Samarati and S. C. di Vimercati, "Access control: Policies, models, and mechanisms," in International School on Foundations of Security Analysis and Design. Springer, 2000, pp. 137–196.

[25] S. Pearson and M. Casassa-Mont, "Sticky policies: An approach for managing privacy across multiple parties," Computer, vol. 44, no. 9, 2011, pp. 60–68.

[26] Q. Tang, On Using Encryption Techniques to Enhance Sticky Policies Enforcement, ser. CTIT Technical Report Series. Netherlands: Centre for Telematics and Information Technology (CTIT), 2008, no. WoTUG-31/TR-CTIT-08-64.

[27] B. W. Lampson, "Protection," ACM SIGOPS Operating Systems Review, vol. 8, no. 1, 1974, pp. 18–24.

[28] M. Falkenthal et al., "Opentosca for the 4th industrial revolution: Automating the provisioning of analytics tools based on apache flink," in Proceedings of the 6th International Conference on the Internet of Things, ser. IoT'16. New York, NY, USA: ACM, 2016, pp. 179–180.

[29] T. Binz, U. Breitenbücher, O. Kopp, and F. Leymann, TOSCA: Portable Automated Deployment and Management of Cloud Applications. New York, NY: Springer New York, 2014, pp. 527–549.

[30] M. Zimmermann, F. W. Baumann, M. Falkenthal, F. Leymann, and U. Odefey, "Automating the provisioning and integration of analytics tools with data resources in industrial environments using opentosca," in Proceedings of the 2017 IEEE 21st International Enterprise Distributed Object Computing Conference Workshops and Demonstrations (EDOCW 2017). IEEE Computer Society, Oct. 2017, pp. 3–7.

[31] Y. Wang, P. N. Ajoku, J. C. Brustoloni, and B. O. Nnaji, "Intellectual property protection in collaborative design through lean information modeling and sharing," Journal of computing and information science in engineering, vol. 6, no. 2, 2006, pp. 149–159.

[32] S. Li, T. Zhang, J. Gao, and Y. Park, "A sticky policy framework for big data security," in 2015 IEEE First International Conference on Big Data Computing Service and Applications (BigDataService). IEEE, 2015, pp. 130–137.

[33] T. Waizenegger et al., "Policy4TOSCA: A Policy-Aware Cloud Service Provisioning Approach to Enable Secure Cloud Computing," in On the Move to Meaningful Internet Systems: OTM 2013 Conferences. Springer, Sep. 2013, pp. 360–376.

[34] U. Breitenbücher, T. Binz, O. Kopp, F. Leymann, and M. Wieland, "Policy-aware provisioning of cloud applications," in Proceedings of the 7th International Conference on Emerging Security Information, Systems and Technologies (SECURWARE). Xpert Publishing Services (XPS), 2013, pp. 86–95.

[35] A. A. E. Kalam et al., "Organization based access control," in IEEE 4th International Workshop on Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE, 2003, pp. 120–131.

[36] M. Li, S. Yu, K. Ren, and W. Lou, "Securing personal health records in cloud computing: Patient-centric and fine-grained data access control in multi-owner settings," in International conference on security and privacy in communication systems. Springer, 2010, pp. 89–106.

[37] F. Li, B. Luo, and P. Liu, "Secure information aggregation for smart grids using homomorphic encryption," in 2010 First IEEE International Conference on Smart Grid Communications (SmartGridComm). IEEE, 2010, pp. 327–332.

[38] S. Ruj and A. Nayak, "A decentralized security framework for data aggregation and access control in smart grids," IEEE transactions on smart grid, vol. 4, no. 1, 2013, pp. 196–205.

[39] Oracle. Understanding signing and verification. [Online]. Available: https://docs.oracle.com/javase/tutorial/deployment/jar/intro.html [retrieved: July, 2018]

[40] J. Huang and M. S. Fox, "An ontology of trust: formal semantics and transitivity," in Proceedings of the 8th international conference on electronic commerce: The new e-commerce: innovations for conquering current barriers, obstacles and limitations to conducting successful business on the internet. ACM, 2006, pp. 259–270.

[41] M. Falkenthal et al., “Requirements and Enforcement Points for Policies in Industrial Data Sharing Scenarios,” in Proceedings of the 11th Advanced Summer School on Service Oriented Computing. IBM Research Division, 2017, pp. 28–40.

[42] F. W. Baumann, U. Breitenbücher, M. Falkenthal, G. Grünert, and S. Hudert, “Industrial data sharing with data access policy,” in Cooperative Design, Visualization, and Engineering. Springer International Publishing, 2017, pp. 215–219.

[43] M. Cooper et al. Internet X.509 Public Key Infrastructure: Certification Path Building. [Online]. Available: https://tools.ietf.org/html/rfc4158 [retrieved: July, 2018]

[44] SePiA.Pro. [Online]. Available: http://projekt-sepiapro.de/en/ [retrieved: May, 2019]

[45] O. Kopp, T. Binz, U. Breitenbücher, and F. Leymann, “Winery – A Modeling Tool for TOSCA-based Cloud Applications,” in Proceedings of the 11th International Conference on Service-Oriented Computing (ICSOC 2013). Springer, Dec. 2013, pp. 700–704.

[46] Winery. [Online]. Available: https://eclipse.github.io/winery/ [retrieved: July, 2018]

[47] U. Breitenbücher, T. Binz, O. Kopp, and F. Leymann, “Vinothek – a self-service portal for tosca,” in Proceedings of the 6th Central-European Workshop on Services and their Composition (ZEUS 2014). CEUR-WS.org, Feb. 2014, Demonstration, pp. 69–72.

[48] Prototypical implementation of the secure csar concepts. [Online]. Available: https://github.com/OpenTOSCA/winery/releases/tag/paper%2Fvy-secure-csar [retrieved: July, 2018]

[49] Bouncy Castle Crypto APIs. [Online]. Available: http://bouncycastle.org/ [retrieved: Dec., 2018]

[50] W. Penard and T. van Werkhoven, On the Secure Hash Algorithm family, 2008, ch. 1, pp. 1–18.

[51] C. D. Cera, T. Kim, J. Han, and W. C. Regli, “Role-based viewing envelopes for information protection in collaborative modeling,” Computer-Aided Design, vol. 36, no. 9, 2004, pp. 873–886.

[52] L.-C. Huang, H.-C. Chu, C.-Y. Lien, C.-H. Hsiao, and T. Kao, “Privacy preservation and information security protection for patients’ portable electronic health records,” Computers in Biology and Medicine, vol. 39, no. 9, 2009, pp. 743–750.

[53] G. Miklau and D. Suciu, “Controlling access to published data using cryptography,” in Proceedings of the 29th international conference on Very large data bases-Volume 29. VLDB Endowment, 2003, pp. 898–909.

[54] S. D. C. di Vimercati and S. Foresti, “Privacy of outsourced data,” in IFIP PrimeLife International Summer School on Privacy and Identity Management for Life. Springer, 2009, pp. 174–187.

[55] S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, “Encryption policies for regulating access to outsourced data,” ACM Transactions on Database Systems (TODS), vol. 35, no. 2, 2010, p. 12.

[56] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. [Online]. Available: http://bitcoin.org/bitcoin.pdf [retrieved: July, 2018]

Synthesis of Formal Specifications From Requirements for Refinement-based Real Time Object Code Verification

Eman M. Al-Qtiemat*, Sudarshan K. Srinivasan*, Zeyad A. Al-Odat *, Mohana Asha Latha Dubasi*, Sana Shuja†
*Electrical and Computer Engineering, North Dakota State University,
Fargo, ND, USA
†Department of Electrical Engineering, COMSATS University,
Islamabad, Pakistan
Emails: *eman.alqtiemat@ndsu.edu, *sudarshan.srinivasan@ndsu.edu, *zeyad.alodat@ndsu.edu, *dubasi.asha@gmail.com,
†SanaShuja@comsats.edu.pk

Abstract—**Formal verification methods have been shown to be very effective in finding corner case bugs and ensuring safety of embedded software systems. The use of formal verification requires a specification, which is typically a high-level mathematical model that defines the correct behavior of the system to be verified. However, embedded software requirements are typically described in natural language. Transforming these requirements to formal specifications is currently a big gap. While there is some work in this area, this paper proposes solutions to address this gap in the context of refinement-based verification, a class of formal methods that have shown to be effective for embedded object code verification. The proposed approach also addresses both functional and timing requirements and has been demonstrated in the context of safety requirements for software control of infusion pumps.**

Keywords–requirements analysis; safety-critical IoT embedded devices; timing specifications; timing transition systems; formal model; formal verification.

I. Introduction

Ensuring the correctness of control software used in safety-critical embedded devices is still an ongoing challenge [1]. For example, from 2001 to 2017, the Food and Drug Administration (FDA) has issued 54 Class-1 recalls on infusion pumps (medical devices used to deliver controlled doses of fluid medications to patients intravenously) due to software issues [2]. Class-1 recalls are applied to medical device models whose use can cause serious adverse health consequences or death. With the advent of IoT, such safety-critical embedded devices incorporate a whole slew of additional functionality to interface with the network and other components, in addition to their core control functions. These additional functions significantly exacerbate the challenge of ensuring that the core functionality of the control software is correct and intact.

Critical devices such as insulin pump still have safeness issues which need valuable software amendments to assure the reliability on design level, this can be handled by either appending new safety insurance specifications to fix existing hazards, or modifying some defined specifications that cause faulty behaviours. Since critical devices are considered as real time systems, most of their specifications have well defined timing constraints must be met else wise the system will fail. This paper works with both functional and timing specifications (called functional and timing requirements), they are basically written in natural language and need to be transformed into a formal model, then it can be tested using a formal verification method. The use of formal verification has become an industry standard when addressing software correctness of safety-critical devices. There are many success stories and commercial adoption of formal verification processes. Examples include Intel [3], Microsoft [4] and [5], and Airbus [6].

Refinement-based verification [7] is a formal verification technology that has been demonstrated to be applicable to the verification of embedded control software at the object-code level [8]. In formal verification and refinement-based verification, typically the design artifact to be verified is called the implementation and the specification is a formal model that captures the correct functionality of the implementation. The goal of refinement-based verification is to mathematically prove that the implementation behaves correctly as defined by the specification. In refinement-based verification, both the implementation and specification are modeled as transition systems and timed transition systems if timing specifications are existed.

One of the key features of refinement-based is the use of refinement maps, which are functions that map implementation states to specification states. In practice, these refinement maps have a very favorable property in that they abstract out behaviors of the implementation not relevant to the specification, but only after determining that these additional behaviors do not actually impact the behaviors of the implementation relevant to the specification. This property of refinement maps makes the refinement-based verification very suitable for the verification of control software used in IoT devices as refinement maps can be used to abstract out the additional functionality of software in IoT devices; again, only after determining that these additional functionality are not impacting the behavior of the core functionality of the implementation as defined by the specification.

One of the crucial challenges in applying refinement-based verification to commercial devices is the availability of formal specifications. For commercial devices, typically, the specification of a device is given as natural language requirements. There are many approaches towards transforming natural language requirements to formal specifications, however none targeted towards refinement-based verification. In this paper, we present methodologies for transforming natural language requirements (both functional and timing) into formal specifications that can be used in the context of refinement-based verification.

The rest of the paper is organized as follows. An overview of the background is presented in Section II. Section III details the related work. A formal model describing the synthesis procedure of functional requirements is presented in Section IV, while Section V presents a different formal model describing the synthesis procedure of timing requirements. Section VI details the case study. Section VII gives the verification results for the proposed formal model. Conclusions and direction for future work are noted in Section VIII.

II. Background

This section explores the parsing tree, the definition of transition systems and the definition of timed transition systems as key terms related to our work.

A. Parsing tree

A parse tree is an ordered tree that pictorially represents how words in a sentence are connected to each other. The connection between each word in the sentence gives the *syntactic categories* for the sentence. The parsing process represents the syntactic analysis of a sentence in natural language. For example, when the parsing process is applied on a simple sentence like ”Adam eats banana”, the parse tree categorizes the two parts of speech: N for nouns (Adam, banana) and V for the verb (eats). Here N, V are the syntactic categories. The parsing process is considered to be a preprocessing step for some applications, where natural language should be converted into other forms. Usually, the system requirements are written in natural language, which needs to be converted into a structural form that can then be used to create the transition system(s) (explained in Section II-B). Enju [9] is an English consistency-based parser, which can process very long complex sentences like system requirements using an accurate analysis (the accuracy relation is around 90 percent of news articles and bio-medical papers). Besides, Enju is a high-speed parser with less than 500 msec per sentence. The output is the resulting tree in an XML format which is considered to be one of the commonly used formats by various applications. As will be described later, the case study used to describe the proposed methodology is from the bio-medical area, Enju was the perfect tool as the natural language processing (NLP) parser.

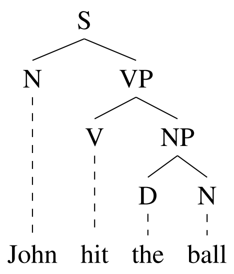


Figure 1. A simple example of a parsing tree using Enju parser [10].

Figure 1 shows a simple tree example using Enju. Here, Enju distinguishes between terminal nodes (John is a terminal node) and non-terminal nodes (VP is a verb phrase). The abbreviations of the syntactic categories of Figure 1 are: S stands for sentence (the head of the tree), N stands for noun, VP stands for verb phrase (which is a subtree), NP stands for noun phrase, V stands for verb, and finally D stands for determiner (comes with noun phrases). Using these syntactic categories, we have developed an extraction technique that would help in translating the natural language to a formal model of the requirements.

B. Transition systems

The implementation and specification in refinement-based verification are represented using Transition Systems (TSs) [7], [8]. The definition of a TS is given below:

Definition 1: A TS $M = \langle S, R, L\rangle$ is a three tuple in which S denotes the set of states, $R \subseteq SXS$ is the transition relation that provides the transition between states, and L is a labeling function that describes what is visible at each state.

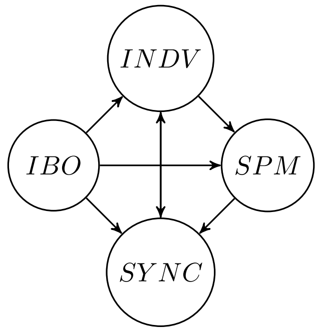


Figure 2. An example of a transition system (TS).

An Atomic Proposition (AP) is a statement that can be evaluated to be either true or false. The labeling function maps state to the APs that are true in every state. An example of a TS is shown in Figure 2. Here S = {IBO, SPM, SYNC, INDV}, R = {(IBO, SPM), (SPM, SYNC), (SYNC, INDV), (INDV, SYNC), (INDV, SPM), (IBO, INDV)} and, $L(SPM)$ represents the atomic propositions that are true for the SPM state. Similarly, labeling function can be applied to all the states in this TS.

C. Timed Transition Systems

Some applications have requirements with timing conditions on the state's transitions called as timing requirements. Timing requirements explain the system behaviour under some timing constraints. Timing constraints are very important especially if we deal with a critical real time systems. As mentioned in the previous section (Sec II-B), transition systems are used to represent the implementation and specification in refinement-based verification, however they do not contain timing requirements. Hence, in the verification of real time systems that contain timing constraints, timed transition systems (TTSs) [8] are used to represent the implementation and specification.

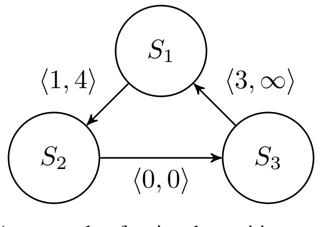


Figure 3. An example of a timed transition system (TTS)

Definition 2: A TTS $M_t = \langle S, R_t, L\rangle$ is a three tuple in which S denotes the set of states and L is a labeling function that describes what is visible at each state. The state transition R_t has the form of $\langle x, y, l_t, u_t\rangle$ where $x, y \in S$ and $l_t, u_t \in N$ represents the lower and upper bounds as the timing condition for the transition.

Figure 3 shows an example of a timed transition system that consists of three states { S1, S2, S3 }, for instance; if the system is in state $S1$ it can go to state $S2$ only between 1 and 4 units of time, while going from $S2$ to $S3$ the time is zero meaning that it should happen immediately, and so on.

III. Related Work

In the last few years, there has been a tremendous growth in finding the optimal technique of requirement transformation into a formal model. While most of them proposed system-driven models, our approach is user-driven to ensure a safe product.

Automatic Requirements Specification Extraction from Natural Language (ARSENAL) [11] is a system based framework that applies some semantic parsers in multi-level to get the grammatical relations between words in the requirement. ARSENAL transforms natural language requirements into formal and logical forms expressed in Symbolic Analysis Laboratory (SAL) (a formal language to describe concurrent systems), and Linear Temporal Logic (LTL) (a mathematical language that describes linear time properties) respectively. The LTL formulas are then used to build the SAL model. Multiple validation checks are applied on Natural Language Processing (NLP) stage and LTL formulas to check for their correctness. However, ARSENAL records some inaccuracies in NLP stage that need a user intervention.

Aceituna et al. [12] have proposed a front end framework that builds a model to exhibit the system behavior (for synchronous systems only) and help in creating temporal logic properties automatically. This framework can be used before applying the model checking technique, it exposes accidental scenarios in the requirements. The framework is designed in a manner that helps in understanding the errors in a non-technical manner for users who do not have a formal background. In contrast, our work does not need the temporal logic in defining the specifications for a model.

A semantic parser has been developed by Harris [13] to extract a formal behavioral description from natural language specifications. The proposed semantic parser was employed to extract key information describing bus transactions. The natural language descriptions are then converted to verilog (a hardware description language) tasks.

Kress-Gazit et al. [14] have proposed a human-robot interface to translate natural language specification into motions. This interface allows a user to instruct the robot using a controller. LTL formulas are employed to formalize the desired behavior requested by the user.

An approach supporting property elucidation (called PROPEL) has been introduced by Smith et al. [15], it provides templates that capture properties for creating property pattern. Natural language and finite state automation are used to represent the templates.

Two approaches have been proposed by Shimizu [16] to solve the ambiguity of natural language specifications using formal specification. The first approach simplifies the formal specification development for the popular PCI bus protocol and the Intel Itanium bus protocol. The second approach explains how formal specifications can help in automating many processes that are now done manually.

A natural language parsing technique has been used with the default reasoning, which is a requirement formalism to support requirement development, this work helps stakeholders to easily deal with requirements in a formal manner, in addition, a method has been proposed for discovering any existed requirements inconsistencies. A prototype tool called CARL was used for implementation and verification by Zowghi et al. [17].

Gervasi et al. [18] have also worked on solving the requirement's inconsistencies issues by using a well-known formalism called monotonic logic, it has been used especially for requirement's transformation. Multiple natural language processing tools [19]–[22] in additional to grammatical analysis methodologies for requirement's development have been done to get requirements in a formal manner.

Bouyer et al. [23] have recently presented a survey on timed automata and how it can be applied for model checking of real-time systems. This survey has summarized the work that has been done since the inception of timed automata in the early 1990s till now. The timing information in real-time models is expressed as temporal logic. However, the survey does not specify gathering timing information from natural language requirements, which has been the focus of our paper.

Knorreck et al. [24] have presented a graphical tool called AVATAR-TEPE (Automated Verification of reAl Time softwARe - TEmporal Property Expression Language), in which the logical and temporal properties are expressed in formal language. This tool can perform all tasks from requirement capture to verification in one language and in one environment. However, the tool requires the knowledge of logical and temporal properties to verify the application. The tool is heavily based on property modeling.

A standardized testing method for distributed real-time cyber-physical systems (CPS) has been proposed by Shrivastava et al. [25]. Temporal properties have been used to express the timing constraints. Peters et al. [26] have proposed a new language that considers timing requirement and checks for errors in the description of the timing constraints. Kang et al. [27] have presented a model-driven approach to verify the timing requirements for automotive systems at the design level. However, in all these works, gathering the timing constraints from natural language requirements, which has been the focus of this paper, has not been addressed.

Carvalho et al. [28] have proposed a symbolic model for translating natural language requirements to a formal model which consider time. Model-based testing techniques are then applied to these formal models. Hassine [29] has presented a formal framework to describe, simulate and analyze real-time systems. This framework considers timing requirements. However, this proposed framework is yet to be applied on large scale industrial projects. In these works, even though the timing requirements are considered, none of the these works are targeted at refinement based verification.

The main advantages of our work over prior algorithms in requirements engineering is its ability to generate a full formal model directly from natural language requirements by an expert supervision to emphasis on the safety transformation. Also, our work does not require that the expert user know any temporal logic languages which has been case for most of the current literature.

IV. Formal Model Synthesis Procedure for Functional Requirements

The first step of computing the TSs is to extract the APs from the requirements. We have developed three Atomic Proposition Extraction Rules (APERs) that work on the parse tree of the requirement obtained from Enju. The resulting APs are then used to compute the states and transitions. The APERs are described next.

A. Atomic Proposition Extraction Rule 1 (APER 1)

APER 1 is based on the hypothesis that noun phrases in a requirement correspond to APs. Each subtree of the parse tree with an NX root (called an NX head) corresponds to a noun phrase and hence an AP. Therefore, APER 1 computes the subtrees corresponding to NX heads. If NX heads are nested, then the highest-level NX head is used to compute the AP. The terminal nodes of the subtree are conjoined together to form the noun phrase. APER 1 returns AP-list, which is the set of APs corresponding to a parse tree.

Procedure 1 APER1

```
Require: Parse-tree
1: AP-list ← $\emptyset$ ;
2: for each $n \in$ TerminalNodes(Parse-tree) do
3:     Start-cat = head(head($n$));
4:     if Start-cat = NX then
5:         $X$ = Sub-tree(Start-cat);
6:         while (head($X$) = NX ) $\vee$ (head($X$) = COOD)
                 $\vee$ (head($X$) =NX-COOD ) do
7:             $X$ = Sub-tree(head($X$));
8:         AP-list ← AP-list $\cup$ TerminalNodes(X) ;
```

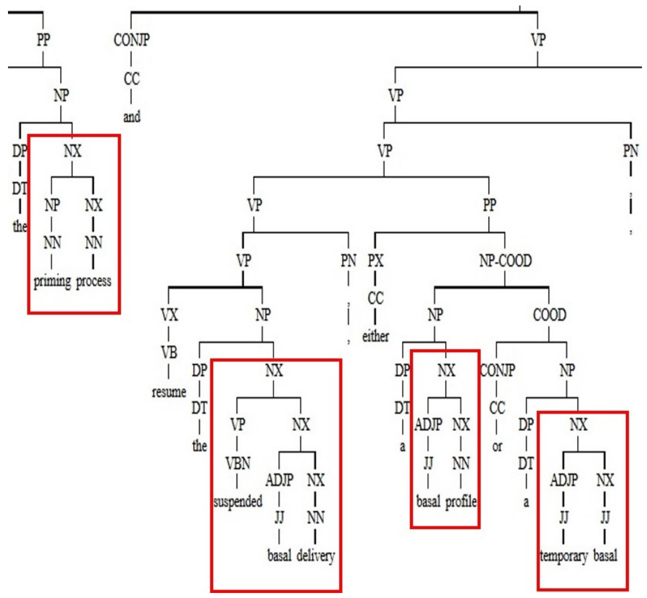


Figure 4. An Enju parsing tree portion shows some resulting APs by applying APER 1.

We now describe the procedure corresponding to APER 1 in detail. Firstly, AP-List is initialized to the empty set (line 1). The procedure then iterates through each terminal node n (line 2). The head of a node is its parent. If a terminal node is part of an NX subtree, its level two head will be marked as NX, which is checked in line 3. The level-two NX node of the terminal node is stored in variable State-cat. If the Start-cat is of NX category (line 4), a function called Sub-tree is used to get the resulting subtree (line 5), which is stored in variable X. A while loop is used to traverse the tree of X upwards checking if the head syntactic category is NX or COOD or NX-COOD (line 6). Only when one of the conditions is satisfied the subtree is stored in X (line 7). The terminal nodes of the resulting sub tree ’X’ will be added to AP-List as a new suggested AP (line 8). Figure 4 gives a sub tree example for APER 1. Note that APER 1 may result in the same AP being duplicated. Duplicates are checked and removed from the AP list in the overall approach.

As shown in Figure 4, the terminal nodes ’the’ and ’priming’ does not have head(head(n)) = NX. The first terminal node that has the NX category is ’process’. Traversing upwards, the NX related categories gives us the subtree which contains ’priming process’. This now constitutes the first AP for this part of requirement. Applying the APER 1 rule on the visible part of the sentence in Figure 4 gives us the following APs: ’priming process’, ’suspended basal profile’, ’basal profile’, and ’temporary basal’.

B. Atomic Proposition Extraction Rule 2 (APER 2)

APER 2 and APER 3 correspond to the two other parse tree patterns that also lead to noun phrases. APER 2 examines the parse tree for noun categories along with its upper verb head. APs will be the conjoined terminal nodes of the resulting sub tree. APER 2 states that APs are the terminal nodes under the head VP passing through NX (or its related phrases such as NX-COOD, COOD), NP (or its related phrases NP-COOD, COOD), and VX phrase.

Procedure 2 APER 2

```
Require: Parse-tree
1: AP-list ← $\emptyset$ ;
2: for each $n \in$ TerminalNodes(Parse-tree) do
3:     Start-cat = head(head($n$));
4:     $X_1$ ← $\emptyset$;
5:     if Start-cat = NX then
6:         $X$ = Sub-tree(Start-cat);
7:         while (head($X$) = NX ) $\vee$ (head($X$) = COOD)
                 $\vee$ (head($X$) =NX-COOD ) do
8:             $X$= Sub-tree(head($X$));
9:         while (head($X$) = NP) $\vee$ (head($X$) = COOD)
                 $\vee$ (head($X$) = NP-COOD) do
10:            $X_1$ = Sub-tree(head($X$));
11:        if (head($X_1$) = VX) $\wedge$ (head(head($X_1$)) = VP) then
12:            $X$ = Sub-tree(head(head($X_1$));
13:        else
14:            if (head($X_1$) = VP) then
15:                $X$ = Sub-tree(head($X_1$);
16:        AP-list ← AP-list $\cup$ TerminalNodes(X);
```

APER 2 is built on top of APER 1 to get atomic propositions for requirements that APER 1 is not able to collect. While APER 1 looks only for APs that are noun phrases, APER 2 looks for noun phrases that are further characterized by verb phrases. For example, if APER 1 finds the AP ”suspended basal delivery,” APER 2 will find ”resume the suspended basal delivery.”

APER 1 and APER 2 have the same algorithmic flow until finding the sub tree of X that is the top NX head (line 8). However, APER 2 does not consider the resulting X to be an AP like APER 1 does. Instead, X is the input of the next step. A while loop is used to search if the head category of X is in NP category or one of its related phrases (line 9). Only when the while loop condition is true, the new sub-tree is stored temporarily in the variable X_1 (line 10), where X_1 is a temporary variable initialized to null (line 4). This ensures that X does not change in this step for future use. The search for VX and VP categories is to be performed only when X_1 is not null.

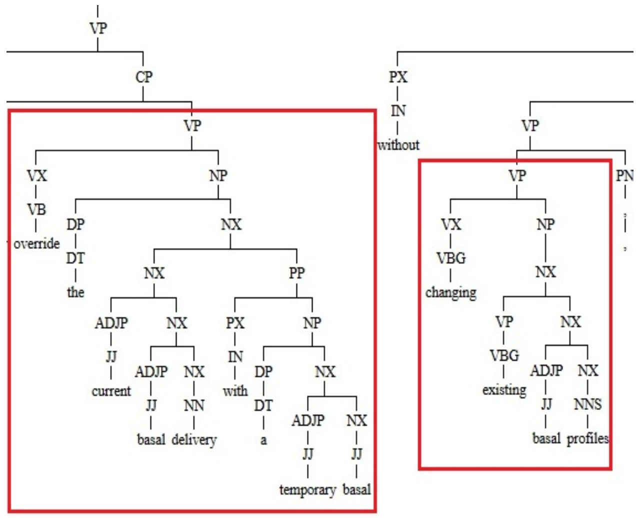


Figure 5. An Enju parsing tree portion shows some resulting APs by applying APER 2.

On the successful completion of NP category search, the search for VX category followed by VP categories is performed (line 11). When the if condition is satisfied, X is updated with the new sub-tree (line 12). In the case of failure of the if condition in line 11, a new search for VP category is performed on the head of NP category sub-tree (line 14). On success, X is updated with the new sub-tree (line 15). If either of the if conditions (line 11 and line 14) fail, then X will remain as the sub-tree of NX category. The terminal nodes of the resulting subtree in X is appended to the AP-list (line 16). Figure 5 shows a resulting sub tree example by applying APER 2.

Figure 5 shows that the procedure starts from left to right looking for level two NX nodes and traversing upward until higher NX nodes are accounted for. NP phrases are selected to expand the tree. Then choosing the upper level which is VP in this particular case (sometimes its VX $\rightarrow$ VP). The output of APER 2 for this tree portion is ’override the current basal delivery with a temporary basal’, and ’changing existing basal profiles’.

C. Atomic Proposition Extraction Rule 3 (APER 3)

APER 3 is built on top of APER 2, it explores the verb head levels in the parse tree like APER 2, but APER 3 eliminates some verb phrases that is not part of APs. This elimination is done based on the head of the VP category as illustrated in Procedure 3 below.

APER 3 and APER 2 have the same stream up to line 10. The algorithm starts with initializing temporary variables X_1 and Y to null (line 4). The search for syntactic categories start with the top NX phrase (line 7) and the resultant sub tree is stored in X (line 8). Then, the search begins for the top NP phrase (line 9) and the resultant sub tree is stored in X_1 (line 10) since the sub tree in X is needed for future use. As in APER2, the search for either VX phrase followed by VP phrase or just VP phrase is performed on X_1 and the resultant sub tree is stored in Y (lines 11-15). If and only if Y is not empty then the check on the head syntactic category is performed to ensure that it does not contain CP or COOD categories. In this case, X gets only the right child (line 16-18) i.e. the left child of Y is pruned. On the other hand, if Y has a CP or COOD head, X value will be updated to be equal to Y (line 20). Finally, terminal nodes of the resulting sub tree X will be saved in the AP-list as a new AP. The pruning process (line 18) is done to remove some action verbs which are not part of an AP.

Procedure 3 APER 3

```
Require: Parse-tree
 1: AP-list ← $\emptyset$ ;
 2: for each $n \in$ TerminalNodes(Parse-tree) do
 3:     Start-cat = head(head($n$));
 4:     $X_1 \leftarrow \emptyset$ , $Y \leftarrow \emptyset$;
 5:     if Start-cat = NX then
 6:         $X$ = Sub-tree(Start-cat);
 7:         while (head($X$) = NX) $\vee$ (head($X$) = COOD)
                 $\vee$ (head($X$) =NX-COOD ) do
 8:             $X$ = Sub-tree(head($X$));
 9:         while (head($X$) = NP) $\vee$ (head($X$) = COOD)
                 $\vee$ (head($X$) = NP-COOD) do
10:             $X_1$ = Sub-tree(head($X$));
11:         if (head($X_1$) = VX) $\wedge$ (head(head($X_1$)) = VP) then
12:             $Y$ = Sub-tree(head(head($X_1$));
13:         else
14:             if (head($X_1$) = VP) then
15:                 $Y$ = Sub-tree(head($X_1$);
16:         if ($Y \neq \emptyset$) then
17:             if head($Y$) $\neq$ CP) $\wedge$ (head($Y$) $\neq$ COOD) then
18:                 $X$ = Sub-tree(RightChild($Y$));
19:             else
20:                 $X = Y$;
21:         AP-list ← AP-list $\cup$ TerminalNodes(X);
```

Like APER2, APER3 also works on verb head categories. However, APER3 has some pruning techniques to remove parts of the sentence that should not be part of an AP. Consider the snippet in Figure 6, the sub tree ”issue an alert” is subjected to left branch pruning to remove the verb ’issue’ since such verbs do not add value in the AP. According to the algorithm, since the head node of VP is COOD, only the terminal nodes of the right child are considered as an AP. Applying APER

3 on the visible part of the requirement in Figure 6 gives the following APs: 'pump', 'an alert', and 'deny the request'.

The proposed APERs may be used individually or in combination depending on the system requirement and model functionally. However, no one rule is considered to be the best for all models because of the natural language structure.

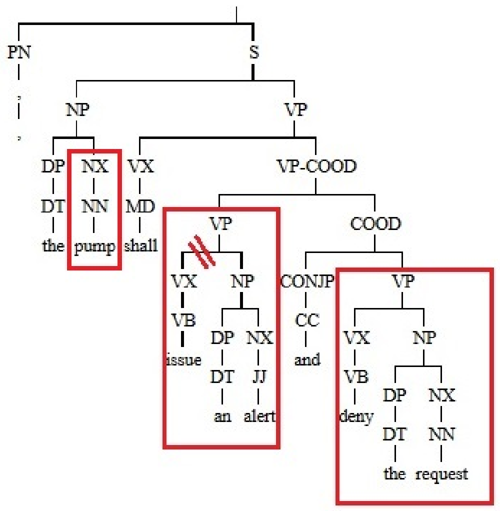


Figure 6. An Enju parsing tree portion shows some resulting APs using APER 3.

D. *High-Level Procedure for Specification Transition System Synthesis*

Procedure 4 Procedure for synthesizing TSs from system requirements

```
Require: set of requirements (System-requirements)
1:  TS-set ← $\emptyset$ ;
2:  for each $Req \in$ System-requirements do
3:      Parse-tree ← Get(Req_tree.xml);
4:      AP-list ← APER(Parse-tree);
5:      AP-list ← Eliminate_Dup(AP-list);
6:      AP-list ← USR_IN(AP-List);
7:      AP-truth-table ← Relation(AP-list);
8:      AP-truth-table ← USR_IN(AP-truth-table);
9:      S-list ← $\emptyset$;
10:     for each A $\in$ AP-truth-table do
11:         S-list[$i$] = A$_i$ ;
12:     S-list ← USR_IN(S-list);
13:     T ← CreateT(S-list);
14:     T ← USR_IN(T);
15:     TS ← CreateTS(T, S-list);
16:     TS-set ← TS-set U TS;
17:     return TS-set;
```

Procedure 4 shows the overall flow for computing the TSs. A set of system requirements in natural language are fed as input to the procedure. TS-set is the output of the procedure and will contain the set of transition systems that capture the input requirements as a formal model. TS-set is initialized to null (line 1). Each requirement is input to the Enju parser. The parser gives an xml file as output. A function called Get is used to obtain the xml file into the variable Parse-tree (line 3). The xml output in Parse-tree is subjected to the proposed APERs, which give the atomic propositions (APs) as output. APs are stored in the AP-list (line 4). Each requirement is subject to all APERs and the AP-list obtained is the union of the APs produced by each of the rules. The output obtained by using the APERs may contain duplicates, which are eliminated by using the function Eliminate_Dup (line 5). AP-list is then subjected to an expert user check, where the AP(s) might be appended, eliminated or revised based on the expert user's domain knowledge (line 6). Some of the APs maybe expressible as a Boolean function of other APs.

Therefore, next, a truth table (AP-truth-table) is created, where each row corresponds to an AP from AP-list and each column also corresponds to an AP from AP-list (line 7). Each entry in the table is a Boolean value (true or false). Completing the truth table determines the relationship of each AP with the other APs in the AP-list. The truth table is completed by the expert user (line 8). The list of states for the input requirements are stored in the variable S-list. S-list is initialized to null (line 9). Each truth table entry (A) is defined to be a single state in the transition system (line 10). This heuristic has worked well in practice. S-list is subjected to expert user input (line 12).

The transitions of the TS are computed next. The list of transitions (T) is initialized to a transition between every two states using function 'CreateT' (line 13). The transition list is subjected to expert user input (line 14). A transition system (TS) is constructed using the CreateTS function, which takes the transitions (T) and the list of states (S-list) as input (line 15). This transition system (TS) is then added to the transition system set (TS-set) (line 16). The procedure finally returns a set of transition systems for all the requirements in an application (line 17).

V. Formal Model Synthesis Procedure for Timing Requirements

In this section, the approach is extended to deal with timing requirements. When synthesizing transition systems (TSs), the core activity was the extraction of APs. For synthesizing timed transition system, the core activity is the extraction of APs and TCs. An additional extraction rule is developed, that can be applied on timed requirements not only to get APs but also to extract the timing constraints (TCs) on each state transition.

A. *Atomic Proposition and Timing Constrains Extraction Rule (APTCER) for Timed Transition System*

This section explains a new proposed rule that analyzes timing requirements to get APs with their corresponding TCs as a base for building TTSs. This rule called Atomic Proposition and Timing Constrains Extraction Rule (APTCER) is specified as Procedure 5 and works as follows. First, the timing requirement is split into smaller phrases that are individually analyzed (lines 1-14 of Procedure 5). These phrases are called Timed Based Sentences (TBSs). Each resulting phrase is then analyzed to extract the APs and TCs in that phrase (lines 15-38 of Procedure 5). The list of APs and TSs are stored in $\langle AP-list, TC-list\rangle$.

APTCER takes the parse tree of the timing requirement as input. The parse tree is obtained by applying the Enju parser on the timing requirement. APTCER initilizes the list of TBSs (TBS-list) to the empty list (line 1). APTCER then searches for sub-trees with root as ”S” and with left child of ”NP” and right child as ”VP” (lines 2-5). Each such sub-tree is a TBS.

Note that TBSs can be nested in that there can be a TBS inside of a TBS. The nested TBSs need to partitioned and analyzed individually. This is done by searching for sub-trees inside the TBS with ”SCP” or ”S” roots. Such sub trees are cut out and the resulting TBS is returned (lines 7-13).

Next, the TBSs are analyzed to extract the APs and TCs. The extraction is performed by analyzing both the left child and the right child of the TBS. The left and right sub-trees are assigned to variables A and B, respectively (lines 19 and 23). Then APER 1 is used to analyze both A and B. Through empirical observation, it has been determined that the APs extracted by APER 1 from sub-tree A (line 21, 22) corresponds to APs but the APs extracted by APER 1 from sub-tree B (line 25, 26) corresponds to TCs. The resulting AP-list and TC-list are corresponds to one TBS (line 27), the TBS's pair is saved in the final TBS-list (line 28).

Applying lines 1-14 of APTCER on the requirement in Figure 7 gives three TBSs, they are shown in separate red boxes. While the rest of the algorithm (lines 15-38) works on each TBS to find it's AP-list and TC-list. The left sentence has one AP which is ”air-line-alarm” and one TC which is ”maximum delay time of x minutes”. The resulting AP and TC will be saved as a pair. This helps in identify that the AP and TC are correlated, which is used to determine the transition for which the TC should be applied. More specifically, the TC will be applied to a transition from a state in which the corresponding AP is true. Overall the three TBS from Figure 7 give the following. AP-list is: 'air-in-line alarm', 'air bubbles larger than y L', and 'insulin administrations'. TC-list will have one TC: maximum delay time of x seconds which related to the AP of 'air-in-line alarm' as one pair.

Note that a TBS can correspond to more than one AP and more than one TC. For example, Figure 8 shows a TBS that has two APs (in red boxes) and one TC (in a green box).

B. High-Level Procedure for Specification Timed Transition System Synthesis

Procedure 6 shows the overall flow for computing the TTSs. A set of natural language timing requirements are input to the procedure. TTS-set is the output of the procedure and will contain the set of timed transition systems that capture the input requirements as a formal model.

TTS-set is initialized to null (line 1). Each timing requirement is input to the Enju parser. The parser gives an xml file as output. A function called Get is used to obtain the xml file into the variable Parse-tree (line 3). The xml output in Parse-tree is subjected to our proposed APTCER, which gives the TBS-list that are pairs of atomic propositions and their related timing constrains lists (line 4). The synthesizing procedure then iterates through all TBSs (line 5) to get thier corresponding pair of APs and TCs (line 6).

AP-lists is subjected to an expert user check, where the APs might be appended, eliminated or revised based on the expert users domain knowledge (line 7). Some of the APs maybe expressible as a Boolean function of other APs. Therefore, next, a truth table (AP-truth-table) is created, where each row corresponds to an AP from AP-lists and each column also corresponds to an AP from AP-lists (line 8). Each entry in the table is a Boolean value (true or false). Completing the truth table determines the relationship of each AP with the other APs in the AP-lists. The truth table is completed by the expert user (line 9). TC-list is then checked by the expert user, where some TCs might be appended, eliminated or revised based on the expert users domain knowledge (line 10).

Procedure 5 APTCER

```
Require: Parse-tree
 1: TBS-list ← $\emptyset$ ;
 2: for each $Head - Cat \in$ Head(Parse-tree) do
 3:     if Head-Cat = S then
 4:         if (Left-Child (S)= NP $\vee$ NP-COOD ) $\wedge$
                                  (Right-Child (S)= VP) then
 5:             TBS = Sub-tree (S);
 6:             for each Child-Head (TBS) do
 7:                 if Child-Head (TBS) = SCP then
 8:                     Cut-Sub-tree (SCP);
 9:                     return TBS;
10:                 else
11:                     if Child-Head (TBS) = S then
12:                         Cut-Sub-tree (S);
13:                         return TBS;
14:             TBS-list ← TBS-list $\cup$ TBS;
15: $k \leftarrow \emptyset$;
16: for each $TBS \in$ TBS-list) do
17:     K = k + 1;
18:     $A \leftarrow \emptyset$ , $B \leftarrow \emptyset$;
19:     A = Sub-tree (left-Child (TBS));
20:     $AP - list_k \leftarrow \emptyset$;
21:     APER 1 (A) → AP-list;
22:     $AP - list_k \leftarrow$ AP-list ;
23:     B = Sub-tree (Right-Child (TBS));
24:     $TC - list_k \leftarrow \emptyset$;
25:     APER 1 (B) → AP-list;
26:     $TC - list_k \leftarrow$ AP-list ;
27:     $TBS_k = \langle AP - list_k, TC - list_k \rangle$;
28:     TBS-list ← TBS-list $\cup$ $TBS_k$;
```

Next, the states and transitions of the TTS are computed. S-list variable (list of states) is initialized to null (line 11). Each truth table entry (A) (line 12) is defined to be a single state in the transition system (line 13). S-list is subjected to expert user input (line 14). The transitions of the TTS are computed next. The list of transitions (T) is initialized to a transition between every two states using function CreateT (line 15). The transition list is subjected to expert user input (line 16) where some transitions might be pruned. A function called 'Apply-TC-list' is applied to link each TC to all transitions emanating from states in which the corresponding APs are true, based on the TBS pair → TBS $\langle AP - list, TC - list \rangle$ (line 17). The expert user will confirm, modify, or apply the TC on specific transition/s based on his domain knowledge (line 18). For the remaining transitions that do not have any timing bounds, the timing bounds are open from zero to infinity $\langle 0, \infty \rangle$. For this reason a new function called 'Apply-TC-bounds' is applied on each transition that has no TC (line 19).

A timed transition system (TTS) is constructed using the CreateTS function as in procedure 4, this function takes the transitions (T) linked with their timing conditions and the list of states (S-list) as input (line 20) to create a TTS. The resulting TTS is then added to the timing transition system set (TTS-set) (line 21). The procedure finally returns a set of timing transition systems for all timing requirements that have been fed to the algorithm (line 22).

Procedure 6 Procedure for synthesizing TTSs from timing requirements

```
Require: set of requirements (Timed-requirements)
 1: TTS-set ← ∅ ;
 2: for each  $Req \in$ Timed-requirements  do
 3:     Parse-tree ← Get(Req_tree.xml);
 4:     TBS-list ← APTCER(Parse-tree);
 5:     for each  $TBS \in$ TBS-list do
 6:         Get ($\langle AP-list, TC-list \rangle$);
 7:     AP-list ← USR_IN(AP-list);
 8:     AP-truth-table ← Relation(AP-list);
 9:     AP-truth-table ← USR_IN(AP-truth-table);
10:     TC-list ← USR_IN(TC-list);
11:     S-list ← ∅;
12:     for each  A ∈ AP-truth-table  do
13:         S-list[$i$] = A$_i$ ;
14:     S-list ← USR_IN(S-list);
15:     T ← CreateT(S-list);
16:     T ← USR_IN(T);
17:     T ← Apply-TC-list ;
18:     T ← USR_IN(TC);
19:     T ← Apply-TC-bounds $\langle 0, \infty \rangle$;
20:     TTS ← CreateTS(T, S-list);
21:     TTS-set ← TTS-set U TTS;
22:     return TTS-set;
```

VI. Case Study: Generic Insulin Infusion Pump (GIIP)

Insulin pump is a medical device that delivers doses of insulin 24 hours a day to patients with diabetes. It is typically used to keep the blood glucose level in an acceptable range. Overdose of insulin can lead to low blood sugar that can lead to coma/death. Therefore, the insulin pump is a safety-critical device.

The Generic Insulin Infusion Pump (GIIP) has been proposed [30], which lists a set of safety requirements for insulin pumps. We use these safety requirements to explain our approach. GIIP has proposed a list of both functional and timing requirements, examples will be given about both cases.

A. *Functional requirements of GIIP*

GIIP model abstracts requirements that explains how specific critical behaviour of the system can be controlled, functional requirements are introduced to solve common hazards in the insulin pump's market that might happen during insulin administration and not related to specific timing constraints.

As an example, consider requirement 1.8.2 (from [30]) which is needed to address a hazard that may happen in the suspension mode of the pump. Suspension mode can occur when the pump may be in refill or priming or insulin delivery processes. The insulin pump has two type of insulin deliveries: bolus and basal. Bolus is a high insulin rate that is recommended in case of low blood glucose level.

Requirement 1.8.2: *When the pump is in suspension mode, insulin deliveries shall be prohibited. Any incomplete bolus delivery shall be stopped and shall not be resumed after the suspension.*

From safety requirement 1.8.2, it is clear that the pump should not resume a suspended bolus automatically after returning from suspension since they would be an unexpected amount of insulin.

Requirement 1.8.5: *When the pump resumes from suspension, calculations shall be performed to synchronize insulin used and remaining reservoir volume.*

Requirement 1.8.5 is an extension of how the pump should function after returning from the suspension mode. Here two requirements are needed to address one safety hazard. When algorithm 4 is applied on these two requirements, the first step is collecting the APs by using the extraction rules. Applying APER 2 on 1.8.2 gives: ”pump”, ”suspension mode”, ”insulin deliveries”, ”incomplete bolus delivery”, and ”suspension”. Applying APER 2 on 1.8.5 gives: ”pump”, ”suspension”, ”calculations”, and ”synchronize insulin used and remaining reservoir volume”. Next, duplicate APs are to be removed. This eliminates 'pump' and 'suspension' from the AP-list. Now, the expert user intervenes for manipulating the AP-list, where APs can be deleted, modified or even inserted based on the expert user's domain knowledge. This yields the final AP-list as ”suspension mode” (SPM), ”insulin deliveries” (INDV), ”incomplete bolus delivery” (IBO) and ”synchronize insulin used and remaining reservoir volume” (SYNC). Next, the AP-truth-table to define relations between APs is constructed as shown in Table I.

TABLE I. AP-TRUTH-TABLE FOR REQUIREMENT 1.8.2 AND 1.8.5 FROM AP-LIST

APs → ↓	SPM	INDV	IBO	SYNC
SPM	T	F	F	F
INDV	F	T	F	F
IBO	F	T	T	F
SYNC	F	F	F	T

Here, each row represents a state. For example, SPM represents a state where suspension mode is true, IBO is false, INDV is false, and SYNC is also false; which emphasizes that insulin bolus should not be active during suspension.

Finally, Procedure 4 applies transitions between every two states as shown in Figure 9a. The expert user will approve or remove some unacceptable transitions. Figure 9b shows the final transition system.

B. *Timing requirements of GIIP*

The application of APTCER to some timing requirements of GIIP are described next. Timing requirements are also critical to be preserved. In GIIP, a motor controls the fluid injection and therefore the fluid flow rate and dosage. The motor is in turn controlled by software and the speed of the motor is time controlled by the software. The timing requirements of GIIP are also safety-critical because if the software violates these requirements, the dosage can be affected. Overdose or under dose of medicines can be very harmful or even fatal to the patient.

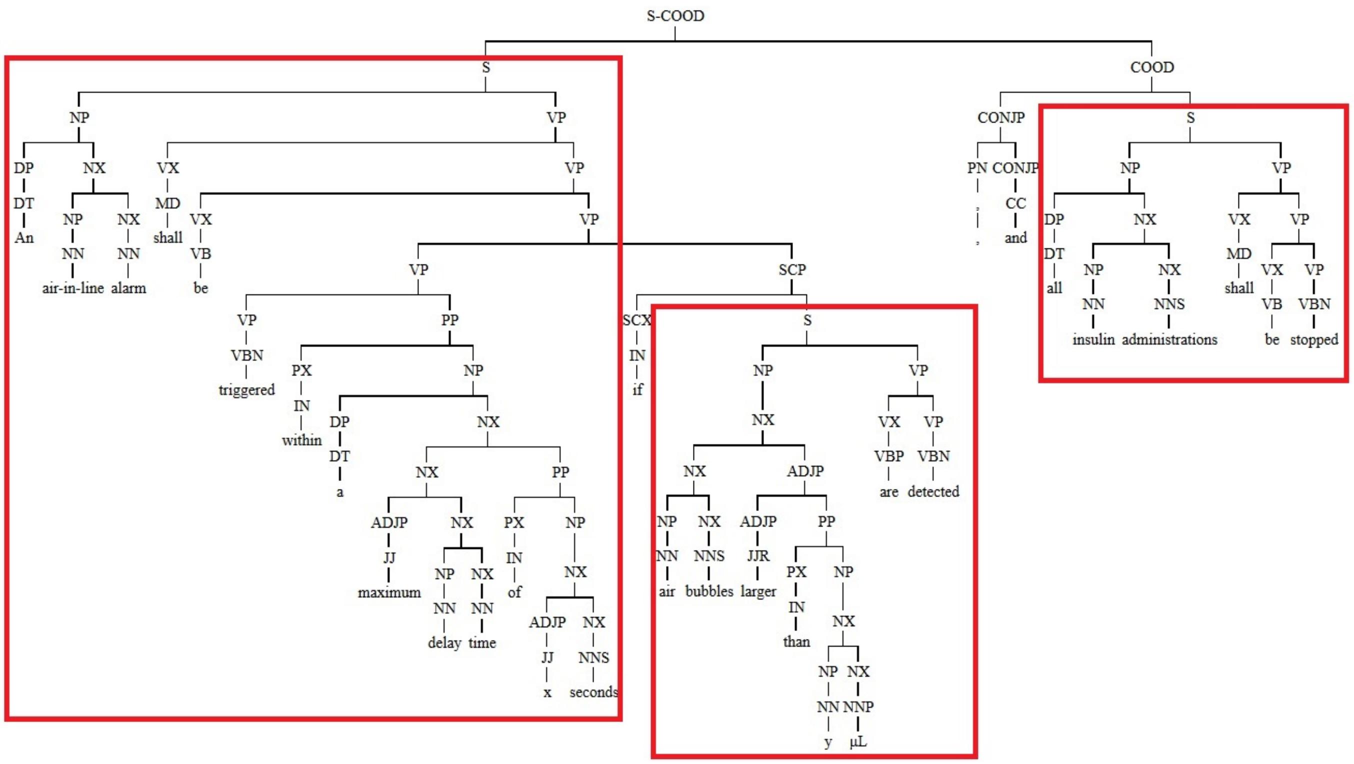


Figure 7. An Enju parsing tree shows three resulting TBSs after applying APTCER.

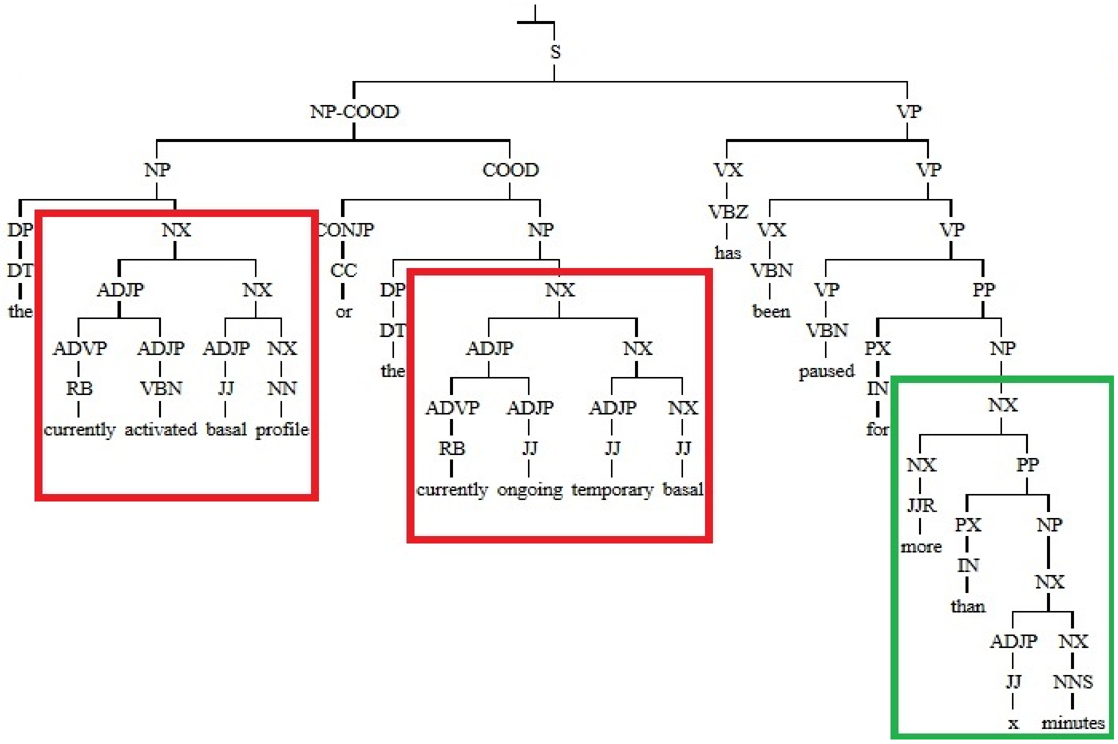


Figure 8. An Enju parsing tree portion shows the resulting TBS $\langle AP - list, TC - list \rangle$) after applying APTCER.

TABLE II. RESULTING TRANSITION SYSTEMS BY APPLYING PROCEDURE 4 AND APERS ON A SET OF SYSTEM REQUIREMENTS

Req. NO.	APER	Total No. of APs	No. of APs Without DP	User input			Final		
				AP added	AP removed	AP modified	APs	states	transitions
1.1.1	1	10	10	0	6	0	5	4	5
	2	10	10	0	5	0			
	3	10	10	0	6	0			
1.1.3	1	7	7	0	3	2	4	4	4
	2	7	7	0	3	2			
	3	7	7	0	3	1			
1.2.4 , 1.2.6, 1.2.7	1	24	12	3	5	1	10	10	14
	2	24	18	0	8	0			
	3	24	16	2	8	0			
1.3.5	1	11	6	1	3	0	4	4	4
	2	11	8	0	4	1			
	3	11	8	1	5	0			
1.8.2, 1.8.5	1	9	7	1	3	1	4	4	5
	2	9	7	0	3	0			
	3	9	7	0	3	0			
2.2.2, 2.2.3	1	6	6	0	3	1	3	3	4
	2	7	6	0	3	1			
	3	7	6	0	3	2			
3.1.1	1	15	14	0	9	0	5	3	3
	2	14	12	0	7	0			
	3	14	13	0	8	0			
3.2.5	1	10	9	0	7	2	3	3	3
	2	7	7	0	4	1			
	3	7	7	0	4	1			
3.2.7	1	4	4	0	1	0	3	3	3
	2	4	4	0	1	1			
	3	4	4	0	1	0			

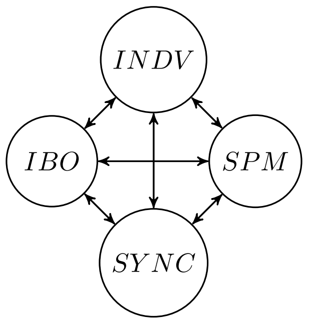
As an example, consider requirement 1.6.1 (from [30]) which helps patients to be aware of the occurrence of an air in line hazard. Air in line hazard is the presence of air bubbles in the pump above the acceptable range. The requirement states that if the air in line problem occurred during insulin delivery, an air in line alarm should start in a time not more than x minutes, in addition, every ongoing insulin delivery must be stopped. The alarm will give the patient a warning that a problem is going to happen, so the patient will interact with the pump and solve the issue to prevent incorrect insulin doses or other problems.

Requirement 1.6.1: *An air-in-line alarm shall be triggered within a maximum delay time of x seconds if air bubbles larger than y μL are detected, and all insulin administrations shall be stopped.*

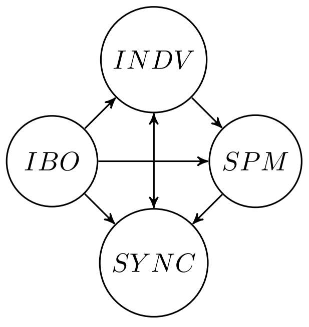
When procedure 6 is applied to this requirement, the first step is collecting the lists of TBS by applying APTCER, which gives three separate TBSs. TBS1 is ”An air-in-line alarm shall be triggered within a maximum delay time of x seconds”, while TBS2 is ”air bubbles larger than y μL are detected”, and TBS3 is ”all insulin administrations shall be stopped”.

Next, the AP-list and TC-list for each TBS is computed. AP-list contains: ”air-in-line alarm”, ”air bubbles larger than y μL”, and ”insulin administrations”. TC-list contains: ”maximum delay time of x seconds” which is related to the AP: ”air-in-line alarm” in TBS1.

Now, the expert user intervenes to manipulate the AP-list, where APs can be deleted, modified or even inserted based on the expert users domain knowledge. This yields the final AP-list as "air-in-line alarm" (ALRM), "air bubbles larger than y μL" (AIRB), "insulin administrations" (INSAD). Next, the AP-truth-table to define relations between APs is constructed as shown in Table III.

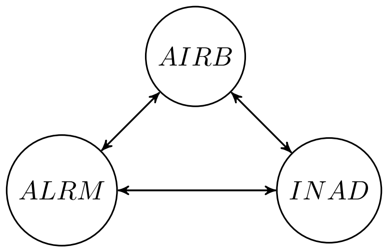


(a) TS with all suggested transitions.

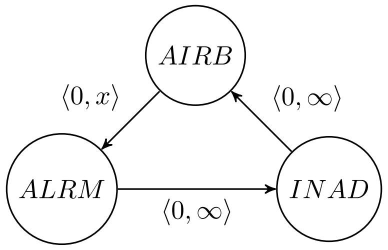


(b) TS after removing some transitions.

Figure 9. Finite state machine for suspension mode requirements (1.8.1 and 1.8.5).



(a) TTS with all suggested transitions.



(b) TTS after applying the TCs and removing some transitions.

Figure 10. Timed finite state machine for air-in-line requirement (1.6.1).

TABLE III. AP-TRUTH-TABLE FOR TIMING REQUIREMENT 1.6.1 FROM AP-LIST

APs → ↓	AIRB	INAD	ALRM
AIRB	T	T	F
INAD	F	T	F
ALRM	F	F	T

As in Table I, each row in the Table III represents a state. For example, AIRB represents a state where AIRB is true, INAD is also true, while ALRM is false; which explains the problem of having air bubbles while an insulin administration is given to the patient. Now, the user can make changes to the TC-list which may have a TC that corresponds to one or more APs. After the states are computed, the expert user can add or modify any of the states if needed.

Then, Procedure 6 applies transitions between every two states, the expert user will approve or remove some unacceptable transitions as shown in Figure 10a. After following Procedure 6, the final TTS is shown in Figure 10b

VII. Results analysis

An evaluation process is applied on the resulting TSs and TTSs by using NuSMV and UPPAAL model checkers respectively. Firslty, evaluation of the first approach (APERs and Procedure 4) for TSs is performed using the NuSMV model checker. A model checker is a tool that can check if a TS or a TTS satisfies a set of properties. The properties have to be expressed in a temporal logic. Here, we have used CTL to express the properties. The CTL properties are written manually for each of the requirements that are subjected to our approach. NuSMV is used to check if the TSs synthesized by the first approach satisfied the CTL properties corresponding to each functional requirement.

Secondly, UPPAAL is used to verify the resulting TTSs by applying APTCER and Procedure 6 (the second approach). UPPAAL is a tool that can verify real time systems and is based on the timed automata theory [31]. UPPAAL is used to check if the TTSs synthesized by the second approach satisfied the CTL properties corresponding to each timing requirement.

Table II shows the results of applying Procedure 4 on a number of GIIP requirements. The requirement numbers in the table are from [30]. All the final TSs satisfied their corresponding CTL properties. Each requirement or set of requirements (listed in column 1) have been subjected to the extraction rules (column 2), where column 3 shows the total number of APs resulting from each extraction rule. Column 4 gives the number of APs after removing the duplicate APs. In addition, a record of the suggested expert user intervention for adding, removing or modifying the APs is shown in column 5. The final number of APs, states, and transitions are shown in column 6.

As shown in Table II, when a requirement is subjected to the APERs, the resultant output from each APER may be different even though the number of APs is the same. For requirements 1.8.2 and 1.8.5, although applying APER1, APER2, and APER3 give the same number of APs, APER1 gives different list of APs from APER2 and APER3.

Table IV presents the results of applying Procedure 6 on a number of GIIP timing requirements from [30]. All applied CTL properties are satisfied by the resulting TTSs. The listed requirements (column 1) are subjected to the APTCER which gives list of TBSs for each requirement (column 2). column 3 and 4 show the number of the resulting APs and TCs respectively. Column 5 shows the pair of AP-list and TC-list. As in Table II, column 6 has the user interventions of appending, deleting, or modifying the AP-lists. The final TTS's components are shown in column 7: the number of APs, the number of states, and finally the number of transitions between states.

VIII. Conclusion

The key ideas of our approach for transforming requirements into transition systems and timed transitions systems

TABLE IV. RESULTING TIMED TRANSITION SYSTEMS BY APPLYING PROCEDURE 6 AND APTCER ON A SET OF TIMING REQUIREMENTS

Req. NO.	Total No. of TBSs	Total No. of APs	Total No. of TCs	(AP,TC)	User input			Final		
					AP added	AP removed	AP modified	APs	states	transitions
1.2.8	2	3	3	$\langle 2,1\rangle$ $\langle 1,2\rangle$	1	0	1	4	3	4
1.6.1	3	3	1	$\langle 1,1\rangle$	0	0	0	3	3	7
1.8.4	2	3	2	$\langle 1,1\rangle$ $\langle 1,1\rangle$	1	1	2	3	3	4
2.2.1	4	6	1	$\langle 4,1\rangle$	0	0	1	6	3	4

are the following. The extraction rules work on the parse tree to get an initial list of APs and TCs. The AP truth table is used to establish relationships between the initial list of APs. For example, an AP may be expressible as a conjunction of two other APs. The initial expert user pruned list of APs gives insight into the states of the transition system. We have found empirically that having one state for this initial pruned AP list is a good heuristic to compute the states of the transition system. Transitions are applied between every two states and then pruned by the expert user. TCs are paired with APs and this information is used to assign TCs to transitions.

Transforming natural language requirements into formal models is quite a hard problem and hard to get right without input from domain expert. Our approach sets up a very structured process, where the tool does lot of the work in analyzing and synthesizing TSs and TTSs, but also allows for input from domain expert. The proposed methodology has worked very well in practice for the GIIP requirements. All the TSs and TTSs computed for the requirements satisfied their corresponding CTL properties.

Acknowledgment

This publication was funded by a grant from the United States Government and the generous support of the American people through the United States Department of State and the United States Agency for International Development (USAID) under the Pakistan - U.S. Science & Technology Cooperation Program. The contents do not necessarily reflect the views of the United States Government. The authors would like to acknowledge Dr. Vinay Gonela for helping with proofreading the paper.

References

[1] E. M. Al-qtiemat, S. K. Srinivasan, M. A. L. Dubasi, and S. Shuja, "A methodology for synthesizing formal specification models from requirements for refinement-based object code verification," in The Third International Conference on Cyber-Technologies and Cyber-Systems. IARIA, 2018, pp. 94–101.

[2] FDA, "List of Device Recalls, U.S. Food and Drug Administration (FDA)," 2018, last accessed: 2018-09-10. [Online]. Available: https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfRES/res.cfm

[3] R. Kaivola et al., "Replacing testing with formal verification in intel coretm i7 processor execution engine validation," in Computer Aided Verification, 21st International Conference, CAV, Grenoble, France, June 26 - July 2, 2009. Proceedings, ser. Lecture Notes in Computer Science, A. Bouajjani and O. Maler, Eds., vol. 5643. Springer, pp. 414–429. [Online]. Available: https://doi.org/10.1007/978-3-642-02658-4_32

[4] T. Ball, B. Cook, V. Levin, and S. K. Rajamani, "SLAM and static driver verifier: Technology transfer of formal methods inside microsoft," in Integrated Formal Methods, 4th International Conference, IFM, Canterbury, UK, April 4-7, 2004, Proceedings, ser. Lecture Notes in Computer Science, E. A. Boiten, J. Derrick, and G. Smith, Eds., vol. 2999. Springer, pp. 1–20. [Online]. Available: https://doi.org/10.1007/978-3-540-24756-2_1

[5] K. Bhargavan et al., "Formal verification of smart contracts: Short paper," in Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security, PLAS@CCS, Vienna, Austria, October 24, T. C. Murray and D. Stefan, Eds. ACM, pp. 91–96. [Online]. Available: http://doi.acm.org/10.1145/2993600.2993611

[6] D. Delmas et al., "Towards an industrial use of fluctuat on safety-critical avionics software," in International Workshop on Formal Methods for Industrial Critical Systems. Springer, 2009, pp. 53–69.

[7] P. Manolios, "Mechanical verification of reactive systems," PhD thesis, University of Texas at Austin, August 2001, last accessed: 2018-10-10. [Online]. Available: http://www.ccs.neu.edu/home/pete/research/phd-dissertation.html

[8] M. A. L. Dubasi, S. K. Srinivasan, and V. Wijayasekara, "Timed refinement for verification of real-time object code programs," in Working Conference on Verified Software: Theories, Tools, and Experiments. Springer, 2014, pp. 252–269.

[9] Tsujii laboratory, Department of Computer Science at The University of Tokyo, "Enju - a fast, accurate, and deep parser for English," 2011, available from http://www.nactem.ac.uk/enju, [accessed: 2018-07-10].

[10] V. Ágel, Dependency and valency: an international handbook of contemporary research. Walter de Gruyter, 2003, vol. 1.

[11] S Ghosh et al., "Automatic requirements specification extraction from natural language (ARSENAL)," SRI International, Menlo Park, CA, Tech. Rep., 2014.

[12] D. Aceituna, H. Do, and S. Srinivasan, "A systematic approach to

transforming system requirements into model checking specifications," in Companion Proceedings of the 36th International Conference on Software Engineering. ACM, 2014, pp. 165–174.

[13] I. G. Harris, "Extracting design information from natural language specifications," in Proceedings of the 49th Annual Design Automation Conference. ACM, 2012, pp. 1256–1257.

[14] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Translating structured English to robot controllers," Advanced Robotics, vol. 22, no. 12, 2008, pp. 1343–1359.

[15] R. L. Smith, G. S. Avrunin, L. A. Clarke, and L. J. Osterweil, "Propel: an approach supporting property elucidation," in Proceedings of the 24th International Conference on Software Engineering. ACM, 2002, pp. 11–21.

[16] K. Shimizu, "Writing, verifying, and exploiting formal specifications for hardware designs," Ph.D. dissertation, PhD thesis, Stanford University, 2002.

[17] D. Zowghi, V. Gervasi, and A. McRae, "Using default reasoning to discover inconsistencies in natural language requirements," in Software Engineering Conference. APSEC 2001. Eighth Asia-Pacific. IEEE, pp. 133–140.

[18] V. Gervasi and D. Zowghi, "Reasoning about inconsistencies in natural language requirements," ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 14, no. 3, 2005, pp. 277–330.

[19] W. Scott, S. Cook, and J. Kasser, "Development and application of a context-free grammar for requirements," in SETE 2004: Focussing on Project Success; Conference Proceedings; 8-10 November 2004. Systems Engineering Society of Australia, 2004, p. 333.

[20] X. Xiao, A. Paradkar, S. Thummalapenta, and T. Xie, "Automated extraction of security policies from natural-language software documents," in Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering. ACM, 2012, p. 12.

[21] Z. Ding, M. Jiang, and J. Palsberg, "From textual use cases to service component models," in Proceedings of the 3rd International Workshop on Principles of Engineering Service-Oriented Systems. ACM, 2011, pp. 8–14.

[22] C. Rolland and C. Proix, "A natural language approach for requirements engineering," in International Conference on Advanced Information Systems Engineering. Springer, 1992, pp. 257–277.

[23] P. Bouyer, U. Fahrenberg, K. G. Larsen, N. Markey, J. Ouaknine, and J. Worrell, "Model checking real-time systems," in Handbook of Model Checking., E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, Eds. Springer, 2018, pp. 1001–1046.

[24] D. Knorreck, L. Apvrille, and P. de Saqui-Sannes, "TEPE: a sysml language for time-constrained property modeling and formal verification," ACM SIGSOFT Software Engineering Notes, vol. 36, no. 1, 2011, pp. 1–8.

[25] A. Shrivastava, M. Mehrabian, M. Khayatian, P. Derler, H. A. Andrade, K. Stanton, Y. Li-Baboud, E. Griffor, M. Weiss, and J. C. Eidson, "A testbed to verify the timing behavior of cyber-physical systems: Invited," in Proceedings of the 54th Annual Design Automation Conference, DAC 2017, Austin, TX, USA, June 18-22, 2017. ACM, 2017, pp. 69:1–69:6.

[26] J. Peters, N. Przigoda, R. Wille, and R. Drechsler, "Clocks vs. instants relations: Verifying CCSL time constraints in UML/MARTE models," in 2016 ACM/IEEE International Conference on Formal Methods and Models for System Design, MEMOCODE, Kanpur, India, November 18-20. IEEE, pp. 78–84.

[27] E. Kang, L. Huang, and D. Mu, "Formal verification of energy and timed requirements for a cooperative automotive system," in Proceedings of the 33rd Annual ACM Symposium on Applied Computing, SAC 2018, Pau, France, April 09-13, H. M. Haddad, R. L. Wainwright, and R. Chbeir, Eds. ACM, pp. 1492–1499.

[28] G. Carvalho, A. Cavalcanti, and A. Sampaio, "Modelling timed reactive systems from natural-language requirements," Formal Asp. Comput., vol. 28, no. 5, 2016, pp. 725–765.

[29] J. Hassine, "Early modeling and validation of timed system requirements using timed use case maps," Requir. Eng., vol. 20, no. 2, 2015, pp. 181–211.

[30] Y. Zhang, R. Jetley, P. L. Jones, and A. Ray, "Generic safety requirements for developing safe insulin pump software," Journal of diabetes science and technology, vol. 5, no. 6, 2011, pp. 1403–1419.

[31] G. Behrmann, A. David, and K. G. Larsen, "A tutorial on uppaal," in Formal methods for the design of real-time systems. Springer, 2004, pp. 200–236.

Reviewing National Cybersecurity Awareness for Users and Executives in Africa

Maria Bada

Cambridge Cybercrime Centre
Computer Laboratory,
Cambridge University UK
Global Cyber Security Capacity Centre,
University of Oxford, UK
Academy for Computer Science
and Software Engineering
University of Johannesburg
South Africa
Email: maria.bada@cl.cam.ac.uk

Basie Von Solms

Academy for Computer Science
and Software Engineering
University of Johannesburg,
South Africa
Email: basievs@uj.ac.za

Ioannis Agrafiotis

Department of Computer Science
University of Oxford
Email: ioannis.agraotis@cs.ox.ac.uk

Abstract—**There is an unprecedented increase in cybercrime globally observed over the last years. One of the regions driving this increase is Africa, where significant financial losses are reported. Yet, citizens of African countries are not aware of the risks present in cyberspace. The design and implementation of national awareness campaigns by African countries to address this problem are in their infancy state, mainly due to the absence of capacity building efforts. As part of the Global Cybersecurity Capacity Centre (GCSCC) programme, we conducted a series of focus groups in six African countries, in order to assess their cybersecurity posture, a critical component of which is user and executive awareness of cyber risks. This paper is an extended version of previous work where an initial analysis of awareness for cyber risk in African countries was presented. In this extended version, we reflect on best practice approaches for developing national awareness campaigns and use these as a framework to analyse qualitative data from the focus groups. We discuss the current state of African countries with regards to the implementation of national cybersecurity awareness campaigns for users and executives, the main obstacles in combating cybercrime and conclude with recommendations on how African countries can identify and prioritise activities to increase their capacity regarding cybersecurity awareness.**

Keywords–Cybersecurity; National strategies; Cyber threat awareness; Risk

I. Introduction

Over the last years, there has been an unprecedented increase in cybercrime globally [1], [2], [3]. Africa is a region with one of the highest rates of cybercrime affecting the strategic, economic and social growth development of the region [4]. Reports suggest that, inter alia, estimated costs have soared up to \$550 million for Nigeria, \$175 million for Kenya and \$85 for Tanzania [4].

Additionally, the growth in Internet use has been facilitated by high proliferation and adoption of mobile communications. Speedy diffusion and adoption have exposed the public to unprecedented individual security threats via the mobile platform [5].

Studies have confirmed that mobile phones have been used as a platform for distributing viruses as well as a transmission of viruses over Bluetooth services [6]. In some instances, mobile phones have been used to propagate hate speech as evidenced in Kenya after the December 2007 elections that fuelled ethnic violence [7].

One of the factors creating a permissive environment for cybercrime is the lack of awareness in the African public regarding risks when using cyberspace [4]. Additionally, the level of development of digital infrastructure in African countries directly influences their security posture. Reports suggest that cyber criminals rely on the very poor security habits of the general population [8] and urge policy makers to engage in awareness campaigns [4] since there is strong evidence that such initiatives can efficiently lower the success rate of cybercrime [9]. More specifically, there are white papers estimating that an investment in security awareness and training can potentially change user's behavior and reduce cyber-related risks by 45% to 70% [9].

It is evident that Cybersecurity Awareness is a very important step in the fight against cybercrime in Africa. For that reason, it is essential for any African country that intends to implement interventions in this area to have a holistic understanding of the level of Cybersecurity Awareness in that country. Towards this direction, there have been efforts to capture the status of Cybersecurity Awareness (understanding on cyber threats and risk, cyber hygiene, and appropriate response options) in Africa [10], and in general, the findings suggest that the absence of awareness campaigns regarding cybersecurity and Internet safety create a lax environment for information security [10]. In 2016, only 11 (20.3%) out of 54 countries had implemented cybersecurity (CS) laws and regulations [11]. Additionally, the lack of an adequately skilled workforce on cybersecurity can impose great challenges to many African countries.

This paper is an extended version of previous work [1] where an initial analysis of awareness for cyber risk in African countries was presented. In this extended version, we analyse qualitative data from six African countries that was collected when applying the Cybersecurity Capacity Maturity Model for Nations (CMM) developed by the Global Cybersecurity Capacity Centre (GCSCC) at the University of Oxford [12]. We reflect on best practice approaches for developing campaigns and draw conclusions on what the current state of African countries is regarding awareness in risks from cybercrime, what are the main obstacles in combating cybercrime and what actions countries should prioritise in order to increase awareness of risks from cybercrime in their population.

In what follows, Section II provides a literature and best practice review on developing cybersecurity awareness campaigns and existing efforts in Africa. Section III provides a brief overview of the CMM and the CMM methodology when deployed in a country. Section IV describes the results from the CMM reviews in six African countries and our analysis of the qualitative data obtained from focus groups during these reviews. As this paper concentrates on Cybersecurity Awareness, which is one component of the CMM, only the results of this component will be discussed. No countries will be referenced, but a general overview of the outcome will be described. Section V discusses the results of our analysis and Section VI concludes the paper.

II. Cybersecurity awareness raising campaigns

According to the UK Her Majesty's Government (HMG) Security Policy Framework [13], it is government's role to raise cybersecurity awareness within a country. "*People and behaviours are fundamental to good security. The right security culture, proper expectations and effective training are essential. Everyday actions and the management of people, at all levels in the organisation, contribute to good security. A strong security culture with clear personal accountability and a mature understanding of managing risk, responsibility and reputation will allow the business to function most effectively*".

Awareness presentations are "*intended to allow individuals to recognize IT security concerns and respond accordingly. In awareness activities, the learner is the recipient of information, whereas the learner in a training environment has a more active role. Awareness relies on reaching broad audiences with attractive packaging techniques. Training is more formal, having a goal of building knowledge and skills to facilitate the job performance* [14].

Awareness is used to stimulate, motivate, and remind the audience what is expected of them [15]. This is an important aspect of cybersecurity policy or strategy because it enhances the knowledge of users about security, changes their attitude towards cybersecurity, and their behaviour patterns.

A. Developing cybersecurity awareness raising campaigns

There is an abundance of best practice approaches describing principles in designing and implementing an awareness-raising campaign. Little emphasis, however, was put on how to strategically decide the areas where awareness campaigns should focus. National Institute of Standards and Technology (NIST) [16] is one of the pioneers in this field. Their framework provides three alternatives on how organisations should be structured, detailing for each category the processes for an effective and efficient campaign.

For all three approaches, namely centralised, partially decentralised and fully decentralized, NIST provides information on how a 'needs assessment' should be conducted; a strategy should be developed; an awareness training program be designed; and an awareness program be implemented. The key criteria to decide which approach an organisation should adopt are the size of the organisation, similarities in missions between different departments, knowledge of the topics into question and how spread the geographical area where campaigns will be implemented is.

Focusing on the design and implementation of awareness-raising campaigns, literature suggests that successful awareness campaigns need to be a 'learning continuum' [15], commencing from awareness, evolving to training and resulting in education. According to Organisation of American States (OAS) [17], it is of paramount importance that stakeholders from the public and private sector, Non-profit Government Organisations (NGOs), and technology and finance corporations to be involved. Once stakeholders are identified, the next steps in the OAS model provide instructions on how to define the goals of the campaign, the audience it targets and the strategy via which the campaign will be implemented.

Even by following best practise, several difficulties exist when it comes to creating a successful campaign: a) not understanding what security awareness really is; b) a compliance awareness program does not necessarily equate to creating the desired behaviours; c) usually there is lack of engaging and appropriate materials; d) usually there is no illustration that awareness is a unique discipline; e) there is no assessment of the awareness programmes [18]; f) not arranging multiple training exercises but instead focusing on a specific topic or threat does not offer the overall training needed [19].

Perceived control and personal handling ability, the sense one has that he/she can drive specific behaviour, has also been found to affect the intention of behaviour but also the real behaviour [20]. Culture is another important factor for consideration when designing education and awareness messages [21] as it can have a positive security influence to the persuasion process. Moreover, even when people are willing to change their behaviour, the process of learning a new behaviour needs to be supported [21].

Messages and advertisements are usually preferred when they match the cultural theme of the message recipient. Overall, a campaign should use simple consistent rules of behaviour that people can follow. This way, their perception of control will lead to better acceptance of the suggested behaviour. Moreover, even when people are willing to change their behaviour, the process of learning a new behaviour needs to be supported [21].

B. Cybersecurity awareness campaigns in Africa

A review in cybersecurity policies in African countries [22] shows that awareness raising is key issue either as a separate factor or as part of the role of the proposed National CSIRT. A cybersecurity policy and strategy may not be in place yet for all countries in Africa. However, there are already a number of organisations that have identified the need for continental coordination and increased cybersecurity awareness including the African Information Society Initiative (UNECA/AISI) [23], The Internet Numbers Registry for Africa (AfriNIC) [24], ITU/GCA [25], Interpol, The Southern African Development Community (SADC) [26] and ISG-Africa [27].

There are existing efforts in Africa such as the ISC Africa [28]. This is a coordinated, industry and community-wide effort to inform and educate Africa's citizens on safe and responsible use of computers and the Internet, so that the inherent risks can be minimised and consumer trust can be increased. Also, Parents' Corner Campaign [29] is intended to co-ordinate the work done by government, industry and civil society. Its objectives are to protect children, empower parents, educate children and create partnerships and collaboration amongst concerned stakeholders.

Recently Facebook has also announced partnerships with over 20 non-governmental organisations and official agencies from the DRC, Ghana, Kenya, Nigeria and South Africa in support of Safer Internet Day (SID) marked on 6 February [30]. SID advocates making the internet safer, particularly for the youth, and is organised by the joint Insafe-INHOPE network with the support of the European Commission and funded by the Connecting Europe Facility programme (CEF).

Usually, most of official awareness-campaign sites include advice, which usually comes from security experts and service providers, who monotonically repeat suggestions such as use strong passwords. Such advice pushes responsibility and workload for issues that should be addressed by the service providers and product vendors onto users. One of the main reasons why users do not behave optimally is that security systems and policies are often poorly designed [31]. There is a need to move from awareness to tangible behaviours.

C. *Cybersecurity awareness raising for executives*

The view that executives are often not sufficiently prepared to handle cybersecurity risk has raised concerns in boardrooms nationwide and globally. Even if companies increase their investments in security, we see more and more serious cyber attacks. The main concern is whether executives are prepared to make the right cybersecurity investment decisions and develop effective cybersecurity strategies [32].

Most executives realize the threat cyber risk represents to their organisation and to the nation's economy and are being found liable for cyber systems failures. Governing agencies are taking regulatory action against boards and management with the full support of the courts [33].

To improve the situation, companies need to address two issues. First, directors need to have basic training in cybersecurity that addresses the strategic nature, scope, and implications of cybersecurity risk. Second, top management needs to provide meaningful data about not just the state of data security as defined by viruses quarantined or the number of intrusions detected, but also about the resilience of the organisation's digital networks [32].

Developing a common language for management and corporate directors to discuss cybersecurity issues is also important. Digital security specialists, like all subject-area experts, must be able to communicate effectively with board members and other leaders. Information security executives must be capable to present information at a level and in a format that is accessible to non-technical corporate directors [32].

Both management and directors need to be aware of:

1) the limitations of security (no practical cybersecurity strategy can prevent all attacks) and
2) the need for resilience (strategies to sustain business during a cyberattack and to recover quickly in the aftermath of a breach).

This means that having strategies which will ensure sustainability of business during a cybersecurity breach and quick recovery in its aftermath, is important. Networks constantly change, so tracking cyber risks and vulnerabilities over time and adapting accordingly is essential [32].

Additionally, the involvement by business executives ensures that possible adverse impacts from security incidents are viewed from a bottom-line as well as from an asset valuation perspective [34]. In response to the gaps mentioned above, executives follow two different paths of cyber governance. First, they add a cybersecurity expert to the board and second, they assess the cybersecurity maturity of the organization or nation against accepted standards such as NIST [34].

In Africa, on the issue of who attends cybersecurity awareness training, the responses showed that although 70 percent of core business staff in organisations attended cybersecurity awareness training, and there is significant attendance by other groups of participants, such as supervisors/functional managers (59 percent) and middle management (57 percent) [35].

According to the ISACA State of cybersecurity report published in 2017 [36], globally just 21 percent of chief information security officers (CISO) report to the chief executive officer (CEO) or the board, while 63 percent report through the chief information officer (CIO). This latter reporting structure, which is even more common in Africa, positions security as a technical issue rather than a business concern, reducing the scope of action and effectiveness of any cybersecurity initiatives.

III. The Cybersecurity Capacity Maturity Model for Nations (CMM)

The CMM developed by the Global Cybersecurity Capacity Centre (GCSCC) [12] at the University of Oxford is a comprehensive framework which assesses the cybersecurity capacity maturity of capabilities which are fundamental to building resilience of a country over 5 different dimensions: 1) Cybersecurity Policy and Strategy; 2) Cyber Culture and Society; 3) Cybersecurity Education, Training and Skills; 4) Legal and Regulatory Frameworks; 5) Standards, Organisations, and Technologies.

Every Dimension consists of a number of Factors which describe what it means to possess cybersecurity capacity. Each Factor is composed of a number of Aspects that structure the Factor's content. Each Aspect is composed of a series of indicators within five stages of maturity. These indicators describe the steps and actions that must be taken to achieve or maintain a given stage of maturity in the aspect/factor/dimension hierarchy. These 5 maturity stages are: 1) Start up; 2) Formative; 3) Established; 4) Strategic; 5) Dynamic. The progressive nature of the model assumes that lower stages have been achieved before moving to the next.

The five stages are defined as follows:

1) Start-up: at this stage either no cybersecurity maturity exists, or it is very embryonic in nature. There might

be initial discussions about cybersecurity capacity building, but no concrete actions have been taken. There is an absence of observable evidence of cybersecurity capacity at this stage.

2) Formative: some aspects have begun to grow and be formulated, but may be ad-hoc, disorganised, poorly defined - or simply new. However, evidence of this aspect can be clearly demonstrated.
3) Established: the indicators of the aspect are in place, and functioning. However, there is not well thought-out consideration of the relative allocation of resources. Little trade-off decision-making has been made concerning the relative investment in this aspect. But the aspect is functional and defined.
4) Strategic: at this stage, choices have been made about which indicators of the aspect are important, and which are less important for the particular organisation or state. The strategic stage reflects the fact that these choices have been made, conditional upon the state's or organisation's particular circumstances.
5) Dynamic: at this stage, there are clear mechanisms in place to alter strategy depending on the prevailing circumstances such as the technological sophistication of the threat environment, global conflict or a significant change in one area of concern (e.g. cybercrime or privacy). Dynamic organisations have developed methods for changing strategies in-stride. Rapid decision-making, reallocation of resources, and constant attention to the changing environment are features of this stage.

The assignment of maturity stages is based upon the evidence collected, including the general or average view of accounts presented by stakeholders, desktop research conducted and the professional judgment of GCSCC research staff. Using the GCSCC methodology recommendations are provided as to the next steps that might be considered by a nation to improve cybersecurity capacity.

In this paper, we focus on the factor 'Cybersecurity Awareness Raising' (shown in detail in Figure 3 and Figure 4 in the Appendix section). The aspects, within this factor are 'Awareness Raising Programmes' and 'Executive Awareness Raising' with various indicator specialisations for every maturity stage. The aspect 'Awareness Raising Programmes' examines the existence of a national coordinated programme for cybersecurity awareness raising, covering a wide range of demographics and issues, while the aspect 'Executive Awareness Raising' examines efforts raising executives' awareness of cybersecurity issues in the public, private, academic and civil society sectors, as well as how cybersecurity risks might be addressed. The CMM model was developed by conducting systematic reviews on best practice approaches which are publicly available, as well as consulting experts from various disciplines.

According to the CMM, the aspect 'Awareness Raising Programmes' will be measured to be on a Start-up stage of maturity if the indicator 'The need for awareness of cybersecurity threats and vulnerabilities across all sectors is not recognised, or is only at initial stages of discussion' is met and indicators from the next level are absent. This stage of maturity is comprised only by this one indicator. In order to be at the formative stage of maturity, the next two relevant Indicators must be met. As seen in Appendix, the number of Indicators may differ between maturity stages. In order to elevate a country's cybersecurity capacity maturity, all of the indicators within a particular stage will need to be fulfilled.

The first version of the CMM was finalized in 2014 [37] and the revised edition was published in 2017 [12]. So far, the CMM has only been deployed on the national level (rather than at the company/enterprise level), and 54 countries have been fully evaluated through engagement and collaboration with the host country.

A. *The CMM implementation methodology*

The process by which a (host) country is assessed is as important as the model itself. This process actually forms the basis of the whole review/research methodology on which a country review is based. This process forms and motivates the underlying research and application methodology of a CMM review and provides scientific validity to the results coming from a review the process guarantees the validity and verification of the outputs. The first step of a country review is to identify a country host that is the body which is responsible for all logistical arrangements in the host country.

The CMM employs a focus group methodology since it has been acknowledged to offer a rich set of data compared to other qualitative approaches [38], [39], [40]. Like interviews, focus groups are an interactive methodology with the advantage that during the process of collecting data and information diverse viewpoints and conceptions can emerge. It is a fundamental part of the method that rather than posing questions to every interviewee, the researcher(s) should facilitate a discussion between the participants, encouraging them to adopt, defend or criticise different perspectives [41].

Stakeholders are identified based on their expertise in each one of the components of every Dimension of the CMM. There will be 9 stakeholder (cluster) groups planned for each CMM review (Table I). For example, for the specific factor Cybersecurity Awareness Raising, academia, civil society groups, internet governance experts, one or more representatives from Universities, and Internet societies will be invited to take part in the focus group. This group of representatives are called a stakeholder (cluster) group.

While these stakeholder clusters are useful as a guiding structure for conducting the assessments, selecting the participants in country to engage with the CMM would be challenging without a thorough understanding of the complex network of domestic actors. Therefore, in order to gain a more thorough understanding of the domestic context in which the model will be applied, we worked alongside either international organisations with knowledge of such domestic dynamics or directly with a host ministry or organisation within a country. Additionally, we complemented that process by conducting desk research on stakeholders that might provide valuable insight into domestic cybersecurity capacity and then make recommendations to the host team.

Focus group sessions are led by the CMM Review Team. The assessments are typically conducted over the course of three to four days for 1.5 to 2 hours for each cluster. The host team would indicate in the invitation which cluster the participant will engage with. The CMM review team will lead the discussion facilitation with the different participants in order to aid the selection of stages of maturity in each category,

TABLE I. Stakeholder groups participating in focus groups

Stakeholder (cluster) groups
Academia, Civil Society Groups, and Internet Governance Representatives
Criminal Justice and Law Enforcement
Critical National Infrastructure
CSIRT and IT Leaders from Government and the Private Sector
Defense and Intelligence Community
Government Ministries
Legislators/Policy owners
Private Sector and Business
Cyber Task Force

factor and dimension. Due to the depth and nuance of cybersecurity capacity within the CMM, it would be impossible to go through the entire model with each stakeholders cluster. Therefore, each cluster responds to two dimensions of the CMM, depending on their relevant expertise. The participants should be able to provide or indicate evidence supporting their selection, so that subjective responses are minimised. If a country does not fulfill all of the criteria within a particular stage of maturity, the previous stage is selected, while noting which particular elements of capacity are missing for achieving the proceeding stage. This nuance is key, as it allows for more flexibility in understanding existing capacity, rather than assigning a stage of maturity that does not account for subtle variations.

Additionally, the CMM review team would be facilitating the discussion, trying to keep the discussion on track without influencing the opinions of the group, but also avoiding only a few of the participants dominating the discussion [42], [43].

The consultations result in a comprehensive report indicating the relevant Maturity stage for all Factors and Aspects in all Dimensions. A comprehensive set of Recommendations is also provided to indicate to the country how to improve capacity and progress onto the next stages of maturity.

IV. CMM RESULTS FOR AWARENESS RAISING IN AFRICA

In Africa, a team from the GCSCC has reviewed and evaluated 6 countries based on the CMM and following the methodology described in Section III. These countries were selected for a review at the time because they were in the process of drafting a national cybersecurity strategy. Therefore, the review would assist this process. These reviews have been conducted during the period June 2015 to January 2018.

Regarding the aspect 'Awareness Raising Programs' and 'Executive Awareness Raising', 12 focus groups have been conducted in total. The stakeholders who participated in the focus groups are from the following sectors: Public Sector Entities; Legislators/Policy Makers; Criminal Justice and Law Enforcement; Armed Forces; Academia; Civil Society; Private Sector; CSIRT and IT Leaders from Government and the Private Sector; Critical national infrastructure; Telecommunications Companies; and Finance Sector. Each focus group session had approximately 10-15 stakeholders and lasted on average 2 hours.

In order for the stakeholders to provide evidence on how many indicators have been implemented by a nation and to determine the maturity level of every aspect of the model, a consensus method is used to drive the discussions within sessions. During focus groups, researchers use semi-structured questions to guide discussions around indicators. During these discussions, stakeholders should be able to provide or indicate evidence regarding the implementation of indicators, so that subjective responses are minimised.

A. Analysis of maturity level data

Figure 1 illustrates the results from the six African countries of the CMM maturity levels for the cybersecurity awareness raising dimension. Three countries have been identified to be at a start-up stage of maturity, two countries have been identified at a formative stage and one at a start-up stage with few of the indicators from the formative stage of maturity being present (this is denoted as from start-up to formative in the diagram).

The results clearly indicate that the majority of examined countries in Africa are identified at a start-up stage of maturity. This translates into lack of a national programme for cybersecurity awareness raising. The need for awareness of cybersecurity threats and vulnerabilities across all sectors is not recognised, or is only at initial stages of discussion. Furthermore, awareness raising programmes (if existing) may be informed by international initiatives but are not linked to a national strategy.

Finally, it was identified that awareness raising programmes, courses, seminars and online resources might be available for target demographics from public, private, academic, and/or civil sources, but no coordination or scaling efforts have been conducted. In the next section, we provide further details, based on our qualitative analysis, on these initial findings.

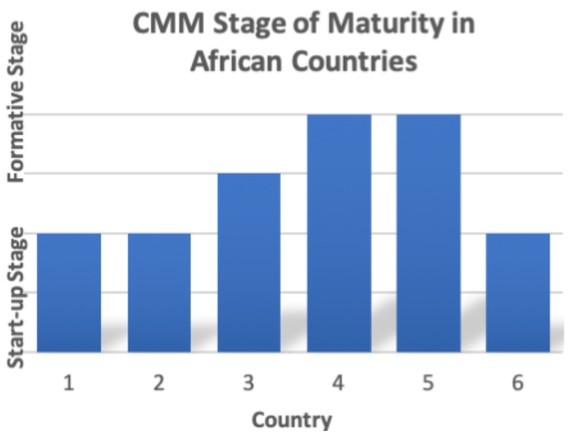


Figure 1. CMM results from six African Countries

B. Qualitative analysis of results

We have transcribed all the recordings from focus groups and conducted a thematic analysis on the qualitative data for each country. We adopted a blended approach (a mix of deductive and inductive approach) to analyse focus group data and used the indicators of the CMM as our criteria for a deductive analysis. The inductive approach is based on 'open coding' meaning that the categories or themes are freely created by the researcher, while the deductive content analysis requires the prior existence of a theory to underpin the classification process.

Excerpts that did not fit into themes were further analysed to highlight additional issues that stakeholders might have raised during the focus groups or to inform our understanding on what the next steps should be for a country.

Overall, we identified eight themes in our qualitative analysis for every country. Four themes were based on the aspects described in the CMM model and four themes emerged from the inductive approach. The themes from the inductive approach pertained information on what actions African countries should implement next. Since these eight themes were common for all six countries, we merged the excerpts for each theme from every country. We further examined these excerpts to identify common areas which hindered progress in cybersecurity awareness raising as well as key actions which countries should implement next to improve their cybersecurity posture in awareness raising.

More specifically, the four main themes that emerged from the deductive approach are: a) the lack of national level programmes; b) the existence of ad-hoc initiatives; c) the relationship between ICT literacy (the ability to use digital technology and tools) and awareness and d) executive awareness. In a similar vein, the inductive approach identified four themes which revolved around the same concepts described in the deductive analysis; the difference being that excerpts in the inductive themes pertained information about recommendations and next steps.

Lack of national level programmes
Existence of ad-hoc initiatives
Lack of ICT literacy
Executive awareness

Figure 2. Themes from Inductive and Deductive Analysis

1) Deductive theme analysis: For all countries, it is evident that a national programme for cybersecurity awareness raising is absent. In many cases, stakeholders mentioned that '*lack of awareness is an institutional problem, not a user problem*' and also that '*a proper cyber awareness programme is needed*'. The importance for such a programme was acknowledged across the various stakeholders in all countries reviewed in Africa. A main hindrance for the implementation of a national programme is the general lack of cybersecurity awareness outside the technical communities, which stakeholders pointed that its origin is the low ICT literacy in the population of these countries.

Concerns were also expressed regarding the security of nationwide projects involving big volumes of personal data. Participants, mentioned that '*cybersecurity awareness, in particular in relation to the protection of personal data, needs to be prioritised for such projects*'.

It was further emphasised that awareness-raising programmes need to be developed alongside other capacity enhancements, such as incident response, training for cybersecurity educators, and national and organisational cybersecurity policies.

Regarding the initiatives theme, there are ad-hoc initiatives in cybersecurity awareness raising that are supported by various institutions. These are being offered from various organisations such as Facebook while the financial sector, civil society and academia organise programmes for schools to raise awareness. According to a stakeholder, '*some telecommunication companies and banks are engaged in awareness activities which includes messages via the media, directed to end-users, e.g. password security*'.

These initiatives, however, are not yet coordinated at the national level. Therefore, it was widely recognised that a more centralised awareness-raising programme would greatly expand a fundamental understanding of cybersecurity capacity.

Often, civil society actors initiate efforts into targeted cybersecurity awareness-raising. Different stakeholders agree that a '*common ground*' between government, private sector and civil society could enable the proliferation of awareness raising to the broader society. Moreover, often it was mentioned that the government needs to work alongside existing efforts in academia to ensure that new initiatives capitalise from the academic experience. Such synergy is critical to ensure that awareness-raising efforts are efficient and effective.

As often mentioned by stakeholders '*people trust social media and do not expect that someone will harm them, we are brothers!*'. A stakeholder also noted that '*It is common in African countries that mobile phones are used to access the Internet, use social media, for e-banking services etc. but people who use online services are not aware of risks*'. Often, lack of awareness leads to a sense of '*blind trust online*'. A stakeholder noted that '*users trust social media and think that their information is secure, although often websites are still insecure*'.

Another interesting theme that emerged from the analysis of data is the low ICT literacy rate in Africa. Stakeholders indicated that awareness of the effective use of ICT is still only gaining initial traction and that security is seen as only relevant once ICT and Internet literacy is sufficient. Stakeholders suggested that '*integrating cybersecurity awareness efforts into ICT literacy courses could provide an established vehicle for cybersecurity awareness-raising campaigns*'.

Regarding the theme revolving around awareness among executives, both in public and private sectors, cybersecurity awareness is very limited, which is one reason why cybersecurity awareness raising is not yet perceived as a priority. This has been identified as an important gap, as executives are usually the final arbiters on investment into security.

Some major telecommunications companies conduct internal awareness raising training across all levels, but there is not a publicly available initiative which targets executives. As mentioned by a stakeholder, '*the reason for that is that there is limited awareness for cybersecurity threats and risks in the private sector overall, unless in major international organi-*

sations, in particular in the banking and telecommunications sectors which face strategic implications of cybersecurity'.

It was commonly stated that there is a sharp disconnect between the terminology and priorities of the engineers working in IT systems and security, and those at the higher level seeking to make sound business decisions based on risk.

2) Inductive Theme Analysis: Stakeholders mentioned during focus group sessions that '*aspects of cybersecurity need to be introduced in the school curricula and improve ICT literacy*'. It was also noted that '*even in universities, people are not aware of the possible risks and procure without following standards*'. Integrating cybersecurity awareness efforts into ICT literacy courses could provide an established vehicle for cybersecurity awareness campaigns.

Culture is another factor that can impact the effectiveness of cybersecurity awareness programmes. As seen above, the collectivist cultural aspect that characterises off-line behaviour in Africa, is also pertained in online behaviour [44].

Currently, due to the lack of national level awareness programmes, '*being hacked brings awareness usually*' as a stakeholder noted. Therefore, the development of such a programme with specified target groups focusing on most vulnerable users is identified as necessary [45]. Also, appointing a designated organisation (from any sector) to lead the cybersecurity awareness raising programme and engaging relevant stakeholders from public and private sectors in the development and delivery of the awareness raising programme is crucial. As stakeholders mentioned in one of the reviews in Africa '*The government realises that lack of awareness is crucial and recognises the importance of a multi-stakeholder approach towards this goal*'. Moreover, it was noted that '*People access social media through their smart phones and security is the last thing on their mind and that convenience is usually coming first*'.

Stakeholders mentioned that '*even though the telecommunications sector has started to place emphasis on cybersecurity standards compliance, small- and medium-sized enterprises (SMEs) are mostly not worried about adopting and implementing standards*'. An area of particular concern for SMEs is that of encouraging good security behaviour by employees [46], [47], [48]. Developing a strong security culture could address many of the behavioural issues that underpin data breaches in such companies [49], [50]. Here, the development of cybersecurity skills involves addressing digital threats using technology and complementary factors including policy guidelines, organisational processes, and education and awareness strategies. By having an organisational security setting where employees intuitively protect corporate information assets, SMEs could improve their overall security [51].

Regarding the executive awareness raising aspect, developing a dedicated awareness raising programme for executives within the public and private sectors is essential. A stakeholder noted that '*different levels of authority need different kind of awareness in order to promote collaboration as well*'. Currently, executives and management are being called upon to address cyber risk alongside other risks that businesses face.

V. Discussion

Reflecting on the results presented in Section 4, the lack of a central authority, which is crucial in all modes of operation as presented by NIST model [16], is evident. The absence of such authority prohibits the execution of holistic 'needs assessments', amplifies the difficulties in prioritising the areas in which campaigns should be implemented and renders the design of ad-hoc campaigns being created by a limited number of stakeholders. It is imperative that African countries allocate an authority to conduct a national needs assessment, identify the areas where campaigns should focus first, develop a strategy for how these campaigns will be designed and implemented, and coordinate the ad-hoc efforts of different stakeholders.

The main objectives for cybersecurity in Africa and globally is online security by improving knowledge, capabilities and decision making. In order to enable the full benefits of cyberspace to all African countries, investing in human capacity development of all the citizens is vital.

Focusing on the design and implementation of awareness-raising campaigns, literature suggests that successful awareness campaigns need to be a 'learning continuum' [52], commencing from awareness, evolving to training and resulting in education. Our results highlight the need of African countries to involve stakeholders who are established in all the aforementioned sectors. Our analysis suggests that the audience of the campaigns should prioritise smartphone users, employees of SMEs and board members. The goals should be to communicate the risks from cybercrime, illustrate the need for better security controls and practices, and the need to establish a chief information security officer (CISO), respectively.

This means that businesses and government agencies should start to take steps to increase their awareness and understanding of cybersecurity with a view of the potential impact on overall business performance. Lack of boardroom expertise makes it challenging for directors and councilors to effectively oversee management's cybersecurity activities.

Cybersecurity awareness should reach all levels and inform all users of the internet - from vulnerable, school-going children to families, industry, critical national infrastructures, governments and the African continent with its unique needs [53], [54], [55]. This will enhance resilience against cybercrimes and attacks and inform African policy development.

If a country has already developed a national cybersecurity strategy, or is working towards that goal, then linking the development of the programme to that Strategy will facilitate the coordination of different capacities towards the development of the programme and its effective implementation.

Regarding the implementation of these campaigns, there are several organisations with ad-hoc initiatives that could facilitate the design and implementation of cybersecurity campaigns, such as ISC Africa [28] and Parents corner [29]. To conclude, it is worth mentioning that the timing for the development of these campaigns coincides with efforts in African countries to increase ICT literacy. As our findings underline, it is a unique opportunity for all African countries to combine ICT development with cybersecurity awareness. In contrast to western societies, where cybersecurity campaigns endeavour to change the norms on how users currently behave online (behaviour shaped since the inception of the Internet), campaigns in Africa can reflect on best practice and create new norms which will encompass cybersecurity requirements.

Creating a single online portal linking to appropriate cybersecurity information and disseminating it via the cybersecurity awareness programme can also enhance the effectiveness of such a programme. Moreover, enacting evaluation measurements to study the effectiveness of an awareness programme will not only lead to the assessment of the programme but also identify possible gaps that need to be addressed [45].

Moreover, enacting evaluation measurements to study effectiveness of the awareness programme will not only lead to the assessment of the programme but also identify possible gaps that need to be addressed [16], [45].

VI. Conclusions and future work

Several reports are depicting a bleak picture regarding the unprecedented increase of cybercrime in Africa. Yet, efforts to raise cybersecurity awareness in the general public and executives are in an embryonic stage. In this paper, we conducted twelve focus groups in six different African countries to shed light into the current situation and identify critical actions which can significantly decrease the success rate of cybercriminals.

Our results suggest that all six African countries do not possess a national programme for raising awareness, there are extremely low ICT literacy levels which hinder any design of cybersecurity campaigns and that executive members in organisations myopically underestimate the problem. To better defend against cybercrime, African countries need to establish a central authority which will coordinate the existing ad-hoc efforts in awareness campaigns and identify the target groups of these campaigns with particular focus on SMEs, mobile-phone users and executive board members. We believe that African countries have a unique opportunity to combine ICT literacy campaigns with cybersecurity principals and shape the norms of the society towards best practice.

By improving knowledge, cybersecurity can also be enhanced as well as capabilities and decision making. In Africa, but also at the global level the full benefits of cyberspace can be enabled by investing in human capacity development. Executives are also users and they need also to be aware of how cyber risks can threaten their assets in order to make effective strategy decisions.

At a national and an organisational level, strategies need to be developed linked to awareness campaigns with clear objectives, design and implementation processes and coordination of the ad-hoc efforts of different stakeholders. As part of our future work, we intend to explore the effectiveness of a national coordinated cybersecurity awareness programme and how it relates to the actual security posture of a country. Our future work will be based on data from developed countries where the CMM has already been applied, as well as on data collected by other international organisations such as the International Telecommunication Union - GCI [56], Australian Strategic Policy Institute - ASPI [57], The Potomac Institute for Policy Studies - Cyber Readiness Index [58] and World Economic Forum - Global Competitive Index [59].

*Acknowledgment

The authors would like to thank Ms. Eva Ignatuschtschenko, Ms. Eva Nagyfejeo, Mr. Taylor Roberts and Ms. Carolin Weisser from the GCSCC for conducting field work and data collection. We are also immensely grateful to Prof. Sadie Creese and Prof. Michael Goldsmith for their comments on an earlier version of the manuscript.

References

[1] M. Bada, B. Von Solms, and I. Agrafiotis, "Reviewing national cybersecurity awareness in africa: An empirical study," in The Third International Conference on Cyber-Technologies and Cyber-Systems. Thinkmind Digital Library, IARIA, 2018.

[2] Trend Micro, "Is there a budding west African underground market?" 2017, uRL: https://www.trendmicro.com/vinfo/us/security/news/cybercrime-and-digital-threats/west-african-underground, [Last accessed: 20 Jan 2019].

[3] T. Oladipo, "Cyber-crime is Africa's 'next big threat', experts warn," 2015, uRL: http://www.bbc.co.uk/news/world-africa-34830724,[Last accessed: 20 Jan 2019].

[4] Serianu, "Africa cyber security report," 2016, uRL: http://www.serianu.com/downloads/AfricaCyberSecurityReport2016.pdf, [Last accessed: 20 Jan 2019].

[5] A. Okuku, K. Renaud, and B. Valeriano, "Cybersecurity strategy's role in raising kenyan awareness of mobile security threats," Information & Security, vol. 32, no. 2, 2015, p. 1.

[6] B. van Niekerk and M. Maharaj, "Mobile security from an information warfare perspective," in 2010 Information Security for South Africa. IEEE, 2010, pp. 1–7.

[7] A. Okeowo, "Smss used as a tool of hate in Kenya," 2008, uRL: https://mg.co.za/article/2008-02-19-smss-used-as-a-tool-of-hate-in-kenya, [Last accessed: 25 Jan 2019].

[8] Symantec, "Cyber crime and cyber security trends in Africa," 2016, uRL: https://www.thehaguesecuritydelta.com/media/com_hsd/report/135/document/Cyber-security-trends-report-Africa-en.pdf, [Last accessed: 25 Jan 2019].

[9] T. Skye, "The last mile in IT security: Changing user behaviors," 2016, uRL: https://www.wombatsecurity.com/changing-end-users-behavior-espion, [Last accessed: 25 Jan 2019].

[10] Wombat Security Technologies (Wombat) and the Aberdeen Group, "African union cybersecurity profile: Seeking a common continental policy," 2016, uRL: https://jsis.washington.edu/news/african-union-cybersecurity-profile-seeking-common-continental-policy/, [Last accessed: 2 Feb 2019].

[11] M. M. Waldrop, "How to hack the hackers: The human side of cybercrime," Nature News, vol. 533, no. 7602, 2016, p. 164.

[12] Global Cyber Security Capacity Centre, "Cybersecurity capacity maturity model for nations (cmm): Revised edition," 2017, uRL: https://www.sbs.ox.ac.uk/cybersecurity-capacity/content/cybersecurity-capacity-maturity-model-nations-cmm-0, [Last accessed: 25 Jan 2019].

[13] HMG, "Security policy framework," 2016, uRL: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/316182/Security_Policy_Framework_web_April_2014.pdf, [Last accessed: 25 Jan 2019].

[14] National Institute of Standards and Technology, "Building an information technology security awareness and training program," 2003, URL: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-50.pdf, [Last accessed: 25 Feb 2019].

[15] T. R. Peltier, "Implementing an information security awareness program," Information Systems Security, vol. 14, no. 2, 2005, pp. 37–49.

[16] National Institute of Standards and Technology, "Framework for improving critical infrastructure cybersecurity," 2014, uRL: https://www.nist.gov/sites/default/files/documents/cyberframework/cybersecurity-framework-021214.pdf, [Last accessed: 25 Feb 2019].

[17] Organization of American States, "Cybersecurity awareness toolkit," 2015, uRL: https://www.sbs.ox.ac.uk/cybersecuritycapacity/system/files/2015%20OAS%20%20Cyber%20Security%20Awareness%20Campaign%20Toolkit%20%28English%29.pdf, [Last accessed: 10 Feb 2019].

[18] B. Khan, K. S. Alghathbar, S. I. Nabi, and M. K. Khan, "Effectiveness of information security awareness methods based on psychological theories," African Journal of Business Management, vol. 5, no. 26, 2011, p. 10862.

[19] I. Winkler and S. Manke, “Reasons for security awareness failure,” CSO Security and Risk, 7.

[20] I. Ajzen, “Perceived behavioral control, self-efficacy, locus of control, and the theory of planned behavior,” Journal of applied social psychology, vol. 32, no. 4, 2002, pp. 665–683.

[21] M. W. Kreuter and S. M. McClure, “The role of culture in health communication,” Annu. Rev. Public Health, vol. 25, 2004, pp. 439–455.

[22] I. Dlamini, B. Taute, and J. Radebe, “Framework for an African policy towards creating cyber security awareness,” in Proceedings of Southern African Cyber Security Awareness Workshop (SACSAW), 2011.

[23] United Nations: Economic Commission for Africa, “The african information society initiative (aisi) - a decades perspective,” 2015, uRL: https://www.uneca.org/publications/african-information-society-initiative-aisi-decade%E2%80%99s-perspective, [Last accessed: 10 Feb 2019].

[24] AfriNIC, “The internet numbers registry for Africa,” 2018, uRL: https://www.afrinic.net, [Last accessed: 10 Feb 2018].

[25] International Telecommunication Union, “Towards a common future,” 2018, uRL: https://www.itu.int/en/action/cybersecurity/Pages/gca.aspx, [Last accessed: 10 Feb 2019].

[26] The Southern African Development Community, “Global cybersecurity agenda (gca),” 2018, uRL: http://www.sadc.int/, [Last accessed: 10 Feb 2019].

[27] Information Security Group of Africa, “Profile,” 2018, uRL: http://pressoffice.itweb.co.za/isgafrica/profile.html, [Last accessed: 10 Feb 2019].

[28] ISC, “Internet safety campaign,” 2018, uRL: http://iscafrica.net/, [Last accessed: 10 Feb 2019].

[29] Parents Corner, “Digital curfews — what are they & do your kids need one?” 2017, uRL: https://parentscorner.org.za, [Last accessed: 10 Feb 2019].

[30] M. Lunika, “Africa rallies in support of safer internet day,” 2018, uRL: http://www.itwebafrica.com/ict-and-governance/523-africa/242730-africa-rallies-in-support-of-safer-internet-day, [Last accessed: 10 Feb 2019].

[31] J. R. Nurse, S. Creese, M. Goldsmith, and K. Lamberts, “Guidelines for usable cybersecurity: Past and present,” in Cyberspace Safety and Security (CSS), 2011 Third International Workshop on. IEEE, 2011, pp. 21–26.

[32] R. A. Rothrock, J. Kaplan, and F. Van Der Oord, “The board’s role in managing cybersecurity risks,” MIT Sloan Management Review, vol. 59, no. 2, 2018, pp. 12–15.

[33] B. Barker, “Emerging cybergovernance discipline protects directors from new risks,” 2016, uRL: https://www.cybernance.com/emerging-cybergovernance-discipline-protects-directors-new-risks/, [Last accessed: 2 Feb 2019].

[34] T. Braithwaite, “Executives need to know: The arguments to include in a benefits justification for increased cyber security spending,” Information systems security, vol. 10, no. 4, 2001, pp. 1–14.

[35] Department of Telecommunications and P. Services, “Cybersecurity readiness report, south africa,” 2017, uRL: https://www.cybersecurityhub.gov.za/images/docs/Cyber-Readiness-Report.pdf, [Last accessed: 2 Feb 2019].

[36] ISACA, “State of cybersecurity,” 2017, uRL: https://cybersecurity.isaca.org/csx-resources/state-of-cyber-security-2017, [Last accessed: 2 Feb 2019].

[37] Global Cyber Security Capacity Centre, “Cyber security capability maturity model (cmm) v1.2,” 2015, uRL: https://www.sbs.ox.ac.uk/cybersecurity-capacity/content/cybersecurity-capacity-maturity-model-nations-cmm, [Last accessed: 25 Jan 2019].

[38] M. Williams, Making sense of social research. Sage, 2002.

[39] J. Knodel, “The design and analysis of focus group studies: A practical approach,” Successful focus groups: Advancing the state of the art, vol. 1, 1993, pp. 35–50.

[40] R. A. Krueger and M. A. Casey, Focus groups: A practical guide for applied research. Sage publications, 2014.

[41] J. Kitzinger, “Qualitative research: introducing focus groups,” Bmj, vol. 311, no. 7000, 1995, pp. 299–302.

[42] A. Gibbs, “Focus groups,” Social Research Update, vol. 19, no. 8, 1997, pp. 1–8.

[43] J. Kitzinger, “Introducing focus groups,” British Medical Journal, no. 311, 1995, pp. 299–302.

[44] H. C. Triandis, “Cultures and organizations: Software of the mind.” 1993.

[45] M. Bada and A. Sasse, “Cyber security awareness campaigns: Why do they fail to change behaviour?” arXiv preprint arXiv:1901.02672, 2014.

[46] V. Dimopoulos, S. Furnell, M. Jennex, and I. Kritharas, “Approaches to it security in small and medium enterprises.” in AISM, 2004, pp. 73–82.

[47] M. Taylor and A. Murphy, “Smes and e-business,” Journal of small business and enterprise development, vol. 11, no. 3, 2004, pp. 280–289.

[48] J. R. Nurse, S. Creese, M. Goldsmith, and K. Lamberts, “Trustworthy and effective communication of cybersecurity risks: A review,” in 2011 1st Workshop on Socio-Technical Aspects in Security and Trust (STAST). IEEE, 2011, pp. 60–68.

[49] A. Santos-Olmo, L. Sánchez, I. Caballero, S. Camacho, and E. Fernandez-Medina, “The importance of the security culture in smes as regards the correct management of the security of their assets,” Future Internet, vol. 8, no. 3, 2016, p. 30.

[50] B. Contos, “Cyber security culture is a collective effort,” CSO. Retrieved from, 2015.

[51] S. Dojkovski, S. Lichtenstein, and M. J. Warren, “Fostering information security culture in small and medium size enterprises: An interpretive study in australia.” in ECIS, 2007, pp. 1560–1571.

[52] E. Kritzinger, M. Bada, and J. R. Nurse, “A study into the cybersecurity awareness initiatives for school learners in South Africa and the UK,” in IFIP World Conference on Information Security Education. Springer, 2017, pp. 110–120.

[53] H. Twinomurinz, A. Schofield, L. Hagen, S. Ditsoane-Molefe, and N. A. Tshidzumba, “Towards a shared worldview on e-skills: A discourse between government, industry and academia on the ICT skills paradox,” South African Computer Journal, vol. 29, no. 3, 2017, pp. 215–237.

[54] E. Kritzinger, “Growing a cyber-safety culture amongst school learners in South Africa through gaming,” South African Computer Journal, vol. 29, no. 2, 2017.

[55] ——, “Short-term initiatives for enhancing cyber-safety within South African schools,” South African Computer Journal, vol. 28, no. 1, 2016, pp. 1–17.

[56] International Telecommunication Union, “Global Cybersecurity Index,” 2018, uRL: https://www.itu.int/en/ITU-D/Cybersecurity/Pages/GCI.aspx, [Last accessed: 25 Feb 2019].

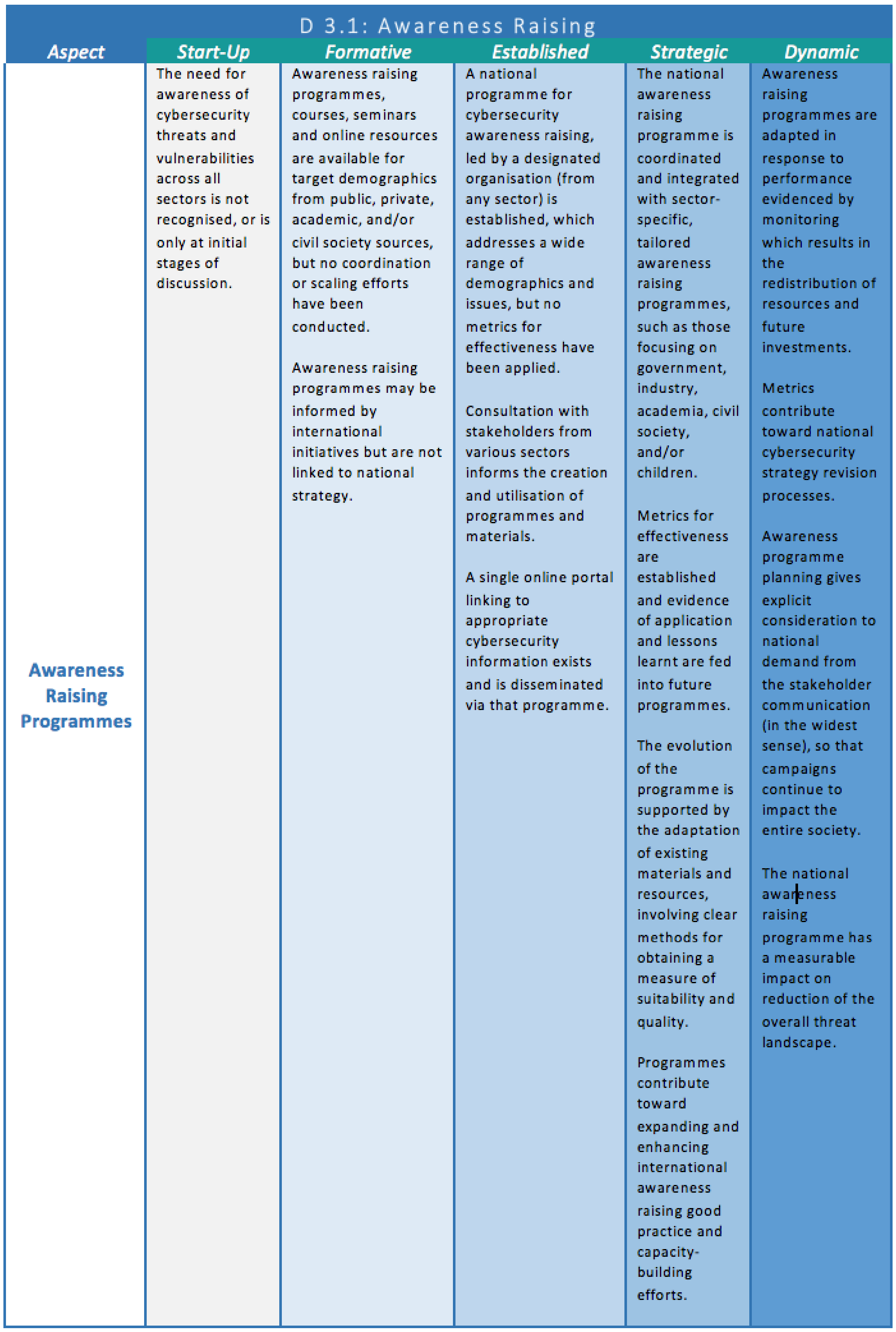
[57] Australian Strategic Policy Institute, “Cyber maturity in the Asia Pacific region,” 2017, uRL: https://www.aspi.org.au/, [Last accessed: 25 Feb 2019].

[58] The Potomac Institute for Policy Studies, “Cyber readiness index 2.0,” 2015, uRL: http://www.potomacinstitute.org/images/CRIndex2.0.pdf, [Last accessed: 15 Feb 2019].

[59] World Economic Forum, “The global competitiveness report 20172018,” 2018, uRL: http://reports.weforum.org/global-competitiveness-index-2017-2018/, [Last accessed: 15 Feb 2019].

APPENDIX

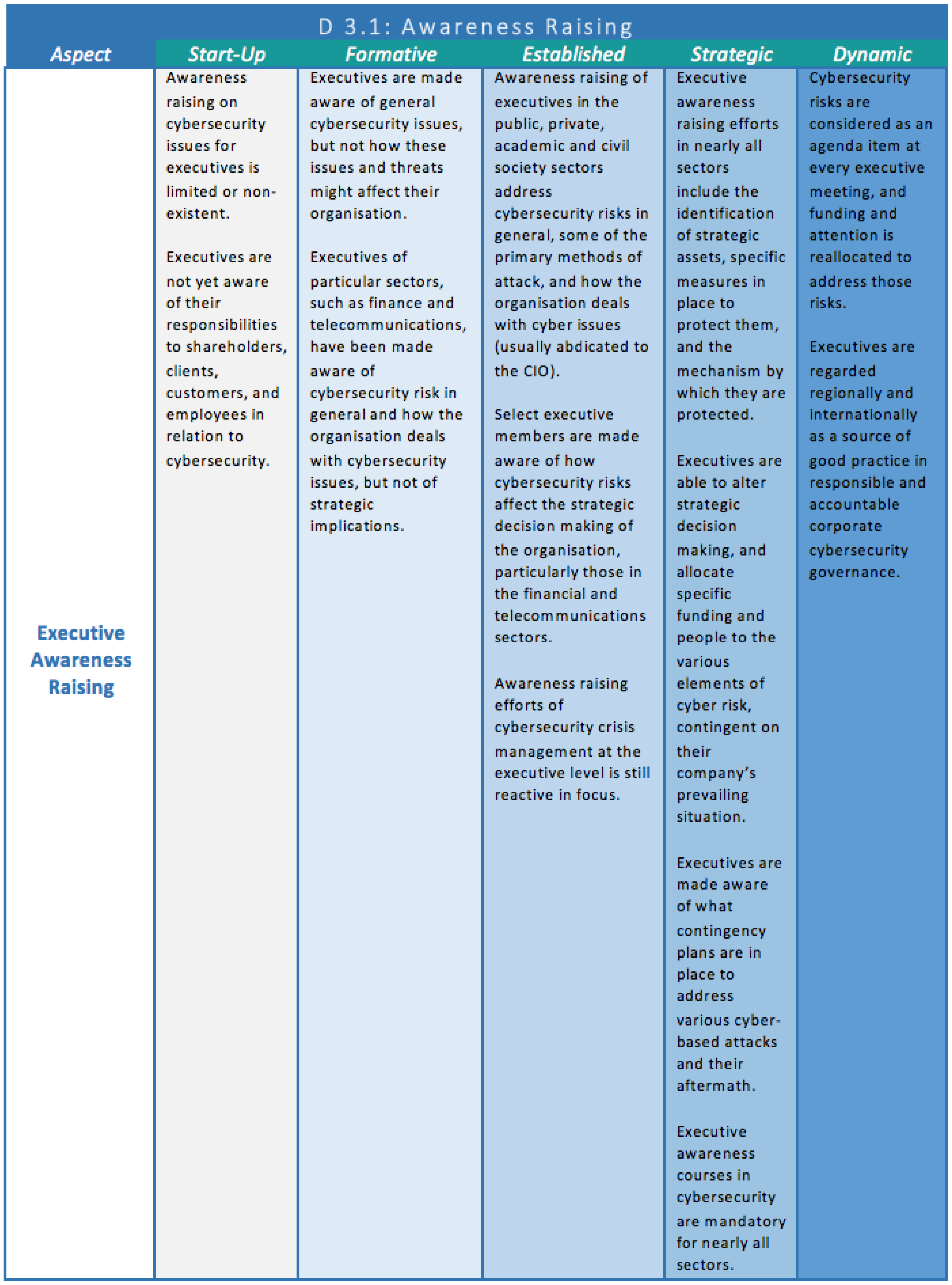
In this section we present the details of the capacity maturity model for dimension 3 used to analyse the results of the qualitative research.



D 3.1: Awareness Raising

Aspect	Start-Up	Formative	Established	Strategic	Dynamic
Awareness Raising Programmes	The need for awareness of cybersecurity threats and vulnerabilities across all sectors is not recognised, or is only at initial stages of discussion.	Awareness raising programmes, courses, seminars and online resources are available for target demographics from public, private, academic, and/or civil society sources, but no coordination or scaling efforts have been conducted. Awareness raising programmes may be informed by international initiatives but are not linked to national strategy.	A national programme for cybersecurity awareness raising, led by a designated organisation (from any sector) is established, which addresses a wide range of demographics and issues, but no metrics for effectiveness have been applied. Consultation with stakeholders from various sectors informs the creation and utilisation of programmes and materials. A single online portal linking to appropriate cybersecurity information exists and is disseminated via that programme.	The national awareness raising programme is coordinated and integrated with sector-specific, tailored awareness raising programmes, such as those focusing on government, industry, academia, civil society, and/or children. Metrics for effectiveness are established and evidence of application and lessons learnt are fed into future programmes. The evolution of the programme is supported by the adaptation of existing materials and resources, involving clear methods for obtaining a measure of suitability and quality. Programmes contribute toward expanding and enhancing international awareness raising good practice and capacity-building efforts.	Awareness raising programmes are adapted in response to performance evidenced by monitoring which results in the redistribution of resources and future investments. Metrics contribute toward national cybersecurity strategy revision processes. Awareness programme planning gives explicit consideration to national demand from the stakeholder communication (in the widest sense), so that campaigns continue to impact the entire society. The national awareness raising programme has a measurable impact on reduction of the overall threat landscape.

Figure 3. Dimension 3: Cybersecurity Education, Training and Skills for Awareness Raising Programmes



D 3.1: Awareness Raising					
Aspect	**Start-Up**	**Formative**	**Established**	**Strategic**	**Dynamic**
Executive Awareness Raising	Awareness raising on cybersecurity issues for executives is limited or non-existent. Executives are not yet aware of their responsibilities to shareholders, clients, customers, and employees in relation to cybersecurity.	Executives are made aware of general cybersecurity issues, but not how these issues and threats might affect their organisation. Executives of particular sectors, such as finance and telecommunications, have been made aware of cybersecurity risk in general and how the organisation deals with cybersecurity issues, but not of strategic implications.	Awareness raising of executives in the public, private, academic and civil society sectors address cybersecurity risks in general, some of the primary methods of attack, and how the organisation deals with cyber issues (usually abdicated to the CIO). Select executive members are made aware of how cybersecurity risks affect the strategic decision making of the organisation, particularly those in the financial and telecommunications sectors. Awareness raising efforts of cybersecurity crisis management at the executive level is still reactive in focus.	Executive awareness raising efforts in nearly all sectors include the identification of strategic assets, specific measures in place to protect them, and the mechanism by which they are protected. Executives are able to alter strategic decision making, and allocate specific funding and people to the various elements of cyber risk, contingent on their company's prevailing situation. Executives are made aware of what contingency plans are in place to address various cyber-based attacks and their aftermath. Executive awareness courses in cybersecurity are mandatory for nearly all sectors.	Cybersecurity risks are considered as an agenda item at every executive meeting, and funding and attention is reallocated to address those risks. Executives are regarded regionally and internationally as a source of good practice in responsible and accountable corporate cybersecurity governance.

Figure 4. Dimension 3: Cybersecurity Education, Training and Skills for Executive Awareness Raising

The Speech Interface as an Attack Surface: An Overview

Mary K. Bispham, Ioannis Agrafiotis, Michael Goldsmith
Department of Computer Science
University of Oxford, United Kingdom
Email: {mary.bispham, ioannis.agrafiotis, michael.goldsmith}@cs.ox.ac.uk

Abstract—**This paper investigates the security of human-computer interaction via a speech interface. The use of speech interfaces for human-computer interaction is becoming more widespread, particularly in the form of voice-controlled digital assistants. We argue that this development represents new security vulnerabilities, which have yet to be comprehensively investigated and addressed. This paper presents a comprehensive review of prior and related work in this area to date. Based on this review, we propose a high level taxonomy of attacks via the speech interface. Our taxonomy systematises prior work on the security of voice-controlled digital assistants, and identifies new categories of potential attacks, which have yet to be investigated and thus represent a focus for future research. The attack surface presented by the speech interface comprises not only the voice-controlled device itself, but the entire process of human-computer interaction including the human user. In accordance with this, our taxonomy categorises attacks via the speech interface according to human perceptions of the attacks, whilst also aligning the categories of the taxonomy to vulnerabilities in various parts of the architecture of voice-controlled systems. This paper is an extended version of a previous paper in which our taxonomy of attacks via the speech interface was first presented.**

Keywords–cyber security; human-computer interaction; voice-controlled digital assistants; speech interface.

I. Introduction

The introduction of a speech interface represents a potential expansion of a system's attack surface. With regard to voice-controlled digital assistants, there are clearly serious security concerns arising from an increasingly pervasive presence of such agents. This paper presents a comprehensive overview of the types of attacks that might be targeted at voice-controlled systems, and categorises these attacks in a high-level taxonomy. This paper is an extended version of a previous paper in which our taxonomy was first presented (Bispham et al. [1]).

Voice-controlled digital assistants are being used to perform an increasing range of tasks, including Web searching and question answering, diary management, sending emails, and posting to social media. Such 'assistants' are intended to act as brokers between users and the vastly complex, often intimidating cyber world. Their functionalities are being expanded from personal to business use [2]. Sarikaya [3] refers to personal digital assistants as a "metalayer of intelligence" between the user and various different services and actions. With the advent of assistants such as Amazon's Alexa that can be used to control smart home devices, control of systems via a speech interface has extended beyond purely virtual environments to include also cyber-physical systems. Pogue [4] describes voice control as a "breakthrough in convenience" for the Internet of Things. Speech interfaces may eventually be used in time-sensitive and even life-critical contexts, such as hospitals, transport and the military [5] [6]. There is some speculation that communication with computers via natural language represents the next major development in computing technology [7].

Notwithstanding its potential benefits, security concerns associated with such a development have yet to be comprehensively addressed. There has been a considerable amount of debate on the threat to privacy from 'listening' devices, highlighted perhaps most dramatically in a recent request for speech data from Amazon's Alexa as a 'witness' in a murder inquiry [8]. By comparison, the security issues associated with voice-controlled assistants have to date received relatively little attention. Such security issues are however significant. A speech interface potentially enables an attacker to gain access to a victim's system without needing to obtain physical or internet access to their device. Thus, the human-like digital personas intended to give users a sense of familiarity and control in interactions with their systems may in reality be exposing users to additional risks. Internet security company AVG pointed out in 2014 the danger of the speech interface being exploited as a new attack surface, demonstrating how smart TVs and voice assistants might respond to synthesised speech commands crafted by an attacker as well as to their users' voices [9]. The reality of this possibility was recently illustrated by a TV advertisement that contained spoken commands for activation of Google Home on listeners' phones for product promotion purposes. The advert was criticised as a potential violation of computer misuse legislation in gaining unauthorised access to listeners' systems [10]. Another example was an instance in which it was shown to be possible to open a house door from the outside by shouting a command to digital assistant Siri (as discussed by Hoy [11]).

This paper provides a review of the research that has been done to date on attacks via the speech interface, and identifies the gaps in this prior work. Based on this review, we propose a new taxonomy of attacks via the speech interface, and make suggestions for further work. The scope of this taxonomy is limited to attacks that gain unauthorised access to a system by sound. It is possible to attack a voice-controlled system other than by sound - in a security analysis of Amazon's Echo, for example, Haack et al. [12] identify three means of attack on such systems. In addition to sound-based attacks, the paper identifies network attacks (e.g., sniffing of speech data in transmission between an individual user's device and a provider's servers) and API-based attacks (which might involve hacking a voice-controlled assistant's API, e.g., to change the default wake-up word). However, such attacks not based on sound are not within scope of the taxonomy presented here.

The remainder of the paper is structured as follows. Section II provides general background on human-computer interaction by speech with reference to the current generation of voice-controlled digital assistants. Section III contains a review of prior work relevant to the security of voice-controlled digital assistants, as well as of some indirectly relevant work in related areas of research. This section also includes some speculation on the potential for attacks via the speech interface that are not possible on current commercial systems, but may become possible in future based on current trends in research on speech dialogue systems. Section IV proposes a new high-level taxonomy of attacks via the speech interface, including attacks that have been demonstrated in prior work as well as attacks that may be possible in the future. Section V concludes the paper and contains some suggestions for future research.

II. Background on Voice-Controlled Systems

Speech interfaces that facilitate the execution of particular actions in response to voice commands are referred to as 'task-based' speech dialogue systems, as distinct from 'chatbots', whose purpose is simply to hold a conversation with the user without executing any actions. Current task-based dialogue systems have some similarity with chatbots in that they are often anthropomorphosised, with systems being given the persona of a friendly digital assistant in order to create a sense of communication with a human-like conversation partner. The first voice-controlled digital assistant to be released commercially was Apple's Siri in 2011. Siri was based on an earlier system named Cognitive Assistant that Learns and Organizes (CALO), which had been developed with US defence funding. Siri was followed by the release of Amazon's Alexa in 2014, Microsoft's Cortana in 2015, and most recently in 2016 by Google Assistant [13].

Input to a speech dialogue system is provided by a microphone that captures speech sounds and converts these from analog to digital form. Bellegarda and Monz [14] describe the task of the speech recognition component as the task of extracting from a set of acoustic features the words that generated them, and the task of the natural language understanding component as the task of extracting from a string of words a semantic representation of the user intent behind them. The paper by Bellegarda and Monz conceptualises the process of a user's communication of intent to a speech dialogue system as information transmission across a noisy channel, whereby the user first formulates their intent in words and then vocalises these words as speech, and the dialogue system subsequently extracts from the user's speech the words that generated the speech and then extracts from the words a semantic representation of the intent that generated them. This process is illustrated in the diagram in Figure 1, copied from Bellegarda and Monz's paper.

The typical architecture of a generic speech dialogue system consists of components for speech recognition, natural language understanding, dialogue management, response generation and speech synthesis (see Lison and Meena [15]). In current systems, the speech recognition and natural language understanding components are the components most likely to be targeted in an attack via the speech interface. As explained further below, in current systems the dialogue management and subsequent components are fully controlled by input from the speech recognition and natural language understanding components, and can therefore not be targeted directly.

Speech recognition is typically performed using Hidden Markov Models (HMMs). HMMs calculate the most likely word sequence for a segment of speech according to Bayes' rule as the product of the likelihood of acoustic features present in the speech segment and the probability of the occurrence of particular words in the sentence context (see for example Juang and Rabiner [16]). HMM-based systems for speech recognition originally used Gaussian Mixture Models (GMMs) for the acoustic modelling and n-grams for the language modelling. In recent years, a shift in modelling methods has been seen with the advent of deep learning. Huang et al. [17] describe recent developments in which Deep Neural Networks (DNNs) have replaced GMMs to extract acoustic model probabilities, and Recurrent Neural Networks (RNNs), a particular type of DNN, have replaced n-grams to extract language model probabilities. Speech recognition technology has become quite advanced. In 2016, Microsoft Research reported that its automatic speech recognition capability had for the first time matched the performance of professional human transcriptionists, achieving a word error rate of 5.9 per cent on the Switchboard dataset of conversational speech produced by the National Institute of Standards and Technology (NIST) in the US (see Xiong et al. [18]).

Natural language understanding in the context of a voice-controlled system is the task of extracting from a user's request a computational representation of its meaning that can be used by the system to trigger an action. The task of mapping a string of words to a representation of their meaning is known as semantic parsing. Liang [19] gives as an example of semantic parsing the instance where a request to cancel a meeting is mapped to a logical form that can be executed by a calendar API. The process of semantic parsing may include syntactic analysis as an intermediate step. Methods of syntactic analysis used in voice-controlled systems include dependency parsing, which is the task of determining syntactic relationships within a sentence, such as verb-object connections (see for example McTear [20]). Current speech dialogue systems typically use semantic representations known as semantic frames (see Sarikaya et al. [21]). Semantic frames provide a structure for representing the meaning of utterances that requires firstly identification of the general domain or concept that a user request relates to (such as travel), secondly determination of the user intent (such as to book a flight), and thirdly slot-filling, which involves identifying specific information relevant to the particular request (such as destination city). Sarikaya [3] states that the tasks of domain identification and intent determination in semantic parsing to frames are often performed using support vector machines, whereas slot-fitting is commonly performed using Conditional Random Fields (CRFs). Some recent research has indicated that traditional machine learning methods are now being out-performed in the semantic parsing task for spoken dialogue systems by neural networks, similar to the replacement of n-gram-based systems for language modelling in speech recognition by RNNs. Mesnil et al. [22], for example, present results showing superior performance by RNNs on the slot-filling task for the Air Travel Information System (ATIS) dataset in comparison to the performance of CRFs on the same task. Despite such efforts, it is clear that, unlike in the case of speech recognition, the state-of-the-art in natural language understanding remains far from parity with human capabilities. This is evident in the occasional failure of voice assistants to correctly interpret the meaning of a word in context, despite the correct word or meaning being obvious to any human listener. Stolk et al. [23] give the examples of

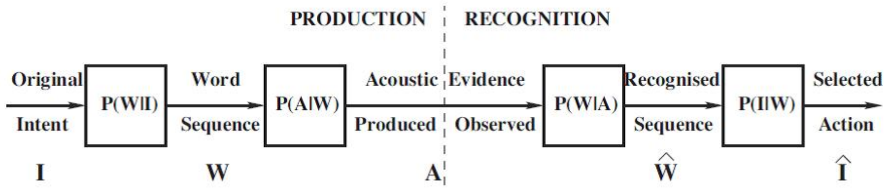


Figure 1. An example of integrated speech and language processing: personal assistance seen as information transmission across a noisy channel [14]

Apple's assistant Siri mistaking the word 'bank' in the sense of 'river bank' for a financial institution, and of Siri giving directions to a casino when asked about a gambling problem.

Dialogue management is the task of determining the most appropriate action that should be taken in response to a user's request. The dialogue management component then instructs the response generation and (in the case that a verbal action is required) speech synthesis components of system to take the necessary action. Sarikaya [3] states that the dialogue management task in personal digital assistants is far more challenging than in older speech recognition systems. Older speech recognition systems were commonly limited to one general purpose, such as providing travel information. Digital assistants, by contrast, are designed to perform a large number of tasks, including providing information on many different topics, connecting with web applications to fulfil a variety of user requests, and controlling devices in the Internet of Things. Sarikaya describes the structure of a dialogue manager in a digital assistant as consisting of a dialogue state tracker, which updates the 'state' of the dialogue based on the representation of user intent generated by the natural language understanding module, and a dialogue policy that controls the execution of tasks in response to the user request.

The dialogue management component in current speech dialogue systems is on the whole still rule-based, i.e., it maps user intent to dialogue states and dialogue states to actions based on hand-crafted rules, as stated by McTear [20]. The dialogue management capabilities in current systems are thus fully dependent on input from the speech recognition and natural language understanding components and do not therefore represent a separate point of attack. Rule-based dialogue management systems have the advantage of limiting the potential for error and unintended functionality in the dialogue management process (see McTear [24]). However, such systems are also likely to be lacking in flexibility and limited in scope. There has been some research on the eventual replacement of current rule-based systems by more sophisticated dialogue management systems based on reinforcement learning, which would enable voice assistants to learn directly from their interactions with users. Young et al. [25] propose ideas for dialogue management based on Partially Observable Markov Decision Processes (POMDPs), which model a dialogue as a Markov process with transition probabilities between states, for which a probability distribution over all possible states is continuously maintained. This approach seeks to represent the uncertainty inherent in the fact that a user's intent is not directly observable, but rather inferred probabilistically from their utterance. Systems based on POMDPs combine Bayesian inference for belief state tracking to determine the most likely interpretation of a user's utterances with reinforcement learning for optimisation of the dialogue policy, whereby a reward function is used to train the system as to the most appropriate action to take in response to a user utterance based on user feedback.

Modern voice-controlled digital assistants implement the generic components of speech dialogue systems in the context of a cloud-based service that enables users to interact by voice with smartphones and laptop/desktop computers, as well as to control smart home devices by voice using bespoke hardware. The speech recognition and natural language understanding functionalities of these systems are performed in the provider's cloud. Chung et al. [26] provide an overview of the typical ecosystem of modern voice-controlled digital assistants in the example of Amazon's Alexa (see Figure 2).

In order to control streaming of audio data to the cloud, current voice-controlled digital assistants include, in addition to the generic speech dialogue system components, an activation component consisting of a wake-up word, which, when spoken by the user, triggers streaming of the subsequent speech audio data to the provider's cloud for processing. Examples of wake-up words include 'Ok Google' for Google Assistant and 'Alexa' for Amazon's Alexa. Wake-up word recognition is the only speech processing capability on users' individual devices, and consists of a short 'buffer' of audio data from the device's environment that is continuously recorded and deleted [27]. Wake-up word activation can be triggered by false positives. Chung et al. [28], for example, refer anecdotally to accidental activation of the Alexa assistant by a sentence containing the phrase 'a Lexus' (see also Michaely et al. [29]), and Vaidya et al. [30] refer to the misrecognition of the phrase "Cocaine Noodles" as "OK Google". False positives in wake-up word recognition may result from misrecognition of a word as the wake-up word, as in the example given by Chung et al., or else from use of a wake-up word in the context of speech not intended to activate a voice assistant, for example the use of the word 'Alexa' as the name of a person in a conversation. Këpuska and Bohouta [31] discuss the latter problem of distinguishing between an 'alerting' and a 'referential' context in wake-up word recognition. It is also possible for voice assistants to be activated by background noise that has frequencies overlapping with those of human speech (see Islam et al. [32]). The vulnerability of wake-up word recognition to false positives was demonstrated in an incident in which an Amazon Alexa device misinterpreted a word spoken in a private conversation as the wake-up word 'Alexa', and subsequently misinterpreted other words in the

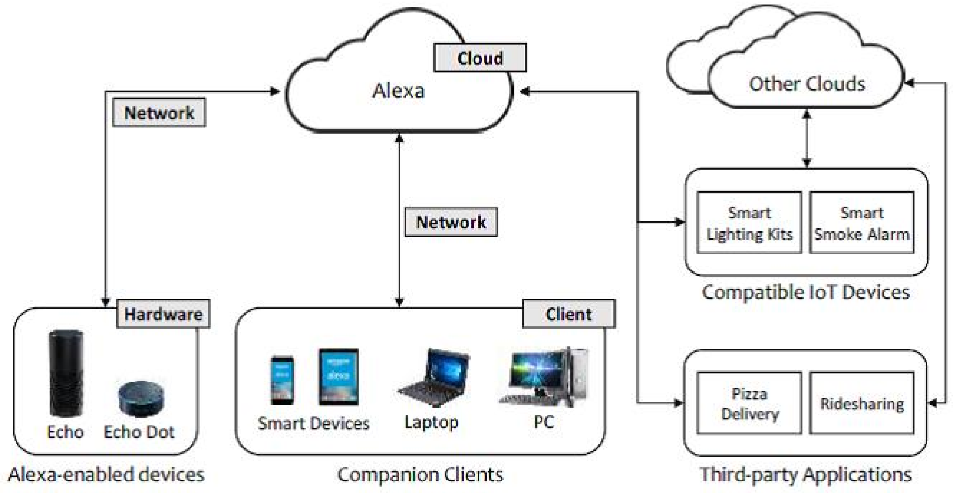


Figure 2. Amazon Alexa Ecosystem [26]

conversation as commands to send a message to a contact, resulting in a recording of a couple's private conversation in their home being sent to a colleague [33].

The current generation of voice-controlled digital assistants have also introduced platforms for the development of third-party voice applications that can be incorporated in the provider's cloud and made available to users via the assistant's speech interface. Examples of such third-party applications are Alexa Skills and Google Conversation Actions. Third-party applications in systems such as Google Assistant can be accessed by users by asking to 'speak' to the voice app (as named by the developer) [34]. Such apps can be used for example to enable users to access information services or to purchase products.

III. Attacks via the Speech Interface in Prior and Related Work

There has been a limited amount of prior work on the security of speech interfaces and voice-controlled digital assistants, as well as some prior work in related areas of research. A review of prior work relevant to attacks on the current generation of voice-controlled digital assistants is presented, and summarised in Table I. Our review further includes some speculation on attacks that are not possible in relation to the current generation of voice-controlled systems, but that may become possible in the future based on current research trends. The review is concerned with sound-based attacks only, whilst recognising that attacks by sound are only a subset of the potential attacks that might be targeted at a voice-controlled digital assistant. The review does not analyse the specific aims of the attacks described in prior work beyond the general goal of gaining unauthorized access to a system via a speech interface. Our review of attacks in prior and related work is organised according the mechanism of attack that they relate to. These mechanisms are plain speech, inaudible sound injection, adversarial learning, and active attack.

A. Plain Speech

Several researchers have investigated the ways in which voice-controlled digital assistants might be exploited simply by using standard voice commands. This possibility arises out of the inherently open nature of natural speech. Such potential vulnerabilities associated with speech-controlled systems have been highlighted for example by Dhanjani [35], who describes a security vulnerability identified in Windows Vista that allowed an attacker to delete files on a victim's computer by playing an audio file hosted on a malicious website or sent to the victim as an email attachment. Dhanjani speculates that the potential for such attacks is magnified with the increasing use of speech recognition technology in the Internet of Things. He postulates a hypothetical attack on Amazon's Echo, a device designed to be used for voice control of home appliances via digital assistant 'Alexa', which would potentially cause psychological or physical harm to the victim by controlling their smart home environment. This hypothetical attack involves a piece of malware consisting of JavaScript code that plays an audio file giving a command to Alexa if there has been no user activity on the mouse or keyboard after a certain period of time (thus aiming to play the file at a time when the user may be away from their computer and therefore will not hear the audio command being played). Diao et al. [36] investigate possibilities for gaining unauthorised access to a smartphone via a malicious Android app that uses the smartphone's own speakers to play an audio file containing voice commands. The attacks proposed by the authors include an attack in which the smartphone is manipulated to dial a phone number that connects to a recording device, and then to disclose information, such as the victim's calendar schedule, by synthesised speech that is recorded by the device. Diao et al. envisage such attacks being executed whilst the victim is asleep and therefore unable to hear the malicious voice command. Such an attack might in fact be executed whilst the victim is neither away from their phone or asleep, but their attention is merely directed elsewhere.

B. Inaudible Sound Injection

Kasmi and Esteves describe a different type of attack in which voice commands are transmitted silently to a victim's phone via electromagnetic interference using the phone's headphones as an antenna [37]. Unlike plain speech attacks, this attack is not detectable even if the victim is consciously present at the time of the attack, although for technical reasons the attack can only be performed if the attacker is in close proximity to the victim's device. The attacks using this mechanism envisaged by Kasmi and Esteves include controlling transmissions from a smartphone by activating or deactivating Wifi, Bluetooth, or airplane mode, and browsing to a malicious website to effect drive-by-download of malware. Young et al. [38] also describe a 'silent' attack on smartphones via the voice command interface that enables an attacker to perform actions such as calling fee-paying phone numbers, posting to Facebook in the victim's name to damage their reputation, accessing email messages, and changing website passwords from the victim's phone. The attack requires a short period of time during which an attacker has unsupervised physical access to the phone in order to attach a Raspberry Pi-based tool that is recognised by the phone as headphones with a microphone. Zhang et al. [39] and Song and Mittal [40] present methods for injecting voice commands to voice-controlled digital assistants at inaudible frequencies by exploiting non-linearities in the processing of sounds by current microphone technology, which can lead voice-controlled systems to detect a command as having been issued within the human audible frequency range, despite the sound not having been perceptible to humans in reality. Silent attacks such as these target the 'voice capture' stage of voice control, i.e., the process of conversion of speech sounds by the microphone from analog to digital form prior to speech recognition.

C. Adversarial Learning

There has also been some prior work towards using adversarial machine learning in attacks on voice-controlled digital assistants. Adversarial learning can be broadly defined as a process of identifying unexpected input that a machine learning-based system classifies in a way that a human would regard as erroneous. This is done by some form of systematic exploration of the system's input space, with the aim of discovering 'adversarial examples' within that space that an attacker can exploit to their advantage. Some adversarial machine learning methods involve manipulating inputs based on knowledge of calculations within the classifier (such 'white-box' methods include approaches such as the Fast Gradient Sign Method and the Jacobian-based Saliency Map Approach for altering input to a DNN, as described for example in Goodfellow et al. [41]). Other methods seek to manipulate input on a 'black-box' basis, i.e., without knowledge of the inner workings of a target system. McDaniel et al. [42] explain that processes of adversarial machine learning rely on identifying 'adversarial regions' in a classification category that have not been covered by training examples. The exact reasons for the effectiveness of particular adversarial examples are difficult to determine, as the decision-making process in a neural network cannot be precisely reverse-engineered (see for example Castelvecchi [43]). In this sense, whilst some adversarial learning methods require more knowledge of the target network than others, all attacks on DNN-based systems are of necessity 'black-box' attacks, although attacks requiring detailed knowledge of the system's functionality are referred to here as white-box in order to distinguish them from attacks not requiring such detailed knowledge.

Adversarial learning to attack DNN-based systems was first demonstrated in image classification (see for example Szegedy et al. [44]), but has recently also been applied to speech recognition. One example is the work presented by Vaidya et al. [30], who used audio mangling to distort commands issued to the precursor to Google Assistant, Google Now (this 'mangling' involved reverse MFCC, where MFCC features extracted from a speech sound were used to generate a mangled version of the sound). The mangled commands included commands to open a malicious website, make a phone-call and send a text, in addition to the Google Now wake-up command 'Ok Google'. The work showed that the distorted commands continued to be recognised by the speech recognition system despite being no longer recognisable by humans, who perceived them instead as mere noise. Thus, the distorted commands represented adversarial examples for the target system. The work by Vaidya et al. was expanded by Carlini et al. [45], who also proved the possibility of prompting Google Now to execute mangled commands that had been shown to be unintelligible to humans in an experiment using Amazon Mechanical Turk. The attacks by Vaidya et al. and Carlini et al. on Google Now were 'black-box' attacks, i.e., they were constructed without knowledge of the inner workings of the speech recognition system. Carlini et al. additionally conducted a successful 'white-box' attack on Carnegie Mellon University's SPHINX speech recognition system (based on GMMs rather than DNNs), in which 'mangled' adversarial commands were crafted with knowledge of the workings of the system.

Other work on adversarial learning targeting speech recognition includes that by Iter et al. [47], who used two adversarial machine learning methods originally applied in image classification to manipulate a speech recognition system based on Google DeepMind's WaveNet technology to mistranscribe a number of utterances. This included prompting the system to transcribe the utterance "Please call Stella" as "Siri call police". The attacks by Iter et al. are white-box attacks, i.e., they rely on some knowledge of the details of the target neural network. The authors mention the possibility of developing a black-box attack methodology in future work. Similar to Iter et al., Cisse et al. [48] were also able to prompt mistranscription of utterances, including mistranscription by Google Voice in a 'black-box attack', using an adversarial machine learning method called Houdini. Alzantot et al. [49] used a black-box attack method based on a genetic algorithm to engineer misclassification of speech command words, such as 'on', 'off', 'stop', etc., by a machine learning-based speech recognition system. Carlini and Wagner [50] have demonstrated a white-box attack on Mozilla's DNN-based DeepSpeech speech-to-text transcription in which it was shown to be possible to prompt mistranscription of a speech recording as any target phrase, regardless of its degree of similarity to the original phrase, by making perturbations to the original recording that did not affect the original phrase as heard by humans. Schöenherr et al. demonstrate a similar type of attack on open-source speech recognition system Kaldi [51]. In contrast to the attacks by Vaidya et al. and Carlini et al., which would be perceived by victims as unexplained noise, attacks based on methods such as those developed by Iter et al., Cisse et al., Carlini and Wagner and Schöenherr et al. would be perceived by victims as ordinary speech and would therefore by more difficult to detect. Schöenherr et al. refer to this type of attack as "psychoacoustic hiding". To date, such work has

TABLE I. SUMMARY OF PRIOR AND RELATED WORK RELEVANT TO ATTACKS VIA THE SPEECH INTERFACE

Paper	Attack Mechanism	Target Component	Human Perception of Attack
Dhanjani [35]	plain speech	speech interface in PC (Windows Vista)	standard voice command
Diao et al. [36]	plain speech	speech interface in voice-controlled digital assistant (Google Voice Search)	standard voice command
Kasmi and Esteves [37]	inaudible sound injection	voice capture in voice-controlled digital assistant (Google Now, Siri)	silence
Young et al. [38]	inaudible sound injection	voice capture in voice-controlled digital assistant (Siri)	silence
Zhang et al. [39]	inaudible sound injection	voice capture in voice-controlled digital assistant (Apple Siri, Amazon Alexa, Microsoft Cortana and others)	silence
Song and Mittal [40]	inaudible sound injection	voice capture in voice-controlled digital assistant (Google Now, Amazon Alexa)	silence
Vaidya et al. [30]	adversarial learning	speech recognition in voice-controlled digital assistant (Google Now)	white noise
Carlini et al. [45]	adversarial learning	speech recognition in voice-controlled digital assistant (Google Now) / speech recognition (CMU Sphinx)	white noise
Yuan et al. [46]	adversarial learning	speech recognition in speech transcription system (Kaldi)	music
Iter et al. [47]	adversarial learning	speech recognition in speech transcription system (WaveNet)	unrelated language
Cisse et al. [48]	adversarial learning	speech recognition in voice-controlled digital assistant (Google Voice)	unrelated language
Alzantot et al. [49]	adversarial learning	speech recognition in speech transcription system (TensorFlow)	unrelated language
Carlini and Wagner [50]	adversarial learning	speech recognition in speech transcription system (DeepSpeech) speech recognition in speech transcription system (DeepSpeech)	music unrelated language
Schöenherr et al. [51]	adversarial learning	speech recognition in speech transcription system (Kaldi)	unrelated language
Papernot et al. [52]	adversarial learning	natural language understanding in sentiment analysis system	nonsensical language
Liang et al. [53]	adversarial learning	natural language understanding in text classification system	unrelated language
Jia and Liang [54]	adversarial learning	natural language understanding in question answering system	unrelated language
Alzantot et al. [55]	adversarial learning	natural language understanding in sentiment analysis and textual entailment systems	unrelated language
Kuleshov et al. [56]	adversarial learning	natural language understanding in spam filtering, fake news detection and sentiment analysis systems	unrelated language
Li et al. [57]	adversarial learning	natural language understanding in sentiment analysis and toxic content detection systems	unrelated language
Bispham et al. [58]	adversarial learning	speech recognition in Google Assistant natural language understanding in Amazon Alexa Skills	nonsensical language unrelated language

been limited to speech-to-text transcription, i.e., it has not yet demonstrated mistranscription of voice commands capable of executing an action.

In addition to prompting mistranscription of speech, Carlini and Wagner demonstrated the possibility of manipulating music recordings so as to prompt them to be transcribed by DeepSpeech as a given string of words, demonstrating for example that a recording of Verdi's Requiem could be manipulated to be transcribed by DeepSpeech as “Ok Google, browse to evil.com”. Yuan et al. [46] similarly demonstrate the possibility of hiding voice commands in music. Unlike the attacks crafted by Carlini and Wagner, the attacks crafted by Yuan et al. are reportedly effective over the air as well as via audio file input, although their attacks are also white-box attacks and are limited to speech-to-text transcription rather than being demonstrated on voice-controlled digital assistants as such. Another type of adversarial learning attack on speech recognition is presented by Bispham et al. [58], who present the results of work demonstrating a black-box attack in which voice commands to a target system are hidden in nonsensical word sounds that are perceived as meaningless by humans. One further, currently hypothetical, type of adversarial learning attack on speech recognition arises from the development of voice-controlled systems that are capable of interacting with users in more than one language (see for example Lopez-Moreno et al. [59]). It could be possible for attackers to identify instances where input in one language is misclassified by a system as a different input in another language. Depending on the language capabilities of the human listener, an adversarial learning attack prompting mistranscription of a utterance in one language as a different utterance in another language would be perceived by the human listener either as unrelated speech, or else as nonsensical or unintelligible speech.

Adversarial learning has also recently been applied to some areas of natural language understanding. This work has been performed mainly outside the context of voice-controlled systems, although there has been some preliminary work on attacks targeting natural language understanding in voice-controlled digital assistants, as discussed below. The generation of adversarial examples in natural language understanding is more complex than the generation of adversarial examples in image or speech recognition. Unlike in the case of continuous data such as image pixels or audio frequency values, adversarial generation of natural language is not a differentiable problem. As word sequences are discrete data, it is not possible to change a word sequence representing an input to a machine learning classifier directly by a numerical value in order to effect a change in output of the classifier. The areas focussed on in prior work include sentiment analysis (see Papernot et al. [52]), text classification (see Liang et al. [53]), and question answering (see Jia and Liang [54]). Papernot et al. [52] use the forward derivative method, a white-box adversarial learning method, to identify word substitutions that can be made in sentences inputted to an RNN-based sentiment analysis system so as to change the ‘sentiment’ allocated to the sentence. In contrast to adversarial examples in image classification and speech recognition, in which alterations made to the original input are imperceptible to humans, the alterations made to sentences in order to mislead the RNN-based sentiment analysis system targeted in the work by Papernot et al. are easily perceptible to humans as nonsensical, albeit that the attack intent remains hidden. For example, substituting the word ‘I’ for the word ‘excellent’ in an otherwise negative review is shown in the paper to lead it to being classified as having positive sentiment. Whereas the altered sentence will appear unnatural to a human, the target system is not capable of identifying the nonsensical nature of the adversarial input.

Papernot et al. state that the lack of naturalness of the adversarial examples in their attacks on natural language understanding will need to be addressed in future work. By contrast to Papernot et al., Liang et al. [53] demonstrate a linguistically plausible attack on a natural language understanding system. The authors adapt the Fast Gradient Sign Method from adversarial learning in image classification to make human-undetectable alterations to a text passage (by adding, modifying and/or removing words) so as to change the category that is allocated to the passage by a DNN-based text classification system. The attack by Liang et al. is white-box, requiring details of the calculations inside the network. Jia and Liang [54] also demonstrate a linguistically plausible attack in the context of question answering. Their work involves misleading a number of question answering systems by adding apparently inconsequential sentences to text passages from which the systems extract answers to questions. The method works by first choosing a target wrong answer to a given question, and then crafting a sentence containing information leading to this wrong answer that can be inserted into the original passage without noticeably changing its overall import. The attack method proposed by Jia and Liang is a black-box method, not requiring knowledge of the internal details of the target network.

Kuleshov et al. [56] use a word replacement approach in an adversarial learning attack targeting spam filtering, fake news detection and sentiment analysis. Their attack selects acceptable replacement words according to a semantic similarity measure based on 'thought vectors' in the form of averages of individual word vectors, and a syntactic similarity measure based on a language model, with the stated aim of 'formalising' the process of generating adversarial examples in natural language classification. The attacks demonstrated by Kuleshov et al. are white-box attacks, in that they rely on knowledge of objective function calculations in order to optimise the attack. Li et al. [57] demonstrate an attack on sentiment analysis and toxic content detection systems under both white-box and black-box conditions, using different types of perturbation of text including deliberate misspellings as well as word replacement. They note that character-level perturbations have a higher success rate in generating adversarial examples than word-level perturbations. Whilst all of the attacks on natural language understanding described above are demonstrated outside the context of voice-controlled systems, Bispham et al. [58] present a proof-of-concept study for attacks targeting natural language understanding in a voice-controlled digital assistant, using third-party Skills for Amazon Alexa as a specific example of a target system. The attack concept developed in the proof-of-concept study involves word replacement in a target command, as well as the transplant of content words from a target command to another meaning context where they are used in a different sense. These processes are shown to generate adversarial utterances that trigger target actions in a dummy Alexa Skill, whilst appearing to humans to have an unrelated meaning. The examples of attacks on natural language understanding described here are indicative of the fact that natural language understanding technology currently represents only a crude approximation of human language understanding that is easily destabilised.

In the specific context of voice-controlled digital assistants, the need to circumvent the wake-up word activation presents a potential issue of linguistic plausibility for adversarial learning attacks on natural language understanding, in that unlike in the case of adversarial learning attacks targeting speech recognition, it is difficult to incorporate a device's wake-up word as part of an attack based on confusion of meaning. However, given the known presence of false positives with respect to wake-up word recognition, this type of attack should not be dismissed as impossible.

D. Active Attack

All of the attacks described in the prior and related work summarised above are 'passive' attacks, in the sense that they seek to exploit vulnerabilities that are already present in a target system. There is also the possibility of 'active' attacks that seek to undermine the functionality of the system itself. Miller et al. [60] refer to these attack types as 'foiling' and 'tampering', respectively. An example of active attack on a natural language interface was seen in an attempt by Microsoft to launch a social media chatbot named Tay. Tay was intended to learn human-like language use from interactions with humans on social media platform Twitter. Within a short time of launching the chatbot had to be closed down on account of having been flooded by some users with offensive language and views, which it then proceeded to imitate (see Følstad and Brandtstæd [61]). In the context of cloud-based voice assistants, active attacks might involve manipulating the response behaviour of the system for malicious ends. Rather than passively exploiting weaknesses in the speech recognition and natural language understanding functionalities of a voice-controlled system, such attacks would seek actively to undermine the system's ability to respond appropriately to spoken input by manipulating the dialogue management functionality. The potential for active attacks on voice-controlled digital assistants arises from the aim of providers of such systems to enable cloud-based assistants to continually 'improve' in interactions with their users. The capacity of voice-controlled digital assistants to learn from feedback from user conversations can be expected to increase with the introduction of commercially available voice assistants based on reinforcement learning. This capacity for learning might be abused by attackers aiming to confuse the system using various means, such as inconsistent verbal inputs over time, incongruous feedback in dialogue turns, or inappropriate corrections of a target system's responses. Attackers might for example launch a denial of service-type attack by mass disconfirmation of legitimate commands. Such attacks remain hypothetical at time of writing, as the current generation of voice-controlled digital assistants still use rule-based rather than reinforcement learning-based dialogue management technology, as explained above. However, this type of attack may become significant in future.

A different type of active attack affecting human interaction with voice-controlled systems in future might arise from the voice-controlled systems themselves, via the evolution of machine-generated languages that diverge from human language use. Whilst mismatches between human and machine understanding of natural language have generally been viewed as failure on the part of machines to attain human levels of language understanding, it is also possible to view such mismatches as a failure on the part of humans to grasp the way in which meaning is represented by a machine. This was illustrated by an instance in which two bots were observed to develop a language for communication between themselves that was unintelligible to humans. This occurred as an unintended consequence of research by Lewis et al. [62], the aim of which was to train two bots to negotiate with one another in natural language using reinforcement learning. In the course of the learning process, the bots began to deviate from natural English in their language use, instead using apparently nonsensical strings of words in their communication with each other. This deviation was presumed to have effected more efficient communication between the two bots in achieving an optimal outcome in their negotiations. The development of bots capable of autonomously evading human language understanding may represent an increasingly significant future security threat, given the potential for loss of control over the behaviour of such systems by their human users. A malicious actor might be able to trigger a machine-machine reinforcement learning process in a target system with the specific aim of prompting it to behave in a way that was unintended by its human developers.

IV. Taxonomy of Attacks via the Speech Interface

Reflecting on the review of prior work and related work in Section III, we propose a high-level taxonomy of categories of attacks via the speech interface. This taxonomy is presented in Figure 3. The principle behind the taxonomy is to identify the various categories of non-speech and speech sounds that humans are capable of perceiving, and to group attacks via the speech interface according to these categories, rather than according to the attack mechanism used by an attack or by the specific technical vulnerability that it exploits. The last column of Table I shows the perceptual category that might be allocated to the attacks described above by humans. By applying this categorisation principle, our taxonomy is capable of encompassing attack types that have been shown to be possible in relation to the current generation of voice-controlled systems, as well as attacks that may become possible in future as the state-of-the-art in voice control advances. Thus our taxonomy fulfils the dual purpose of systematising prior work whilst also identifying new directions for future research. Attacks via the speech interface as categorised under our taxonomy might be targeted at any voice-controlled system, including any voice-controlled digital assistant and any third-party applications accessible through it, and might be delivered via any speaker-enabled device capable of producing sound in the target system's environment.

In the taxonomy, attacks via the speech interface are primarily grouped into two categories: 'overt' attacks, which seek to gain unauthorised access to systems using the same voice commands as might be given by a legitimate user and are thus easily detectable by a human, and 'covert' attacks, which seek to gain access using speech commands that have been distorted in some way so as to escape detection by the victim. Another way of characterizing this division is as a distinction between attacks that make illicit use of the intended functionalities of a speech dialogue system, and attacks that exploit unintended functionalities. Overt attacks use plain speech to exploit an inherent vulnerability in voice-controlled systems that arises from the difficulty of controlling access to a system via the 'speech space'. Covert attacks exploit gaps in the processes of capturing human speech or of translating the captured speech input into computer executable actions in a voice-controlled system. Covert attacks include attacks using inaudible sound injection, adversarial learning, and active attack, as discussed above.

Within the two primary categories of overt and covert attacks, attacks are grouped hierarchically into six final sub-categories based on human perceptual categories, as shown in Figure 3 and explained further below. Malicious inputs in overt attacks consist by definition of ordinary speech. Thus a single sub-category of 'plain-speech' attacks was identified for overt attacks. The attacks demonstrated in prior work using standard voice commands, such as those demonstrated by Dhanjani et al. [35] discussed above, fall into this sub-category. Malicious inputs in covert attacks may include input that consists in human terms of silence, as for example in the attacks demonstrated by Zhang et al. [39], noise, as for example in the attacks demonstrated by Carlini et al. [45], music, as for example in the attacks demonstrated by Yuan et al. [46], nonsense, as for example in the attacks on Google Assistant hiding malicious commands in nonsensical word sounds demonstrated by Bispham et al. [58], and unrelated speech, as for example in the attacks demonstrated by Carlini and Wagner [50]. Based on these examples of attacks in prior work, and in accordance with the categorisation principle chosen for the taxonomy of grouping attacks according to the nature of attacks as they might be perceived by a human listener, five sub-categories of covert attacks via a speech interface were identified, namely attacks consisting of silence, music, noise, 'nonsense', and 'missense'. Nonsense as a malicious input in covert attacks is defined as input that is made up of words or sounds that are in legitimate use in the relevant language, but that combines them in such a way that they do not convey any meaning in terms of human understanding. Missense is defined as unrelated speech that is misheard or misinterpreted by the target system as a target command.

Our taxonomy accords with established criteria for attack taxonomies, as described for example in Hansman and Hunt [63]. These criteria include the requirement that a taxonomy should be 'complete', i.e., cover all possible attacks within its scope, and unambiguous, i.e., it should be possible clearly to allocate every attack to one category within the scope of the taxonomy. The principle of categorising attacks according to human perception ensures that the taxonomy is complete, as all attacks via a speech interface can be allocated to one of the six sub-categories. The taxonomy is also unambiguous, in that it is not possible to allocate the same voice attack to more than one of the six final sub-categories.

At the bottom of Figure 3, the attack categories based on human perceptual distinctions as identified in the taxonomy are aligned to the technical vulnerabilities in the architecture of the current generation of voice-controlled systems that might be targeted by each type of attack. The taxonomy of attacks categorised according to human perception as aligned to technical vulnerabilities at various points of the handling of speech input by voice-controlled systems represents the entire attack surface presented by a speech interface. To the extent that speech processing by voice-controlled systems mimics human speech processing, the attack categories in the taxonomy based on human perception correspond to vulnerabilities in the parts

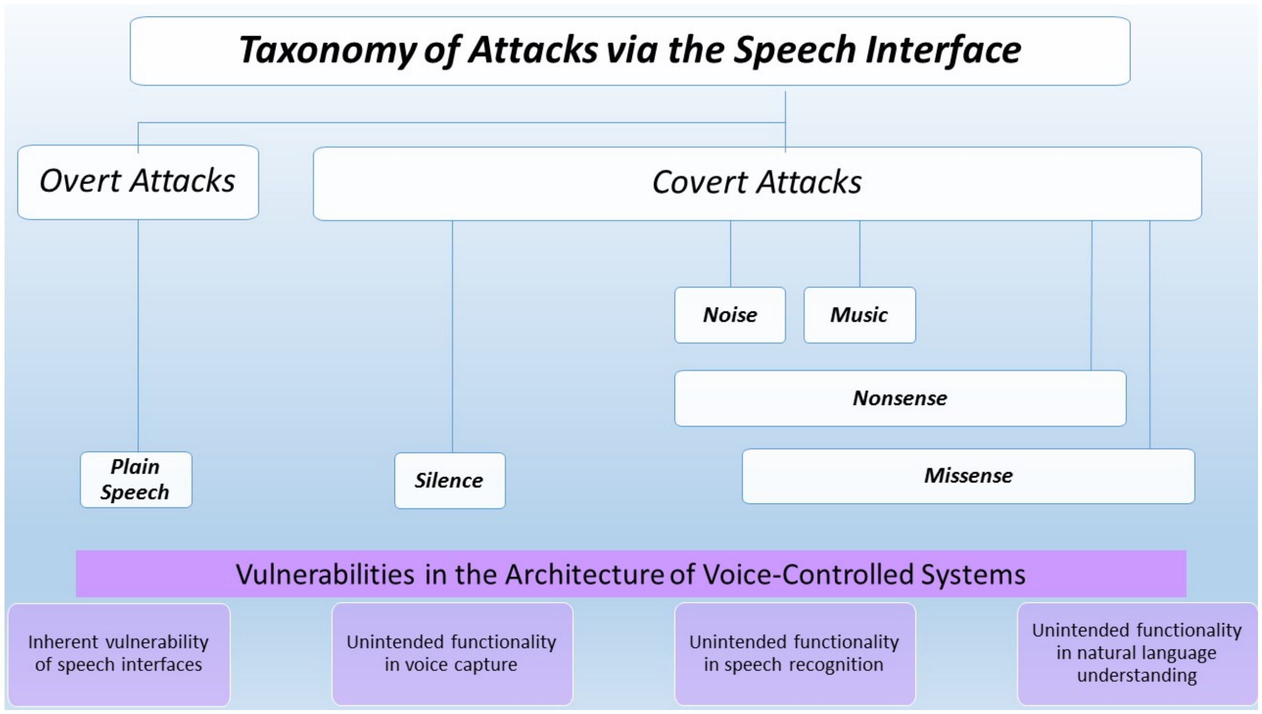

Taxonomy of Attacks via the Speech Interface
Overt Attacks
Covert Attacks
Noise
Music
Nonsense
Plain Speech
Silence
Missense
Vulnerabilities in the Architecture of Voice-Controlled Systems
Inherent vulnerability of speech interfaces
Unintended functionality in voice capture
Unintended functionality in speech recognition
Unintended functionality in natural language understanding

Figure 3. Taxonomy of Attacks via the Speech Interface aligned to Vulnerabilities in the Architecture of Voice-Controlled Systems

of the architecture of voice-controlled systems that represent equivalent human processes, although this correspondence is not exact. The alignment presented in Figure 3 covers the technical vulnerabilities that are present in the current generation of voice-controlled digital assistants, namely the vulnerability arising from the inherent difficulty of controlling access to a system by sound, vulnerabilities in the voice capture process, vulnerabilities in speech recognition, and vulnerabilities in natural language understanding. Whilst the categories of attack based on human perception can be expected to remain stable over time, their alignment to vulnerabilities in the architecture of voice-controlled systems might be expected to shift in future to include new vulnerabilities as the state-of-the-art in voice-controlled systems progresses. Thus, for example, missense attacks might be aligned in future not only to vulnerabilities in the speech recognition and natural language understanding components of voice-controlled systems, but also to vulnerabilities in the dialogue management component, such as the vulnerability presented by the potential for mistraining in the context of dialogue management functionality based on reinforcement learning, as well as the vulnerability presented by the potential for the evolution in reinforcement learning-based systems of bot-generated language that is incomprehensible to humans, as discussed above.

As reflected in the alignment in Figure 3, attacks in plain-speech exploit the inherent vulnerability of speech interfaces on account of the difficulty of controlling access to a system by sound. Attacks in silence attacks exploit vulnerabilities in the voice capture process, as is shown by the alignment of silent attacks to the voice capture component of the architecture in Figure 3. Attacks that use music and noise as malicious input exploit unintended functionality in speech recognition, as is shown by the alignment of these attack categories to the speech recognition component of the architecture. As further reflected in the alignment in Figure 3, nonsense attacks on current voice-controlled systems might be targeted either at the speech recognition or the natural language understanding components of a target system. The attacks in which malicious voice commands were hidden in nonsensical word sounds demonstrated by Bispham et al. [58] can be categorised as nonsense attacks targeting the speech recognition level of handling of speech input in a voice-controlled system. As regards attacks targeting the natural language understanding level, nonsense attacks have yet to be demonstrated with respect to voice-controlled systems directly, although there has been some related work that could be described as nonsense attacks on natural language understanding, such as in the attacks on a sentiment analysis system by Papernot et al. [52] by making nonsensical alterations to text discussed in Section III. Similar attacks might be demonstrated in the context of voice-controlled digital assistants in future.

Similar to nonsense attacks, missense attacks might also be targeted at either the speech recognition or natural language understanding component of current voice-controlled systems, as is also shown in the alignment in Figure 3. Missense attacks targeting speech recognition rely on mistranscription of adversarial input by a target system as a target command. In a missense attack that targets natural language understanding functionality, on the other hand, words might be transcribed correctly by the target system, but their meaning would be misinterpreted. This type of missense attacks would seek to exploit the shortcomings of current natural language understanding functionality in voice-controlled digital assistants in terms of being able to identify the correct meaning of words in context. Prior work on missense attacks in voice-controlled systems has to date been focussed primarily on attacks on speech recognition as incorporated in such systems, as for example in the work of Carlini and Wagner [50]. The attacks described in the proof-of-concept study presented by Bispham et al. [58] for attacks that trigger a target command in an Amazon Alexa Skill using unrelated utterances would fall into the category of missense attacks targeting natural language understanding. There has also been more extensive work demonstrating missense attacks that target natural language understanding functionality in related research areas, such the attacks on question answering by Jia and Liang [54] by making apparently inconsequential alterations to text, or in the work by Kuleshov et al. [56] using word replacement to mislead spam filtering, toxic content detection and sentiment analysis systems, as described in Section III.

As discussed above, attacks on future systems may also include attacks targeting speech recognition in multilingual systems, prompting a target system to mistranscribe input in one language as different input in another. Such attacks would be classed as either nonsense or missense attacks, based on whether or not the cover language used by an attacker was understood by a human listener. As also discussed above, future attacks might further include attacks in which a target system's ability to respond appropriately to spoken input is actively undermined by mistraining of a dialogue management component based on reinforcement learning, as well as attacks that are based on facilitating the evolution of human-incomprehensible languages in autonomous bot-to-bot interactions in reinforcement learning-based systems. The former type of attack would represent a missense attack, with the adversarial input being perceived by human listeners as unrelated language, whereas the latter type of attack would represent a nonsense attack, with the adversarial input being perceived by human listeners as nonsensical language.

V. Conclusion and Future Work

This paper proposes a taxonomy of attacks via the speech interface that covers attacks investigated in prior and related work, as well as attacks that may be possible in the future. The review of prior and related work in this paper indicates that the potential for attacks via a speech interface has yet to be comprehensively assessed. The scope of attacks via a speech interface can be expected to expand with the increasing sophistication of voice-controlled systems. Consequently, there is a need for further security-focussed research in the area of voice-controlled technology.

Future work should seek more extensively to demonstrate the potential for attacks in the various categories of the proposed taxonomy in the context of different technologies and use-case scenarios. Among the taxonomy categories, nonsense and missense attacks targeting the natural language understanding functionality of voice-controlled systems represent types of attacks that have yet to be explored fully in practice. Thus, such attacks should be a special focus of future work. Looking further into the future, attacks based on language confusion in multilingual systems, as well as attacks based on mistraining of dialogue management or facilitation of bot-generated languages in reinforcement learning-based systems, may become a reality requiring the attention of security researchers.

The results of future work should ultimately be used as a basis for the development of more effective defence measures to improve the security of voice-controlled digital assistants and other voice-controlled systems. As a first step in this direction, Bispham et al. [64] present some attack and defence modelling work in which the attack categories in the taxonomy presented here are mapped to currently available defences against attacks via the speech interface, enabling an assessment of the effectiveness of current defences against the various types of attack.

Acknowledgment

This work was funded by a doctoral training grant from the Engineering and Physical Sciences Research Council (EPSRC).

References

[1] M. K. Bispham, I. Agrafiotis, and M. Goldsmith, "A taxonomy of attacks via the speech interface," Proceedings of Third International Conference on Cyber-Technologies and Cyber-Systems, 2018.

[2] "Why Amazon's Alexa may soon become your new colleague," 2017, URL: https://www.inc.com/emily-canal/amazon-alexa-for-business.html [accessed: 2019-05-05].

[3] R. Sarikaya, "The technology behind personal digital assistants: An overview of the system architecture and key components," IEEE Signal Processing Magazine, vol. 34, no. 1, 2017, pp. 67–81.

[4] D. Pogue, "At your command," Scientific American, vol. 315, no. 1, 2016, pp. 25–25.

[5] C. Franzese and M. Coyne, "The promise of voice: Connecting drug delivery through voice-activated technology," vol. 2017, 12 2017, pp. 34–37.

[6] "British navy warships 'to use Siri' as technology transforms warfare," 2017, URL: https://www.theguardian.com/uk-news/2017/sep/12/british-navy-warships-to-use-voice-controlled-system-like-siri [accessed: 2019-05-05].

[7] "The Voice-AI Revolution is a Conversational Interface of Everything," 2017, URL: https://medium.com [accessed: 2019-05-05].

[8] "A Murder Case Tests Alexa's Devotion to Your Privacy," 2017, URL: https://www.wired.com/2017/02/murder-case-tests-alexas-devotion-privacy [accessed: 2018-07-20].

[9] "Voice Hackers Will Soon Be Talking Their Way Into Your Technology," 2014, URL: https://www.forbes.com/sites/jasperhamill/2014/09/29/voice-hackers-will-soon-be-talking-their-way-into-your-technology/ [accessed: 2019-05-05].

[10] "Burger King triggers Google Home devices with TV ad," 2017, URL: https://nakedsecurity.sophos.com/2017/04/18/burger-king-triggers-ok-google-devices-with-tv-ad/ [accessed: 2019-05-05].

[11] M. B. Hoy, "Alexa, Siri, Cortana, and more: An introduction to voice assistants," Medical Reference Services Quarterly, vol. 37, no. 1, 2018, pp. 81–88.

[12] W. Haack, M. Severance, M. Wallace, and J. Wohlwend, "Security analysis of the Amazon Echo," MIT, 2017.

[13] "Google uses Assistant to square up to Siri in AI arms race," 2017, URL: https://www.ft.com/content/f9423056-7efe-11e6-8e50-8ec15fb462f4 [accessed: 2019-05-05].

[14] J. R. Bellegarda and C. Monz, "State of the art in statistical methods for language and speech processing," Computer Speech & Language, vol. 35, 2016, pp. 163–184.

[15] P. Lison and R. Meena, "Spoken dialogue systems: the new frontier in human-computer interaction," XRDS: Crossroads, The ACM Magazine for Students, vol. 21, no. 1, 2014, pp. 46–51.

[16] B.-H. Juang and L. R. Rabiner, "Automatic speech recognition–a brief history of the technology development," Georgia Institute of Technology. Atlanta Rutgers University and the University of California. Santa Barbara, vol. 1, 2005, p. 67.

[17] X. Huang, J. Baker, and R. Reddy, "A historical perspective of speech recognition," Communications of the ACM, vol. 57, no. 1, 2014, pp. 94–103.

[18] W. Xiong et al., "Achieving human parity in conversational speech recognition," arXiv preprint arXiv:1610.05256, 2016.

[19] P. Liang, "Learning executable semantic parsers for natural language understanding," Communications of the ACM, vol. 59, no. 9, 2016, pp. 68–76.

[20] M. McTear, Z. Callejas, and D. Griol, The conversational interface. Springer, 2016.

[21] R. Sarikaya et al., "An overview of end-to-end language understanding and dialog management for personal digital assistants," in IEEE Workshop on Spoken Language Technology, 2016, pp. 391–397.

[22] G. Mesnil et al., "Using recurrent neural networks for slot filling in spoken language understanding," IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP), vol. 23, no. 3, 2015, pp. 530–539.

[23] A. Stolk, L. Verhagen, and I. Toni, "Conceptual alignment: how brains achieve mutual understanding," Trends in Cognitive Sciences, vol. 20, no. 3, 2016, pp. 180–191.

[24] M. McTear, "Conversational modelling for chatbots: Current approaches and future directions," Technical report, Ulster University, Ireland, Tech. Rep., 2018.

[25] S. Young, M. Gašić, B. Thomson, and J. D. Williams, "Pomdp-based statistical spoken dialog systems: A review," Proceedings of the IEEE, vol. 101, no. 5, 2013, pp. 1160–1179.

[26] H. Chung, J. Park, and S. Lee, "Digital forensic approaches for amazon alexa ecosystem," Digital Investigation, vol. 22, 2017, pp. S15–S25.

[27] "Alexa and Google Home Record What You Say, But What Happens To That Data?" 2016, URL: https://www.wired.com/2016/12/alexa-and-google-record-your-voice/ [accessed: 2019-05-05].

[28] H. Chung, M. Iorga, J. Voas, and S. Lee, "Alexa, can I trust you?" Computer, vol. 50, no. 9, 2017, pp. 100–104.

[29] A. H. Michaely, X. Zhang, G. Simko, C. Parada, and P. Aleksic, "Keyword spotting for Google Assistant using contextual speech recognition," in Proceedings of ASRU, 2017, pp. 272–278.

[30] T. Vaidya, Y. Zhang, M. Sherr, and C. Shields, "Cocaine noodles: exploiting the gap between human and machine speech recognition," Presented at WOOT, vol. 15, 2015, pp. 10–11.

[31] V. Këpuska and G. Bohouta, "Improving wake-up-word and general speech recognition systems," in Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence & Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), 2017 IEEE 15th Intl. IEEE, 2017, pp. 318–321.

[32] M. T. Islam, B. Islam, and S. Nirjon, "Soundsifter: Mitigating overhearing of continuous listening devices," in Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services. ACM, 2017, pp. 29–41.

[33] "Amazon Alexa heard and sent private chat," 2018, URL: https://www.bbc.co.uk/news/technology-44248122 [accessed: 2019-05-05].

[34] "How to use third-party Actions on Google Home," 2017, URL: https://www.cnet.com/uk/how-to/how-to-use-third-party-actions-on-google-home/ [accessed: 2019-05-05].

[35] N. Dhanjani, Abusing the Internet of Things: Blackouts, Freakouts, and Stakeouts. " O'Reilly Media, Inc.", 2015.

[36] W. Diao, X. Liu, Z. Zhou, and K. Zhang, "Your voice assistant is mine: How to abuse speakers to steal information and control your phone," in Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices. ACM, 2014, pp. 63–74.

[37] C. Kasmi and J. L. Esteves, "IEMI threats for information security: Remote command injection on modern smartphones," IEEE Transactions on Electromagnetic Compatibility, vol. 57, no. 6, 2015, pp. 1752–1755.

[38] P. J. Young, J. H. Jin, S. Woo, and D. H. Lee, "Badvoice: Soundless voice-control replay attack on modern smartphones," in Ubiquitous and Future Networks (ICUFN), 2016 Eighth International Conference on. IEEE, 2016, pp. 882–887.

[39] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu, "Dolphinattack: Inaudible voice commands," arXiv preprint arXiv:1708.09537, 2017.

[40] L. Song and P. Mittal, "Inaudible voice commands," arXiv preprint arXiv:1708.07238, 2017.

[41] I. Goodfellow, N. Papernot, and P. McDaniel, "cleverhans v0. 1: an adversarial machine learning library," arXiv preprint arXiv:1610.00768, 2016.

[42] P. McDaniel, N. Papernot, and Z. B. Celik, "Machine learning in adversarial settings," IEEE Security & Privacy, vol. 14, no. 3, 2016, pp. 68–72.

[43] D. Castelvecchi, "Can we open the black box of AI?" Nature News, vol. 538, no. 7623, 2016, p. 20.

[44] C. Szegedy et al., "Intriguing properties of neural networks," arXiv preprint arXiv:1312.6199, 2013.

[45] N. Carlini et al., "Hidden voice commands," in 25th USENIX Security Symposium (USENIX Security 16), Austin, TX, 2016.

[46] X. Yuan et al., "Commandersong: A systematic approach for practical adversarial voice recognition," arXiv preprint arXiv:1801.08535, 2018.

[47] D. Iter, J. Huang, and M. Jermann, "Generating adversarial examples for speech recognition," Stanford, 2017.

[48] M. Cisse, Y. Adi, N. Neverova, and J. Keshet, "Houdini: Fooling deep structured prediction models," arXiv preprint arXiv:1707.05373, 2017.

[49] M. Alzantot, B. Balaji, and M. Srivastava, "Did you hear that? adversarial examples against automatic speech recognition," arXiv preprint arXiv:1801.00554, 2018.

[50] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," arXiv preprint arXiv:1801.01944, 2018.

[51] L. Schönherr, K. Kohls, S. Zeiler, T. Holz, and D. Kolossa, "Adversarial attacks against automatic speech recognition systems via psychoacoustic hiding," arXiv preprint arXiv:1808.05665, 2018.

[52] N. Papernot, P. McDaniel, A. Swami, and R. Harang, "Crafting adversarial input sequences for recurrent neural networks," in Military Communications Conference, MILCOM 2016-2016 IEEE. IEEE, 2016, pp. 49–54.

[53] B. Liang, H. Li, M. Su, P. Bian, X. Li, and W. Shi, "Deep text classification can be fooled," arXiv preprint arXiv:1704.08006, 2017.

[54] R. Jia and P. Liang, "Adversarial examples for evaluating reading comprehension systems," arXiv preprint arXiv:1707.07328, 2017.

[55] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang, "Generating natural language adversarial examples," arXiv preprint arXiv:1804.07998, 2018.

[56] V. Kuleshov, S. Thakoor, T. Lau, and S. Ermon, "Adversarial examples for natural language classification problems," 2018.

[57] J. Li, S. Ji, T. Du, B. Li, and T. Wang, "Textbugger: Generating adversarial text against real-world applications," arXiv preprint arXiv:1812.05271, 2018.

[58] M. K. Bispham, I. Agrafiotis, and M. Goldsmith, "Nonsense attacks on Google Assistant and missense attacks on Amazon Alexa," Proceedings of International Conference on Information Systems Security and Privacy, 2019.

[59] I. Lopez-Moreno, J. Gonzalez-Dominguez, O. Plchot, D. Martinez, J. Gonzalez-Rodriguez, and P. Moreno, "Automatic language identification using deep neural networks," in 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2014, pp. 5337–5341.

[60] D. J. Miller, X. Hu, Z. Qiu, and G. Kesidis, "Adversarial learning: a critical review and active learning study," arXiv preprint arXiv:1705.09823, 2017.

[61] A. Følstad and P. B. Brandtzæg, "Chatbots and the new world of HCI," interactions, vol. 24, no. 4, 2017, pp. 38–42.

[62] M. Lewis, D. Yarats, Y. N. Dauphin, D. Parikh, and D. Batra, "Deal or no deal? End-to-end learning for negotiation dialogues," arXiv preprint arXiv:1706.05125, 2017.

[63] S. Hansman and R. Hunt, "A taxonomy of network and computer attacks," Computers & Security, vol. 24, no. 1, 2005, pp. 31–43.

[64] M. K. Bispham, I. Agrafiotis, and M. Goldsmith, "Attack and defence modelling for attacks via the speech interface," Proceedings of International Conference on Information Systems Security and Privacy, 2019.

Threat Analysis using Vulnerability Databases – Topic Model Analysis using LDA and System Model Description –

Katsuyuki Umezawa
Department of Information Science
Shonan Institute of Technology
Fujisawa, Kanagawa 251–8511, Japan
e-mail: umezawa@info.shonan-it.ac.jp

Yusuke Mishina
Cyber Physical Security Research Center (CPSEC)
National Institute of Advanced Industrial Science and Technology (AIST)
Koto-ku, Tokyo 135–0064, Japan
e-mail: yusuke.mishina@aist.go.jp

Sven Wohlgemuth
Research & Development Group
Hitachi, Ltd.
Yokohama, Kanagawa 244–0817, Japan
e-mail: sven.wohlgemuth.kd@hitachi.com

Kazuo Takaragi
Cyber Physical Security Research Center (CPSEC)
National Institute of Advanced Industrial Science and Technology (AIST)
Koto-ku, Tokyo 135–0064, Japan
e-mail: kazuo.takaragi@aist.go.jp

***Abstract*—We proposed a threat analysis method utilizing topic model analysis and vulnerability databases. The method is based on attack tree analysis. We create an attack tree on a evaluation target system and some attack trees on a known vulnerability, and combine the two types of attack trees to create more concrete attack trees. This enables us to calculate the probability of occurrence of a safety accident and to utilize attack trees in future analysis. In this paper, we formulate a topic model analysis and confirm the feasibility of matching known attack cases to vulnerability databases using a topic model analysis tool. In addition, we show that our proposed method can use the results of past threat analysis for the next one. Moreover, we create a system model description based on the attack tree of Tesla's case created using our proposed method. It shows that fake commands can be transmitted from the external information system to the in-vehicle control system. Our approach to automatic threat analysis supports risk analysis in discovering previous unknown relationships and so threats including their potential escalation within an connected IT system.**

Keywords–Threat Analysis; Vulnerability Information; Attack Tree; Topic Model Analysis; System Model Description.

I. Introduction

We proposed a threat analysis method utilizing topic model analysis and vulnerability databases [1]. The background of this proposal is as follows.

Interference and interruption to safety due to security incidents are recognized as a big problem in safety critical systems, such as those for electric power, information communication, automobile, aviation, railway, and medical care. For security of in-vehicle communication in the EVITA project [2], authors have conducted a risk management process. Specifically, risk analysis, security requirement setting, architecture design, prototyping, and demonstration was held. The EVITA project uses attack trees for risk analysis. One way to analyze the causal relationship between safety (hazard) and security (threat) is to express that relationship with a combination of a Fault Tree (FT) and Attack Tree (AT) [3]. The US-based MITRE Corporation provides several tools for vulnerability reporting and aggregation in a vulnerability database (DB). In Common Vulnerabilities and Exposures (CVE) [4], individual software vulnerabilities are stored in a DB. In Common Weakness Enumeration (CWE) [5], common vulnerabilities are cataloged with a focus on the cause of the vulnerability. Furthermore, Common Attack Pattern Enumeration and Classification (CAPEC) [6] is a DB classified by attack pattern. Scientific literature related to safety analysis using FTs is, nowadays, mature. However, the complexity of the problem has significantly increased in security analysis. Elaborate attacks occur with multiple combinations of those vulnerabilities. It is not easy to create an AT that comprehensively captures such possibilities.

We have focused on such problems and proposed a threat analysis method using a vulnerability DB as a practical approach [7][8]. First, we assumed that many attacks were imitations or minor changes of known attacks. Therefore, we believed that expressing attack cases that occurred in the past by using an AT could enable a designer (defender) to become aware of related attacks (recognize the danger). By gradually and continuously applying this approach, it can be useful for reducing vulnerability.

We proposed an algorithm that includes a process for matching each node of an AT described in natural language [7][8]. However, the matching method utilized was not specified. We evaluated the feasibility of this unspecified matching process using a topic model analysis method. In this paper, in addition to our study, we add the formulation of our proposed algorithm, application to examples of a Tesla case [9], the formulation of a topic model, and the possibility of recognizing dangers by using the system model description.

In Section II, we summarize the threat analysis method we proposed in our previous studies [7][8]. In Section III, we formulate the algorithm shown in Section II. We apply the proposed algorithm to the Tesla case examples in Section IV. In Section V, we introduce the topic model analysis. In Section VI, we verify the feasibility of matching attack cases to vulnerability DBs and show the result. We describe a system model description using the attack tree created from the proposed algorithm in Section VII. Section VIII concludes this paper by summarizing the key points and providing an outlook on future activities.

II. Threat analysis using vulnerability databases

This section presents a summary of our proposed method [8]. An overview of the threat analysis method using the vulnerability DB is shown in Figure 1. The proposed threat analysis method conducts the following three procedures:

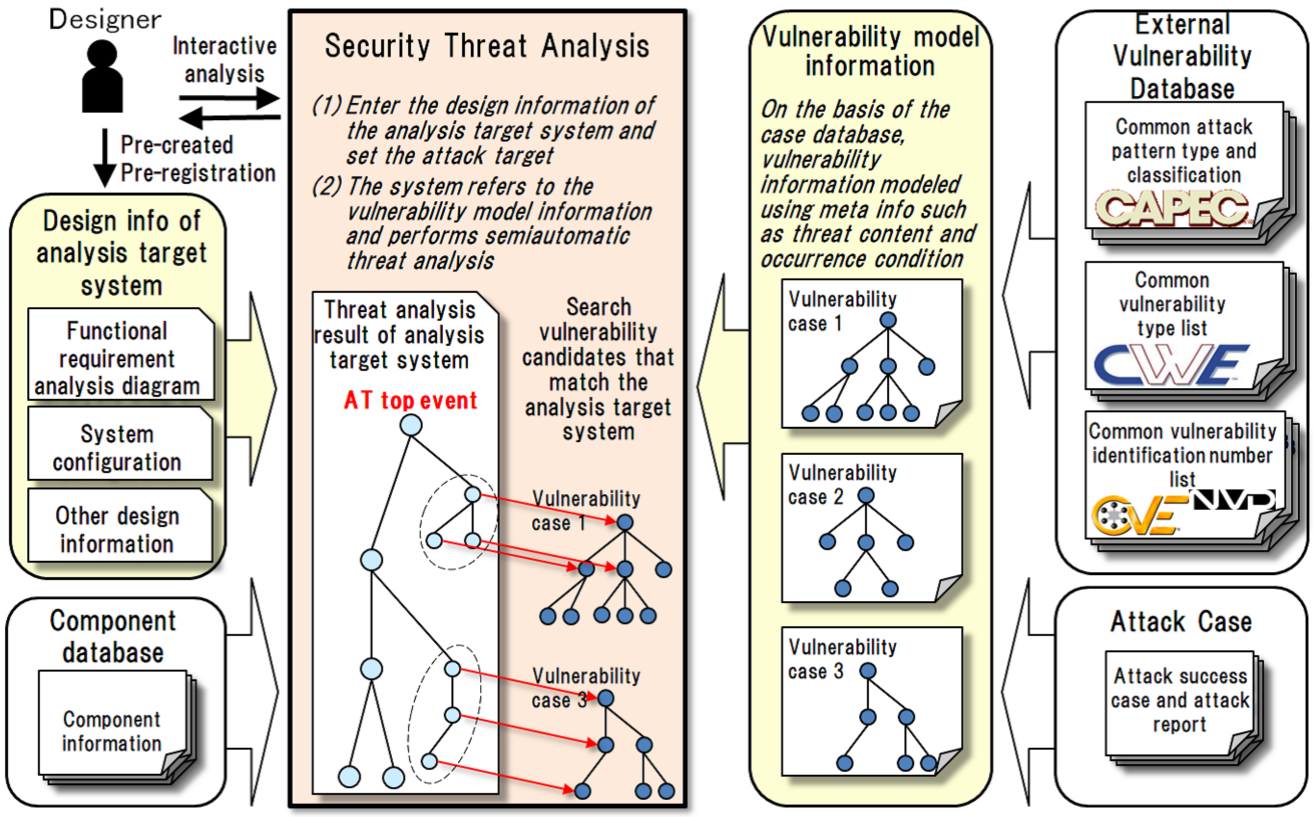

Designer
Interactive analysis
Pre-created
Pre-registration
Design info of analysis target system
Functional requirement analysis diagram
System configuration
Other design information
Component database
Component information
Security Threat Analysis
(1) Enter the design information of the analysis target system and set the attack target
(2) The system refers to the vulnerability model information and performs semiautomatic threat analysis
Threat analysis result of analysis target system
AT top event
Search vulnerability candidates that match the analysis target system
Vulnerability case 1
Vulnerability case 3
Vulnerability model information
On the basis of the case database, vulnerability information modeled using meta info such as threat content and occurrence condition
Vulnerability case 1
Vulnerability case 2
Vulnerability case 3
External Vulnerability Database
Common attack pattern type and classification
CAPEC
Common vulnerability type list
CWE
Common vulnerability identification number list
CVE
NVD
Attack Case
Attack success case and attack report

Figure 1. Overview of proposed threat analysis method

- Create vulnerability model information.
- Create lower-level component information embedded in software.
- Perform threat analysis on the basis of design information of analysis target system.

A. *Creating vulnerability model information*

The MITRE Corporation has published several forms of vulnerability DBs [4]–[6]. However, it is difficult to create an AT for a concrete target (for example, a connected car) simply by referring to these DBs, because elaborate attacks occur with multiple combinations of vulnerabilities. We will create an AT with a reference to existing attack case literature, reports, etc. Thus, let the AT be obtained from the existing vulnerability DB and existing attack report be called the first_AT. This first_AT is hierarchically drawn into a top node, a collection of intermediate nodes and bottom nodes. A single first_AT is created for each vulnerability. A vulnerability DB such as CVE monotonically increases, so it is not necessary to recreate the first_AT once it has been generated.

B. *Proposal of component database*

In some configurations of embedded systems, such as those for automobiles and in general the Internet of Things (IoT), required lower-level components embedded within the software, not the software itself, are incorporated. However, a vulnerability DB such as CVE only includes vulnerability information for software and does not describe information on the lower-level components embedded within the software. Therefore, a correspondence table between the software version and the version for its lower-level components would be beneficial. This makes it easy to check vulnerability information at the manufacturing stage of embedded systems such as those in IoT devices.

Specifically, using Tesla's browser hacking case as an example, the "UserAgent" property defined in Tesla's browser is "Mozilla / 5.0 (X11; Linux) AppleWebKit / 534.34 (HTML, like Gecko) QtCarBrowser Safari / 534.34." In contrast, the related vulnerability outlined in CVE describes "Google Chrome before 16.0.912.77". In this case, Chrome itself is not used, but the WebKit component built into Chrome is used. Therefore, a correspondence table indicating the version of the component built into certain software is required. Table I shows an example of Google Chrome's component DB. The method to create a component DB is outside the scope of this proposal.

TABLE I. Example of Google Chrome's component DB

Version	Release date	Layout engine
0.2.149	2008-09-02	WebKit 522
0.3.154	2008-10-29	WebKit 522
0.4.154	2008-11-24	WebKit 525
...	...	...
10.0.648	2011-03-08	WebKit 534.16
11.0.696	2011-04-27	WebKit 534.24
12.0.742	2011-06-07	WebKit 534.30
13.0.782	2011-08-02	WebKit 535.1
14.0.835	2011-09-16	WebKit 535.1
...	...	...
56.0.2924	2016-12-08	Blink 537.36

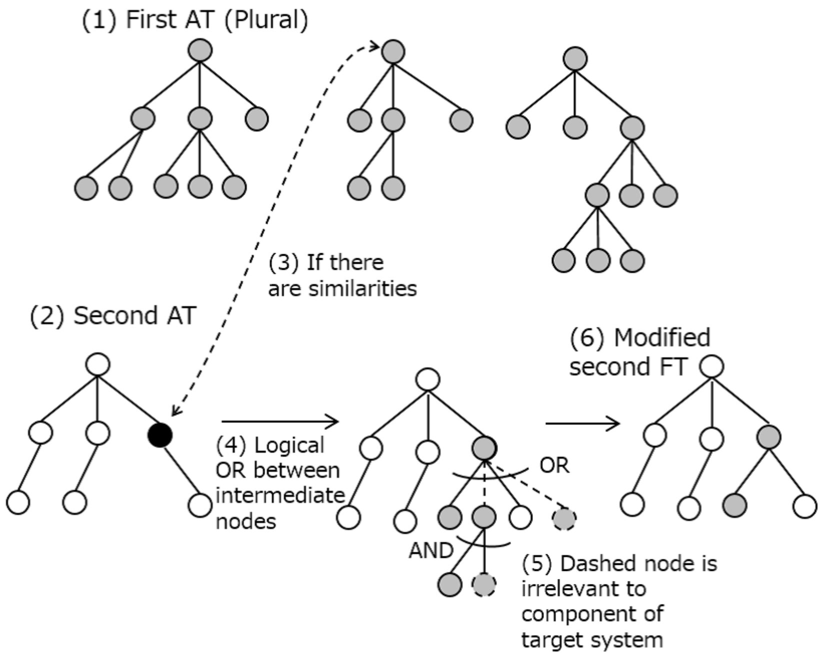

(1) First AT (Plural)
(3) If there are similarities
(2) Second AT
(6) Modified second FT
(4) Logical OR between intermediate nodes
OR
AND
(5) Dashed node is irrelevant to component of target system

Figure 2. Threat analysis algorithm (cited from reference [8])

C. Threat analysis algorithm

This section describes the threat analysis algorithm. It corresponds to the "Safety & Security Threat Analysis" section in Figure 1. The algorithm, which is based on the vulnerability model shown in Section II-A, the component DB shown in Section II-B, and the design information of the analysis target system, is as follows:

(1) Create a second_AT with the top node as a safety accident related to the evaluation target system. At this time, even if the component is not directly included in the evaluation target system, a component judged to be related by referring to the component DB is included in the second_AT (the black circle node in Figure 2 (2)). The second_AT is hierarchically depicted using the top node, the multiple intermediate nodes, and the lowest nodes. Thus, a second_AT is created (Figure 2 (2)).

(2) One of the top nodes or intermediate nodes of the second_AT is selected and Natural Language Processing (NLP) is used to mechanically determine whether there are first_ATs having a natural language expression similar to nodes of the second_AT (Figure 2 (3)). If this is the case, the first_AT is temporarily added to the second_AT (Figure 2 (4)). OR gate is attached to the node of the second_AT temporarily, and the first_AT is attached below it. This is done for all nodes of the second_AT. As a result, the second_AT is expanded more after considering the existing vulnerability database, that is, the entire set of the first_AT.

(3) The focus is now on the temporary added nodes in the expanded second_AT. We check whether the added node is necessary. Specifically, we define a node unrelated to the component of the second_AT (different components or different versions) as FALSE nodes, and the FALSE node and the AND gate that is just above the FALSE node are deleted (Figure 2 (5)).

(4) Repeat steps 1–3 for all the first_ATs that are related to the second_AT as described above. After the modification, we evaluate the occurrence probability of the top node by using the modified second_AT.

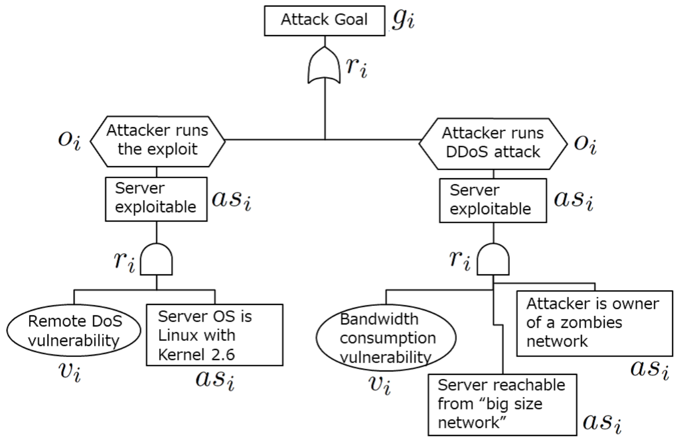

Attack Goal
g_i
r_i
o_i
Attacker runs the exploit
Attacker runs DDoS attack
o_i
Server exploitable
as_i
Server exploitable
as_i
r_i
r_i
Remote DoS vulnerability
v_i
Server OS is Linux with Kernel 2.6
as_i
Bandwidth consumption vulnerability
v_i
Attacker is owner of a zombies network
as_i
Server reachable from "big size network"
as_i

Figure 3. Example of AT (quoted from Figure 2, cited from reference [3])

III. Formulation of proposed algorithm

In this section, we formulate the algorithm shown in Section II-C.

A. Definition

The definition of the attack tree AT according to reference [3] is shown below. An example of AT is shown in Figure 3.

$$\boldsymbol{G} = \{g_i\} : AttackGoals \quad (1)$$
$$\boldsymbol{O} = \{o_i\} : Operations \quad (2)$$
$$\boldsymbol{AS} = \{as_i\} : Assertions \quad (3)$$
$$\boldsymbol{V} = \{v_i\} : Vulnerabilities \quad (4)$$
$$\boldsymbol{R} = \{r_i\} : Relationships \quad (5)$$

Here, an attack goal is the goal of all potential cyber attacks, and operations represent all the basic actions (reads, writes, etc.) that can be performed by either the attacker or the operator of the system. An assertion is a statement (for example, "Web server is not patched") representing "conditions to be verified" in order to take account of the actual branching of the attack tree. Vulnerability is a known vulnerability. Relationships are relationships that exist between elements that make up an attack tree (that is, attack goals, operations, assertions, vulnerabilities). The attack tree AT_k is defined as follows.

$$AT_k = \{g_i, \boldsymbol{O}_i, \boldsymbol{AS}_i, \boldsymbol{V}_i, \boldsymbol{R}_i\} \quad (6)$$

Here, $g_i \in \boldsymbol{G}, \boldsymbol{O}_i \subseteq \boldsymbol{O}, \boldsymbol{AS}_i \subseteq \boldsymbol{AS}, \boldsymbol{V}_i \subseteq \boldsymbol{V}, \boldsymbol{R}_i$ is a set of relationships.

All ATs have one main goal g, and the logic gate output (upper side) becomes an assertion.

B. Formulation of proposed algorithm

The first attack tree AT_k^1 and the second attack tree AT^2 are defined as follows.

$$AT_k^1 = \{g_k, \boldsymbol{O}_k, \boldsymbol{AS}_k, \boldsymbol{V}_k, \boldsymbol{R}_k\} \quad (7)$$
$$AT^2 = \{g_j, \boldsymbol{O}_j, \boldsymbol{AS}_j, \boldsymbol{V}_j, \boldsymbol{R}_j\} \quad (8)$$

Next, look for k, which is $g_j = g_k$ or $as_l \approx g_k$. However, it is $as_l \in \boldsymbol{AS_j}$. In addition, look for n and m, which is $as_n \approx as_m$. However, $as_n \in \boldsymbol{AS_k}, as_m \in \boldsymbol{AS_j}$.

Here, $x \approx y$ means "Comparing the descriptions of both sides with words, it is judged that x and y are close."

Next, update the second attack tree AT^2 as follows.

$$AT^2 = \{ \; g_j, \boldsymbol{O}_j \cup \boldsymbol{O}_k \cup \boldsymbol{O}_n, \boldsymbol{AS}_j \cup \boldsymbol{AS}_k \cup \boldsymbol{AS}_n, \boldsymbol{V}_j \cup \boldsymbol{V}_k \cup \boldsymbol{V}_n, \boldsymbol{R}_j \cup \boldsymbol{R}_k \cup \boldsymbol{R}_n \backslash \boldsymbol{R}'\} \quad (9)$$

Here, $\backslash$ represents the difference set. Also, $\boldsymbol{R}'$ is $\boldsymbol{R}' = \boldsymbol{R}'_{OR} \cup \boldsymbol{R}'_{AND}$. $\boldsymbol{R}'_{OR}$ is the relationship of the FALSE node, and $\boldsymbol{R}'_{AND}$ is the relationship of the upper nodes of the AND relationship just above the FALSE node. A FALSE node is $o \in \boldsymbol{O}_k \cup \boldsymbol{O}_n, as \in \boldsymbol{AS}_k \cup \boldsymbol{AS}_n, v \in \boldsymbol{V}_k \cup \boldsymbol{V}_n$ that is unrelated to the components of AT^2 (such as different components and different versions).

C. Calculation of attack probability

According to the formulation in the previous section, it is possible to calculate the probability of attack with the following formula using the calculation method of the conventional research [3].

If the inputs to the logic gates are independent, the probability of the output value from the ith AND gate $P_{out}AND_i$ and the probability of the output value from the ith OR gate $P_{out}OR_i$ are as follows.

$$P_{out}AND_i = \prod_{k=1}^{n} P_{in}(k,i) \quad (10)$$

$$P_{out}OR_i = \sum_{k=1}^{n} P_{in}(k,i) \quad (11)$$

However, $P_{in}(k,i)$ is the probability of the input of the kth input to the ith gate with n inputs $(1 \leq k \leq n)$.

In addition, in reference [3], calculation formulas when the inputs to the logic gates are not independent are also shown. Furthermore, reference [3] suggests rewriting the operation node with an AND gate and an assertion in order to obtain the probability of the top event (attack goal) of the attack tree. Specifically, replace the operation node with an AND gate, substitute the description of the original operation as an assertion, and input it as an input to the AND gate. With this replacement, the description of the operation disappears from the attack tree, and the probability of the top event can be calculated by sequentially calculating the above expression (10) and expression (11). In addition, reference [8] describes the application of actual cases of car attacks [9][10].

IV. APPLICATION OF PROPOSED METHOD TO ACTUAL CASE

In this section, we apply the proposed algorithm to examples in the Tesla case [9].

A. Creating first_AT

First, a first_AT is created on the basis of the vulnerability DB. We must create first_ATs for all vulnerabilities. In this case, we created a first_AT for CVE-2011-3928 as shown in Figure 4.

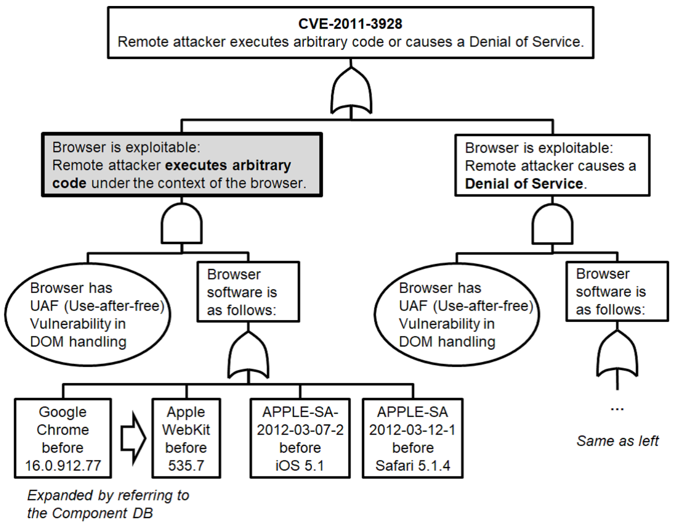

CVE-2011-3928
Remote attacker executes arbitrary code or causes a Denial of Service.
Browser is exploitable: Remote attacker executes arbitrary code under the context of the browser.
Browser is exploitable: Remote attacker causes a Denial of Service.
Browser has UAF (Use-after-free) Vulnerability in DOM handling
Browser software is as follows:
Browser has UAF (Use-after-free) Vulnerability in DOM handling
Browser software is as follows:
Google Chrome before 16.0.912.77
Apple WebKit before 535.7
APPLE-SA-2012-03-07-2 before iOS 5.1
APPLE-SA 2012-03-12-1 before Safari 5.1.4
...
Same as left
Expanded by referring to the Component DB

Figure 4. first_AT created from CVE-2011-3928 (cited from reference [8])

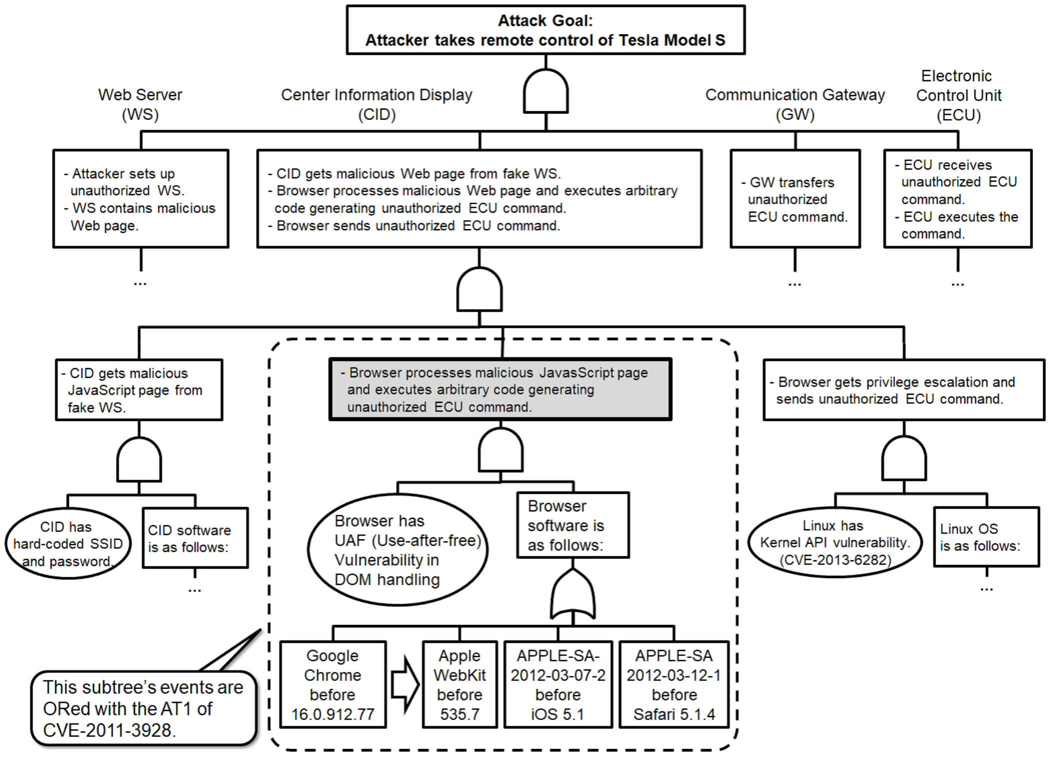

Attack Goal: Attacker takes remote control of Tesla Model S
Web Server (WS)
Center Information Display (CID)
Communication Gateway (GW)
Electronic Control Unit (ECU)
- Attacker sets up unauthorized WS. - WS contains malicious Web page.
- CID gets malicious Web page from fake WS. - Browser processes malicious Web page and executes arbitrary code generating unauthorized ECU command. - Browser sends unauthorized ECU command.
- GW transfers unauthorized ECU command.
- ECU receives unauthorized ECU command. - ECU executes the command.
- CID gets malicious JavaScript page from fake WS.
- Browser processes malicious JavasScript page and executes arbitrary code generating unauthorized ECU command.
- Browser gets privilege escalation and sends unauthorized ECU command.
CID has hard-coded SSID and password.
CID software is as follows:
Browser has UAF (Use-after-free) Vulnerability in DOM handling
Browser software is as follows:
Linux has Kernel API vulnerability. (CVE-2013-6282)
Linux OS is as follows:
This subtree's events are ORed with the AT1 of CVE-2011-3928.
Google Chrome before 16.0.912.77
Apple WebKit before 535.7
APPLE-SA-2012-03-07-2 before iOS 5.1
APPLE-SA 2012-03-12-1 before Safari 5.1.4

Figure 5. second_AT created from Tesla model S case

B. Application to case of Tesla model S

Next, we attempted to create a second_AT for the Tesla model S case [9]. The second_AT we created is shown in Figure 5. The dashed-line area of Figure 5 is the resultant subtree of combining the second_AT with the first_AT shown in Figure 4.

To apply our method to this case, matching of the nodes shown in gray in Figure 4 and Figure 5 was performed manually by humans. The next section will outline how this matching process will be performed automatically using topic model analysis.

V. Topic Model Analysis

In this section, we describe latent Dirichlet allocation (LDA), which is a method of topic model analysis and cosine similarity.

A. LDA

Topic models are formed under the notion that a document contains a number of latent topics, with each keyword either attributing to a certain topic or being generated as a result of said topic. In topic model analysis, we estimate latent topics from keywords. One of the analysis methods of topic models is LDA [12]. This is a language model that assumes the probability distribution of the topic (parameter θ of the multinomial distribution) follows the Dirichlet distribution. In LDA, topics are selected in accordance with the Dirichlet distribution and words are selected in accordance with the probability distribution of words for that topic.

B. Formulation of LDA

The LDA can be represented by the graphical model shown in Figure 6.

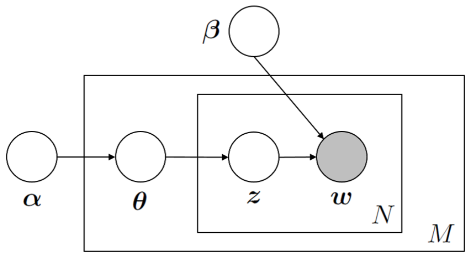


Figure 6. Graphical model representation of LDA (cited from reference [12])

Here, d is the number ID of a document, n is the number ID of a word in a document, and k is the number ID of a topic. M is the number of documents and N and K are the number of words and topics in a document, respectively. The number of words in document d is represented as N_d. The ranges of d, n, and k are $1 \leq d \leq M$, $1 \leq n \leq N_d$, and $1 \leq k \leq K$, respectively. w_{dn} represents the nth word of document d. z_{dn} represents the latent topic of the nth word in the document d. θ_{dk} is the mixing ratio of the latent topic k of document d. For example, if the number of topics of document d is 3, the mixing ratios of topics 1, 2, and 3 are 10%, 70%, and 20%, respectively then $\theta_{d1} = 0.1$, $\theta_{d2} = 0.7$, $\theta_{d3} = 0.2$, in which $\boldsymbol{\theta}_d = \{0.1, 0.7, 0.2\}$. $\boldsymbol{\alpha}$ is a hyper parameter related to the mixing ratio of the latent topic, and $\boldsymbol{\beta}$ is one related to the word generation rate. A set of documents is called a corpus, and a corpus of M documents is denoted by $\boldsymbol{D} = \{\boldsymbol{w_1}, \boldsymbol{w_2}, \cdots, \boldsymbol{w_M}\}$. We obtain the probability of a corpus as follows:

$$p(\boldsymbol{D}|\boldsymbol{\alpha},\boldsymbol{\beta}) = \prod_{d=1}^{M} \int p(\boldsymbol{\theta}_d|\boldsymbol{\alpha}) \left(\prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_{dn}|\boldsymbol{\theta}_d) p(w_{dn}|z_{dn},\boldsymbol{\beta}) \right) d\boldsymbol{\theta_d} \quad (12)$$

Here, we can only observe the word w_{dn}. We want to know from which topic those words were generated. In other words, when document d is given, we calculate the probability distribution of $\boldsymbol{\theta}_d$ and $\boldsymbol{z}$ as follows:

$$p(\boldsymbol{\theta}, \boldsymbol{z}|\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{p(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{w}|\boldsymbol{\alpha}, \boldsymbol{\beta})}{p(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})} \quad (13)$$

The variable Bayes method [12], Markov chain Monte Carlo (MCMC) [13], Gibbs sampling method [14], which is one kind of MCMC, etc. are proposed as a method to solve the above equation.

C. Cosine similarity

We can compute the similarity between document d and d' by calculating the cosine similarity for $\boldsymbol{\theta}_d$ obtained approximately by the above LDA method. The cosine similarity can be expressed by the following equation.

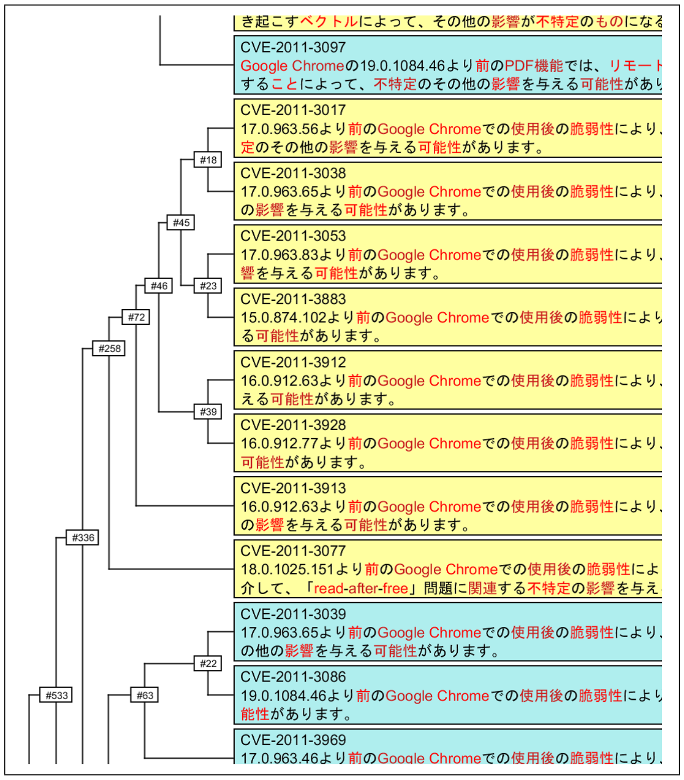
$$sim_{cos}(d, d') = \frac{\sum_{k=1}^{K} \theta_{dk}\theta_{d'k}}{\sqrt{\sum_{k=1}^{K} {\theta_{dk}}^2}\sqrt{\sum_{k=1}^{K} {\theta_{d'k}}^2}} \quad (14)$$

D. Topic model analysis tool

The National Institute of Advanced Industrial Science and Technology (AIST) has developed a security requirement analysis support tool using topic model analysis technology including LDA [15]. We preliminarily used this tool to verify whether the vast number of vulnerabilities CVE [4] listed in the order of discovery can be organized into a hierarchical structure by topic model analysis. Figure 7 shows the result of using 1500 cases from CVE-2011-3001 to CVE-2011-4500 after translating it to Japanese using Google Translate [16]. As shown in Figure 7, we see that similar vulnerabilities are classified near the hierarchical structure. Figure 7 is written in Japanese, the boxes in CVE-2011-3017 to CVE2011-3077 are described as "Use-after-free vulnerability in Google Chrome before xx.0.xxx.xx allows remote attackers to cause ... or possibly have unspecified other impact ...".

VI. Matching Attack Cases to Vulnerability Database

In this section, we describe the method and results of experiments that match attack cases and vulnerability databases.



き起こすベクトルによって、その他の影響が不特定のものになる

CVE-2011-3097
Google Chromeの19.0.1084.46より前のPDF機能では、リモート
することによって、不特定のその他の影響を与える可能性があり

CVE-2011-3017
17.0.963.56より前のGoogle Chromeでの使用後の脆弱性により、
定のその他の影響を与える可能性があります。

CVE-2011-3038
17.0.963.65より前のGoogle Chromeでの使用後の脆弱性により、
の影響を与える可能性があります。

CVE-2011-3053
17.0.963.83より前のGoogle Chromeでの使用後の脆弱性により、
響を与える可能性があります。

CVE-2011-3883
15.0.874.102より前のGoogle Chromeでの使用後の脆弱性により
る可能性があります。

CVE-2011-3912
16.0.912.63より前のGoogle Chromeでの使用後の脆弱性により、
える可能性があります。

CVE-2011-3928
16.0.912.77より前のGoogle Chromeでの使用後の脆弱性により、
可能性があります。

CVE-2011-3913
16.0.912.63より前のGoogle Chromeでの使用後の脆弱性により、
の影響を与える可能性があります。

CVE-2011-3077
18.0.1025.151より前のGoogle Chromeでの使用後の脆弱性によ
介して、「read-after-free」問題に関連する不特定の影響を与え

CVE-2011-3039
17.0.963.65より前のGoogle Chromeでの使用後の脆弱性により、
の他の影響を与える可能性があります。

CVE-2011-3086
19.0.1084.46より前のGoogle Chromeでの使用後の脆弱性により
能性があります。

CVE-2011-3969
17.0.963.46より前のGoogle Chromeでの使用後の脆弱性により、

Figure 7. Example segment of vulnerability DB CVE hierarchy

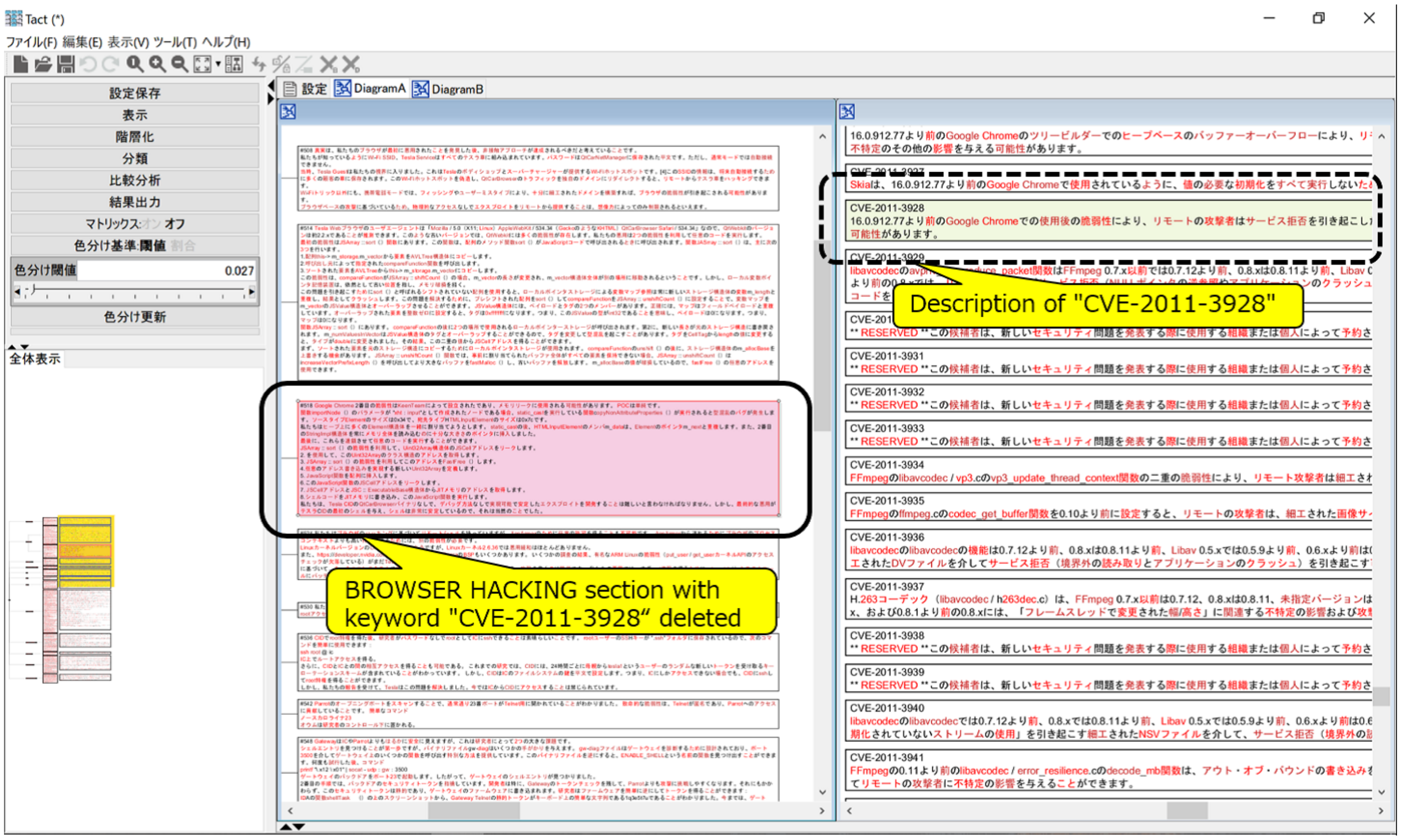
A. Outline explanation

As mentioned in Section II-C(2), we used NLP when matching and connecting the first_AT and the second_AT nodes. We verified the feasibility of this matching process.

We have investigated on various reports to find vulnerabilities that should be related in the second_AT of the target system. However, depending on the report, the procedure of attack is shown but the concrete CVE number is not specified. Even in such a case, we can extract the corresponding CVE number from the attack description described in natural language.

To achieve this, we must find a node of the second_AT that conceivably matches the description in CVE. However, a mechanical word matching process will probably not lead to a correct result as it is dependent on the words used to describe sentences. The context or meaning of the known attack description in each report should be thoroughly examined. Therefore, we have focused on the sentences of existing papers. Specifically, we have focused on the actual case of a car attack [9]. The process flow is as follows.

We translated the paper [9] into Japanese by using Google Translate because the tool we used only corresponded to Japanese. An advantage of utilizing such a translation is that it can prevent notation fluctuation of terms. The impact of Google translation will be discussed in the appendix. Since the section on BROWSER HACKING is long and its content is related to two vulnerabilities, it was divided into two. The vulnerabilities in question were CVE-2011-3928 and CVE-2013-6282. CVE-2011-3928 is described in the section on BROWSER HACKING, and CVE-2013-6282 is described in



16.0.912.77より前のGoogle Chromeのツリービルダーでのヒープベースのバッファーオーバーフローにより、リ
不特定のその他の影響を与える可能性があります。

CVE-2011-3927
Skiaは、16.0.912.77より前のGoogle Chromeで使用されているように、値の必要な初期化をすべて実行しないた

CVE-2011-3928
16.0.912.77より前のGoogle Chromeでの使用後の脆弱性により、リモートの攻撃者はサービス拒否を引き起こし
可能性があります。

CVE-2011-3929
libavcodecのavpr… packet関数はFFmpeg 0.7.x以前では0.7.12より前、0.8.xは0.8.11より前、Libav 0
より前の0.8.xでは… ビス拒否（NULLポインタの逆参照やアプリケーションのクラッシュ
コードを

CVE-201
** RESERVED **この候補者は、新しいセキュリティ問題を発表する際に使用する組織または個人によって予約さ

CVE-2011-3931
** RESERVED **この候補者は、新しいセキュリティ問題を発表する際に使用する組織または個人によって予約さ

CVE-2011-3932
** RESERVED **この候補者は、新しいセキュリティ問題を発表する際に使用する組織または個人によって予約さ

CVE-2011-3933
** RESERVED **この候補者は、新しいセキュリティ問題を発表する際に使用する組織または個人によって予約さ

CVE-2011-3934
FFmpegのlibavcodec / vp3.cのvp3_update_thread_context関数の二重の脆弱性により、リモート攻撃者は細工され

CVE-2011-3935
FFmpegのffmpeg.cのcodec_get_buffer関数を0.10より前に設定すると、リモートの攻撃者は、細工された画像サ

CVE-2011-3936
libavcodecのlibavcodecの機能は0.7.12より前、0.8.xは0.8.11より前、Libav 0.5.xでは0.5.9より前、0.6.xより前は0
工されたDVファイルを介してサービス拒否（境界外の読み取りとアプリケーションのクラッシュ）を引き起こす

CVE-2011-3937
H.263コーデック（libavcodec / h263dec.c）は、FFmpeg 0.7.x以前は0.7.12、0.8.xは0.8.11、未指定バージョンは
x、および0.8.1より前の0.8.xには、「フレームスレッドで変更された幅/高さ」に関連する不特定の影響および攻

CVE-2011-3938
** RESERVED **この候補者は、新しいセキュリティ問題を発表する際に使用する組織または個人によって予約さ

CVE-2011-3939
** RESERVED **この候補者は、新しいセキュリティ問題を発表する際に使用する組織または個人によって予約さ

CVE-2011-3940
libavcodecのlibavcodecでは0.7.12より前、0.8.xでは0.8.11より前、Libav 0.5.xでは0.5.9より前、0.6.xより前は0.6
期化されていないストリームの使用」を引き起こす細工されたNSVファイルを介して、サービス拒否（境界外の

CVE-2011-3941
FFmpegの0.11より前のlibavcodec / error_resilience.cのdecode_mb関数は、アウト・オブ・バウンドの書き込みを
てリモートの攻撃者に不特定の影響を与えることができます。

Figure 8. Matching attack cases to vulnerability DBs

the section on LOCAL PRIVILEGE ESCALATION. If “CVE-2011-3928” or “CVE-2013-6282” is included as a keyword, it may be detected by keyword matching, so the keywords “CVE-2011-3928” and “CVE-2013-6282” were deleted from BROWSER HACKING and LOCAL PRIVILEGE ESCALATION, respectively.

However, regarding BROWSER HACKING, there is a problem of component inclusion relationship stated in Section II-B, and the keyword “Google Chrome” is added to the sentences in which WebKit is described. This is considered to be equivalent to referring to the component DB of the proposed method. Since the topic analysis tool used has an upper limit on the number of items to be handled, it was not possible to cover all CVEs, so we targeted 500 items before and after including the target vulnerability. The limitation of 500 items is not a constraint of the topic model analysis, but an implementation limitation of the tools we used.

We specifically targeted CVEs from CVE-2011-3501 to CVE-2011-4000 including CVE-2011-3928 and those from CVE-2013-6001 to CVE-2013-6500 including CVE-2013-6282. For each section of the paper and each CVE vulnerability, similar sentences were evaluated by topic model analysis. The keyword extraction method was known as “noun and Kana”, the feature quantity extraction method was “LDA”, and the sentence similarity “Cosine” option was used.

B. Analysis result

The result of matching each section of the paper to each CVE vulnerability is shown in Figure 8. Figure 9 shows the enlarged view of BROWSER HACKING section of Figure 8, and Figure 10 shows enlarged view of the description of “CVE-2011-3928” in Figure 8. When we click on a sentence in the left pane, this tool will highlight similar sentences in the right pane. The solid lined area in the left pane is the BROWSER HACKING section with the keyword “CVE-2011-3928” deleted. When clicking on this area, the dashed lined area, which is the description of CVE-2011-3928 in the right pane, is highlighted and is judged to be similar. The number of items that included the appropriate CVE from the original 500 was filtered down to 22. It can be said that the smaller the number, the better. Regarding CVE-2013-6282, a similar result was obtained by matching the information of LOCAL PRIVILEGE ESCALATION with that of CVE, in this case 23 out of the 500.

C. Consideration of topic model analysis

In this research, we used a topic model, which treats a document as a set of words, and replaces distances between different documents with distances between different “sets of words” to measure proximity. Topic model analysis is one of the so-called AI methods that enables high-speed processing of big data and excels in clarifying the reason for the results. For example, the number of pages in the Tesla attack case paper [9], IRB 140 industrial robot attack case paper [11], and Jeep Cherokee attack cases paper [10] are 16, 48, and 91, respectively. Manually analyzing such a large amount of materials requires a great, if not greater, amount of work, placing a heavy burden one those doing it. Furthermore,

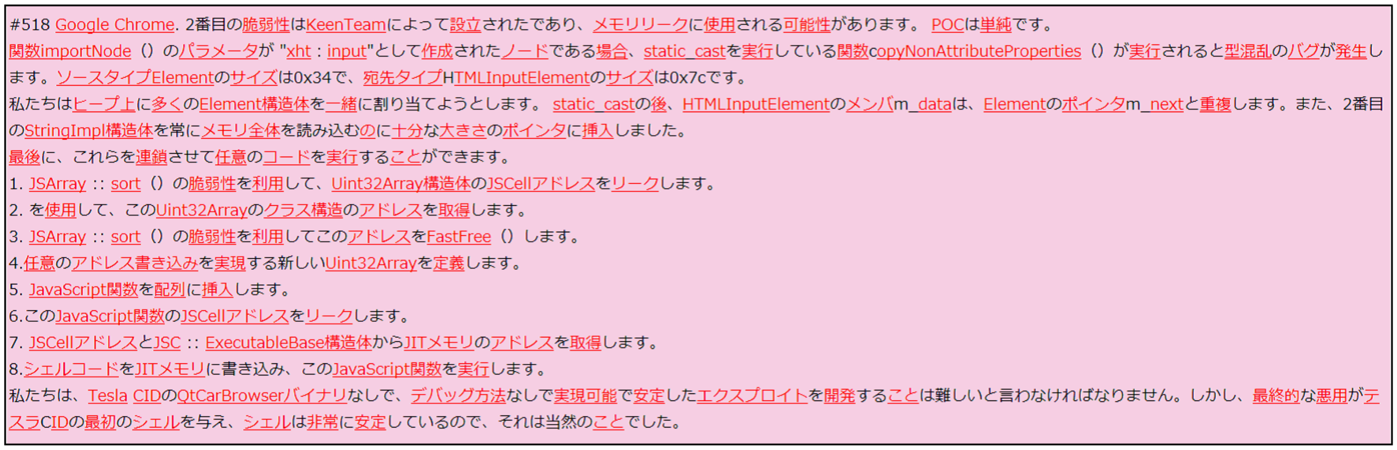

#518 Google Chrome. 2番目の脆弱性はKeenTeamによって設立されたであり、メモリリークに使用される可能性があります。 POCは単純です。
関数importNode（）のパラメータが "xht：input"として作成されたノードである場合、static_castを実行している関数copyNonAttributeProperties（）が実行されると型混乱のバグが発生します。ソースタイプElementのサイズは0x34で、宛先タイプHTMLInputElementのサイズは0x7cです。
私たちはヒープ上に多くのElement構造体を一緒に割り当てようとします。 static_castの後、HTMLInputElementのメンバm_dataは、Elementのポインタm_nextと重複します。また、2番目のStringImpl構造体を常にメモリ全体を読み込むのに十分な大きさのポインタに挿入しました。
最後に、これらを連鎖させて任意のコードを実行することができます。
1. JSArray :: sort（）の脆弱性を利用して、Uint32Array構造体のJSCellアドレスをリークします。
2. を使用して、このUint32Arrayのクラス構造のアドレスを取得します。
3. JSArray :: sort（）の脆弱性を利用してこのアドレスをFastFree（）します。
4.任意のアドレス書き込みを実現する新しいUint32Arrayを定義します。
5. JavaScript関数を配列に挿入します。
6.このJavaScript関数のJSCellアドレスをリークします。
7. JSCellアドレスとJSC :: ExecutableBase構造体からJITメモリのアドレスを取得します。
8.シェルコードをJITメモリに書き込み、このJavaScript関数を実行します。
私たちは、Tesla CIDのQtCarBrowserバイナリなしで、デバッグ方法なしで実現可能で安定したエクスプロイトを開発することは難しいと言わなければなりません。しかし、最終的な悪用がテスラCIDの最初のシェルを与え、シェルは非常に安定しているので、それは当然のことでした。

Figure 9. Enlarged view of BROWSER HACKING section shown in Figure 8

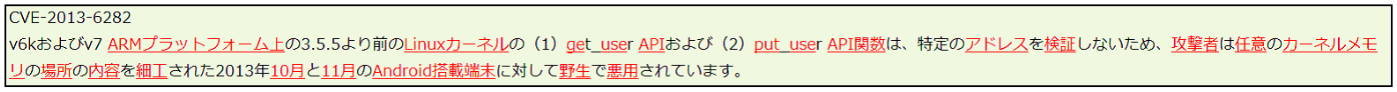

CVE-2013-6282
v6kおよびv7 ARMプラットフォーム上の3.5.5より前のLinuxカーネルの（1）get_user APIおよび（2）put_user API関数は、特定のアドレスを検証しないため、攻撃者は任意のカーネルメモリの場所の内容を細工された2013年10月と11月のAndroid搭載端末に対して野生で悪用されています。

Figure 10. Enlarged view of “CVE-2011-3928” description shown in Figure 8

considering that the number of attack papers will increase in the future, the proposed method of processing attack papers automatically with a topic model is significant.

VII. System Model Description

In this section, we create a system model description using the second attack tree shown in Figure 5 created using the proposed algorithm. This shows that we can recognize the dangers of related attacks using the description. We analyze the input and output of messages in the nodes directly under the attack goal node “Attacker takes remote control of Tesla Model S” in Figure 5 using the system model description. As a result, we achieve transparency in escalation of fake commands from an external information system to the in-vehicle control system.

A. *Date flow diagram*

A system model description of Tesla’s threat analysis target system is shown in Figure 11, which is the description result of the data flow diagram used for safety verification. The system to be analyzed consists of four hardware blocks, five software function modules, an operating system (OS), Web browser, Web server (WS), and various networks. The electronic control unit (ECU), communication gateway (GW), and center information display (CID) of the hardware block are installed inside the vehicle. The CID of the vehicle is connected to the WS over the Internet via Wi-Fi.

First, the in-vehicle control system will be described. The CID executes the vehicle information display function module, which inputs an instruction from the driver and creates a corresponding vehicle control command (for example, a command for monitoring the operating state of the engine, a command for unlocking the door, etc.) to the GW. The vehicle communication control function module of the GW converts the received command into an ECU command and transmits it to the ECU via the CAN bus. The vehicle control function module of the ECU activates the control logic inside the unit corresponding to the received ECU command and executes the vehicle control instructed by the driver.

Next, the external information system will be described. The Web information display function module of the CID inputs a driver’s instruction, creates a necessary Web page request command (for example, a GET command of the HTTP protocol), and transmits it to the WS via Wi-Fi. The Web server function module of the WS searches the corresponding Web page in accordance with the received command and returns the search result (for example, HTML content or JavaScript code).

At this stage, a safety analysis is carried out assuming that there is no vulnerability. Figure 11 shows that the in-vehicle control system and the external information system are separate systems, and there is no interference with each other.

B. *Add vulnerability*

Figure 12 expands the data flow diagram shown in Figure 11 to include the possibility of attack by adding Input / Output by vulnerability attributes as depicted by the gray boxes and dotted arrows.

As shown in Figure 5, the CID vulnerability is established by satisfying three conditions: “CID gets malicious JavaScript page from fake WS,” “Browser processes malicious JavaScript page and executes arbitrary code generating unauthorized ECU command,” and “Browser gets privilege escalation and sends unauthorized ECU command.” The first condition is established by “CID has hard-coded SSID and password.” The second condition applies to both “Browser has UAF (Use-after-free) Vulnerability in DOM handling” and “Apple WebKit before 535.7.” The third condition is established by “Linux has Kernel API vulnerability (CVE-2013-6282).”

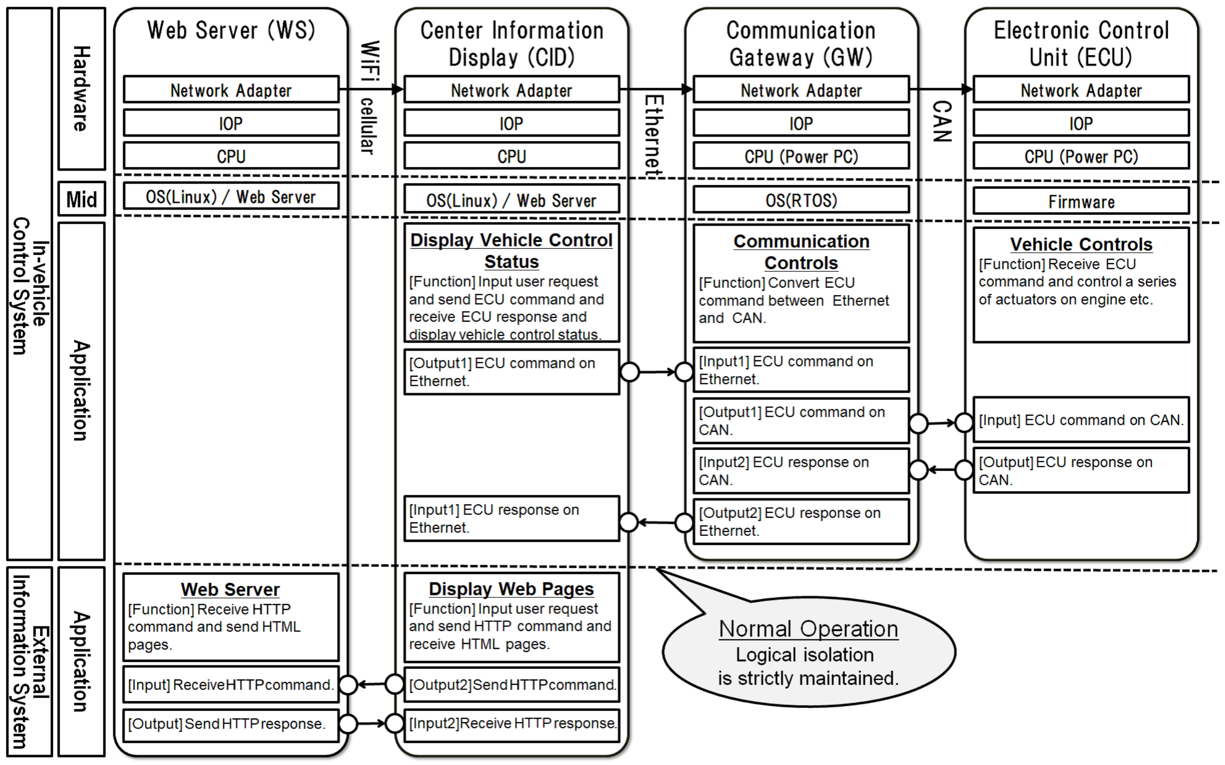

In-vehicle Control System
External Information System
Hardware
Mid
Application
Application
Web Server (WS)
Network Adapter
IOP
CPU
OS(Linux) / Web Server
Web Server
[Function] Receive HTTP command and send HTML pages.
[Input] Receive HTTP command.
[Output] Send HTTP response.
WiFi cellular
Center Information Display (CID)
Network Adapter
IOP
CPU
OS(Linux) / Web Server
Display Vehicle Control Status
[Function] Input user request and send ECU command and receive ECU response and display vehicle control status.
[Output1] ECU command on Ethernet.
[Input1] ECU response on Ethernet.
Display Web Pages
[Function] Input user request and send HTTP command and receive HTML pages.
[Output2] Send HTTP command.
[Input2] Receive HTTP response.
Ethernet
Communication Gateway (GW)
Network Adapter
IOP
CPU (Power PC)
OS(RTOS)
Communication Controls
[Function] Convert ECU command between Ethernet and CAN.
[Input1] ECU command on Ethernet.
[Output1] ECU command on CAN.
[Input2] ECU response on CAN.
[Output2] ECU response on Ethernet.
CAN
Electronic Control Unit (ECU)
Network Adapter
IOP
CPU (Power PC)
Firmware
Vehicle Controls
[Function] Receive ECU command and control a series of actuators on engine etc.
[Input] ECU command on CAN.
[Output] ECU response on CAN.
Normal Operation
Logical isolation is strictly maintained.

Figure 11. System model description

At this stage, as shown in Figure 12, it is possible to create a data flow diagram with attributes of these vulnerabilities added. Safety analysis is performed on this new data flow diagram. As a result, it became clear that fake commands can be transmitted from the external information system to the in-vehicle control system. With a fake command, it is possible to open and close the door of a running vehicle, and so on, demonstrating that a vehicle can be in danger.

VIII. Conclusion

In this paper, we propose a threat analysis method using topic model analysis and vulnerability DBs. We confirmed the feasibility of matching known attack cases to vulnerability DBs using a topic model analysis tool. Moreover, we showed that our proposed method can use the results of past threat analysis for the next threat analysis. In addition, we created a system model description based on the attack tree of Tesla's case created using the proposed method. We achieve transparency in escalation of fake commands from an external information system to the in-vehicle control system. We have shown by the results of our work, that our approach to automatic threat analysis support risk analysis in discovering previous unknown relationships and so threats including their potential escalation with an connected IT system.

However, this approach does not guarantee discovery and prevention of new sophisticated attacks that are completely different from those that occurred in the past. To detect previously unknown critical vulnerabilities, it is necessary to apply this method to advanced security engineering by artificial intelligence that utilize vulnerability DBs and system design information and evaluate it in actual cases.

Acknowledgment

Sven Wohlgemuth's contribution to this work is based on his research at Albert-Ludwig University, Freiburg, Germany, and other organizations before he joined Hitachi, Ltd. in February 2017. This work was supported by the Cabinet Office (CAO), Cross-ministerial Strategic Innovation Promotion Program (SIP), "Cybersecurity for Critical Infrastructure" (funding agency: NEDO).

References

[1] K. Umezawa, Y. Mishina, S. Wohlgemuth, and K. Takaragi, "Threat Analysis using Vulnerability Databases – Matching Attack Cases and Vulnerability Database by Topic Model Analysis –," Proceeding of the Third International Conference on Cyber-Technologies and Cyber-Systems (CYBER 2018), pp. 74-77, Nov. 2018.

[2] A. Ruddle et al., "Deliverable D2.3: Security requirements for automotive on-board networks based on dark-side scenarios," Seventh Research Framework Programme of the European Community, July 2008, pp. 1–138.

[3] I. N. Fovino, M. Masera, and A. D. Cian, "Integrating cyber attacks within fault trees," Reliability Engineering and System Safety 94, 2009, pp. 1394–1402.

[4] MITRE Corporation, "CVE - Common Vulnerability and Exposure," https://cve.mitre.org/ [retrieved: May, 2019]

[5] MITRE Corporation, "CWE List - Common Weakness Enumeration," https://cwe.mitre.org/data/ [retrieved: May, 2019]

[6] MITRE Corporation, "CAPEC - Common Attack Pattern Enumeration and Classification," https://capec.mitre.org/ [retrieved: May, 2019]

[7] K. Umezawa, Y. Mishina, K. Taguchi, and K. Takaragi, "A Proposal of Threat Analyses using Vulnerability Databases," Proceeding of the Symposium on Cryptography and Information Security (SCIS2018), 1C2-6, January 2018, pp. 1–8.

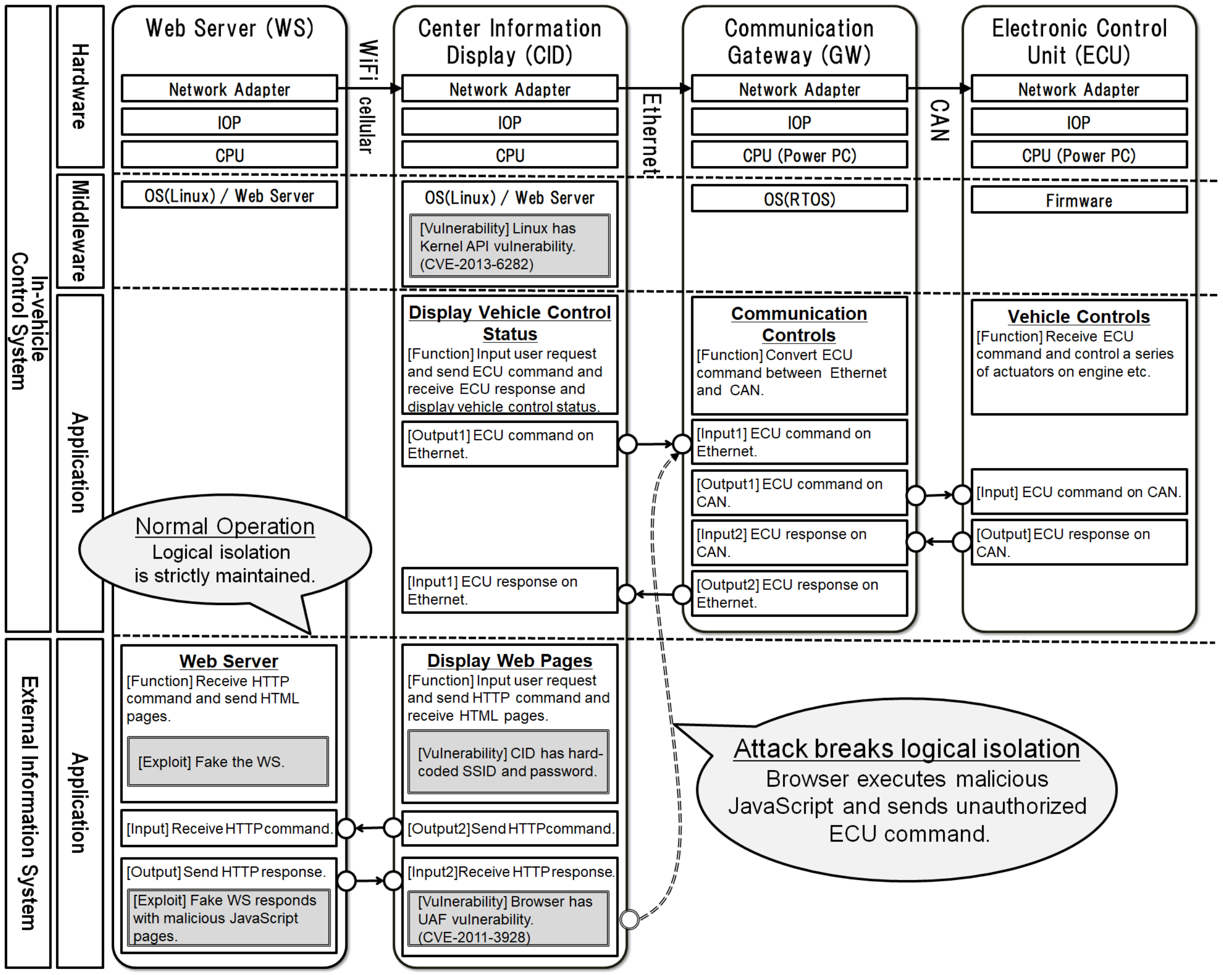

In-vehicle Control System
External Information System
Hardware
Middleware
Application
Application
Web Server (WS)
Center Information Display (CID)
Communication Gateway (GW)
Electronic Control Unit (ECU)
Network Adapter
IOP
CPU
CPU (Power PC)
WiFi cellular
Ethernet
CAN
OS(Linux) / Web Server
[Vulnerability] Linux has Kernel API vulnerability. (CVE-2013-6282)
OS(RTOS)
Firmware
Display Vehicle Control Status
[Function] Input user request and send ECU command and receive ECU response and display vehicle control status.
Communication Controls
[Function] Convert ECU command between Ethernet and CAN.
Vehicle Controls
[Function] Receive ECU command and control a series of actuators on engine etc.
[Output1] ECU command on Ethernet.
[Input1] ECU command on Ethernet.
[Output1] ECU command on CAN.
[Input] ECU command on CAN.
[Input2] ECU response on CAN.
[Output] ECU response on CAN.
[Input1] ECU response on Ethernet.
[Output2] ECU response on Ethernet.
Normal Operation
Logical isolation is strictly maintained.
Web Server
[Function] Receive HTTP command and send HTML pages.
[Exploit] Fake the WS.
Display Web Pages
[Function] Input user request and send HTTP command and receive HTML pages.
[Vulnerability] CID has hard-coded SSID and password.
Attack breaks logical isolation
Browser executes malicious JavaScript and sends unauthorized ECU command.
[Input] Receive HTTP command.
[Output2] Send HTTP command.
[Output] Send HTTP response.
[Input2] Receive HTTP response.
[Exploit] Fake WS responds with malicious JavaScript pages.
[Vulnerability] Browser has UAF vulnerability. (CVE-2011-3928)

Figure 12. System model description with vulnerabilities

[8] Y. Mishina, K. Takaragi, and K. Umezawa "A Proposal of Threat Analyses for Cyber-Physical System using Vulnerability Databases," 2018 IEEE International Symposium on Technologies for Homeland Security (IEEE HST), October 2018.

[9] S. Nie, L. Liu, and Y. Du, "Free-Fall: Hacking Tesla from Wireless to Can Bus," Briefing, Black Hat USA 2017, July 2017. pp. 1–16.

[10] C. Miller and C. Valasek, "Remote Exploitation of an Unaltered Passenger Vehicle," Briefing, Black Hat USA 2015, pp. 1–91.

[11] D. Quarta, M. Pogliani, M. Polino, A.M. Zanchettin, and S. Zaner, "Rogue Robots: Testing the Limits of an Industrial Robot's Security," Briefing, Black Hat USA 2017, July 2017.

[12] D. Blei, A. Ng, and M. Jordan, "Latent Dirichlet Allocation," Journal of Machine Learning Research, 2003, pp. 1107–1135.

[13] T. L. Griffiths and M. Steyvers, "Finding scientific topics," Proceedings of the National Academy of Science, 2004, pp. 5228-5235.

[14] Y. W. Teh, D. Newman, and M. Welling, "A Collapsed Variational Bayesian Inference Algorithm for Latent Dirichlet Allocation," Proceedings of Advances in Neural Information Processing Systems 19, NIPS '07, Cambridge, MA, pp. 1353–1360, 2007.

[15] K. Handa, H. Ohsaki, and I. Takeuti, "Security Requirements Analysis Supporting Tool: TACT," Information Processing Society of Japan (IPSJ) SIG Software Engineering (SIGSE), Proceeding of the Winter Workshop 2017. pp. 5–6.

[16] Y. Wu et al. "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation," arXiv:1609.08144, 2016. pp. 1–23.

[17] G. Biggs, T. Sakamoto, and T. Kotoku, "A profile and tool for modelling safety information with design information in SysML," Software & Systems Modeling 15, 1 (Jan 2016), pp. 147–178.

APPENDIX

As shown in Section VI, we translate English documents into Japanese by Google translator for tool constraints before analyzing. This section examines the effect of this translation.

Figures 13 and 15 show the descriptions from Figures 9 and 10 translated back into English using Google Translate, respectively. Figures 14 and 16 show the original English descriptions.

The word underlined is the one of attention by the analysis tool. The topic model analysis assumes a model ignoring the order of words and relations between words. Therefore, it is considered that the difference in the order of the Japanese and English words does not affect the analysis result. Also, when comparing Figure 13 and 14, only a few words that are similar in meaning with double underline (such as "established" and "founded", "occur" and "triggered") were different. We can see that most other words are being retranslated to the same word. In other words, it can be said that there is almost no mixing of errors due to translation.

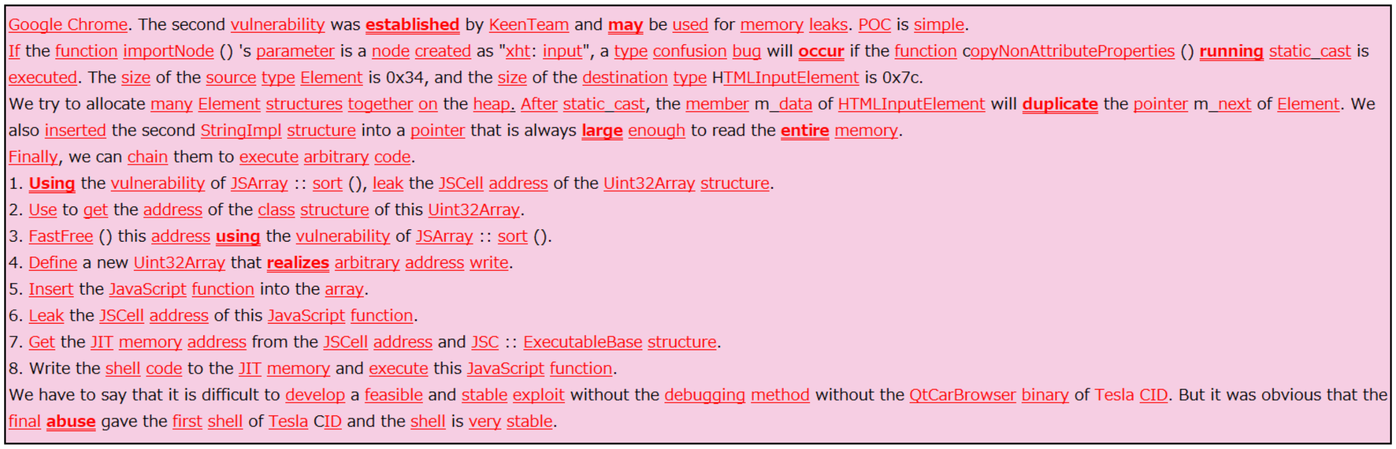

Google Chrome. The second vulnerability was **established** by KeenTeam and **may** be used for memory leaks. POC is simple.
If the function importNode () 's parameter is a node created as "xht: input", a type confusion bug will **occur** if the function copyNonAttributeProperties () **running** static_cast is executed. The size of the source type Element is 0x34, and the size of the destination type HTMLInputElement is 0x7c.
We try to allocate many Element structures together on the heap. After static_cast, the member m_data of HTMLInputElement will **duplicate** the pointer m_next of Element. We also inserted the second StringImpl structure into a pointer that is always **large** enough to read the **entire** memory.
Finally, we can chain them to execute arbitrary code.
1. **Using** the vulnerability of JSArray :: sort (), leak the JSCell address of the Uint32Array structure.
2. Use to get the address of the class structure of this Uint32Array.
3. FastFree () this address **using** the vulnerability of JSArray :: sort ().
4. Define a new Uint32Array that **realizes** arbitrary address write.
5. Insert the JavaScript function into the array.
6. Leak the JSCell address of this JavaScript function.
7. Get the JIT memory address from the JSCell address and JSC :: ExecutableBase structure.
8. Write the shell code to the JIT memory and execute this JavaScript function.
We have to say that it is difficult to develop a feasible and stable exploit without the debugging method without the QtCarBrowser binary of Tesla CID. But it was obvious that the final **abuse** gave the first shell of Tesla CID and the shell is very stable.

Figure 13. Description re-translated Figure 9 into English

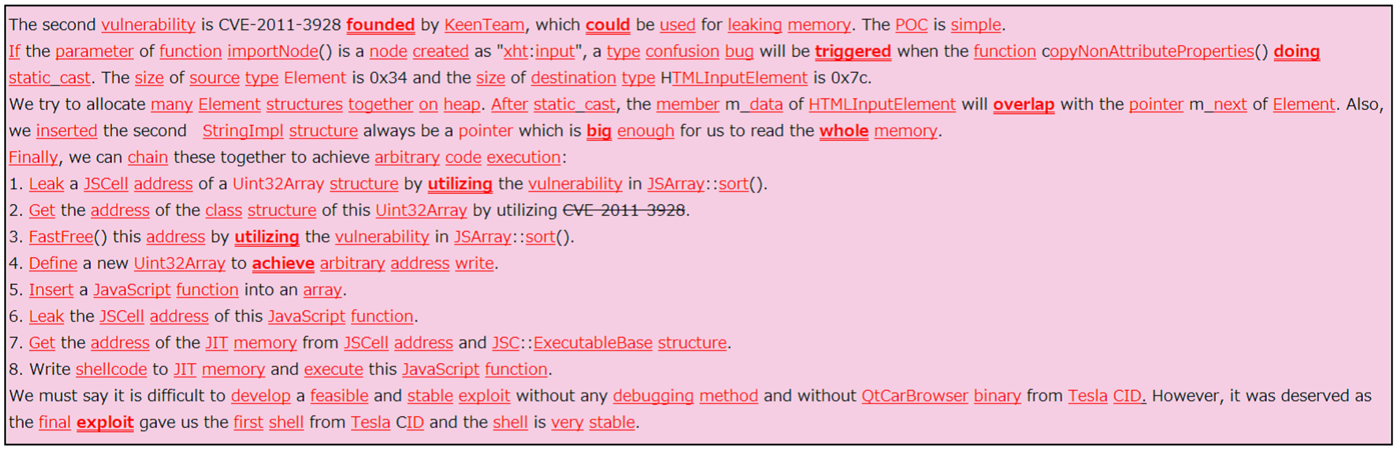

The second vulnerability is CVE-2011-3928 **founded** by KeenTeam, which **could** be used for leaking memory. The POC is simple.
If the parameter of function importNode() is a node created as "xht:input", a type confusion bug will be **triggered** when the function copyNonAttributeProperties() **doing** static_cast. The size of source type Element is 0x34 and the size of destination type HTMLInputElement is 0x7c.
We try to allocate many Element structures together on heap. After static_cast, the member m_data of HTMLInputElement will **overlap** with the pointer m_next of Element. Also, we inserted the second StringImpl structure always be a pointer which is **big** enough for us to read the **whole** memory.
Finally, we can chain these together to achieve arbitrary code execution:
1. Leak a JSCell address of a Uint32Array structure by **utilizing** the vulnerability in JSArray::sort().
2. Get the address of the class structure of this Uint32Array by utilizing ~~CVE-2011-3928~~.
3. FastFree() this address by **utilizing** the vulnerability in JSArray::sort().
4. Define a new Uint32Array to **achieve** arbitrary address write.
5. Insert a JavaScript function into an array.
6. Leak the JSCell address of this JavaScript function.
7. Get the address of the JIT memory from JSCell address and JSC::ExecutableBase structure.
8. Write shellcode to JIT memory and execute this JavaScript function.
We must say it is difficult to develop a feasible and stable exploit without any debugging method and without QtCarBrowser binary from Tesla CID. However, it was deserved as the final **exploit** gave us the first shell from Tesla CID and the shell is very stable.

Figure 14. Original description before translating Figure 9

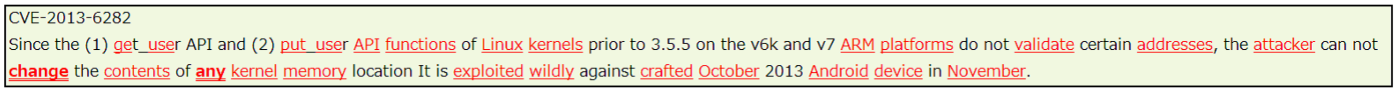

CVE-2013-6282
Since the (1) get_user API and (2) put_user API functions of Linux kernels prior to 3.5.5 on the v6k and v7 ARM platforms do not validate certain addresses, the attacker can not **change** the contents of **any** kernel memory location It is exploited wildly against crafted October 2013 Android device in November.

Figure 15. Description re-translated Figure 10 into English

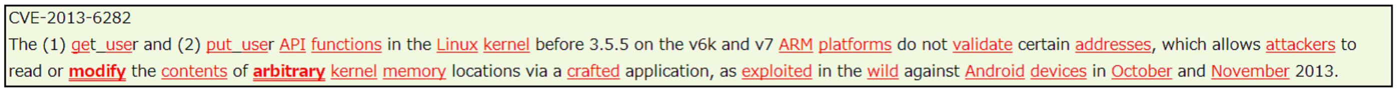

CVE-2013-6282
The (1) get_user and (2) put_user API functions in the Linux kernel before 3.5.5 on the v6k and v7 ARM platforms do not validate certain addresses, which allows attackers to read or **modify** the contents of **arbitrary** kernel memory locations via a crafted application, as exploited in the wild against Android devices in October and November 2013.

Figure 16. Original description before translating Figure 10

Don't Wait to be Breached! Creating Asymmetric Uncertainty of Cloud Applications via Moving Target Defenses

Kennedy A. Torkura
Hasso Plattner Institute
University of Potsdam, Germany
Email: kennedy.torkura@hpi.de

Christoph Meinel
Hasso Plattner Institute
University of Potsdam, Germany
christoph.meinel@hpi.de

Nane Kratzke
Lübeck University of Applied Sciences
Lübeck, Germany
nane.kratzke@th-luebeck.de

Abstract—**Cloud applications expose besides service endpoints also potential or actual vulnerabilities. Therefore, cloud security engineering efforts focus on hardening the fortress walls but seldom assume that attacks may be successful. At least against zero-day exploits, this approach is often toothless. Other than most security approaches and comparable to biological systems we accept that defensive "walls" can be breached at several layers. Instead of hardening the "fortress" walls we propose to make use of an (additional) active and adaptive defense system to attack potential intruders - an immune system that is inspired by the concept of a moving target defense. This "immune system" works on two layers. On the infrastructure layer, virtual machines are continuously regenerated (cell regeneration) to wipe out even undetected intruders. On the application level, the vertical and horizontal attack surface is continuously modified to circumvent successful replays of formerly scripted attacks. Our evaluations with two common cloud-native reference applications in popular cloud service infrastructures (Amazon Web Services, Google Compute Engine, Azure and OpenStack) show that it is technically possible to limit the time of attackers acting undetected down to minutes. Further, more than 98% of an attack surface can be changed automatically and minimized, which makes it hard for intruders to replay formerly successful scripted attacks. So, even if intruders get a foothold in the system, it is hard for them to maintain it. Therefore, our proposals are robust and dynamically change due in response to security threats similar to biological immune systems.**

Keywords*–*zero-day; exploit; moving target defense; microservice; cloud-native; application; security; asymmetric

I. Introduction

This paper extends ideas presented in [1] to improve cloud application security in the context of unknown zero-day exploits and reports on ongoing research in this field. Cloud computing enables a variety of innovative IT-enabled businesses and service models. Several research studies and programs focus on responsibly developing systems to ensure the security and privacy of users. But compliance with standards, audits, and checklists, does not automatically equal security [2] and there is a fundamental issue remaining. Zero-day vulnerabilities are computer-software vulnerabilities that are unknown to those who would be interested in mitigating the vulnerability (including the entity responsible for operating a cloud application). Until a vulnerability is mitigated, hackers can exploit it to adversely affect computer programs, data, additional computers or a network. For zero-day exploits, the probability that vulnerabilities are patched is zero, so the exploit should always succeed. Therefore, zero-day attacks are a severe threat, and we have to draw a scary conclusion: **In principle, attackers can establish footholds in our systems whenever they want.**

This contribution deals with the question how to build "unfair" cloud systems that permanently jangle attackers nerves. We present the latest results from our ongoing research that applies Moving Target Defense (MTD) principles on cloud runtime environment and cloud application layer.

Recent research [3], [4] made successfully use of elastic container platforms (see Table I) and their "designed for failure" capabilities to realize transferability of cloud-native applications at runtime. By transferability, the conducted research means that a cloud-native application can be moved from one IaaS provider infrastructure to another without any downtime. These platforms are more and more used as distributed and elastic runtime environments for cloud-native applications [5] and can be understood as a kind of cloud infrastructure unifying middleware [6]. It should be possible to make use of the same features to immunize cloud applications simply by moving an application within the same provider infrastructure. To move anything from A to A makes no sense at first glance. However, let us be paranoid and aware that with some probability and at a given time, an attacker will be successful and compromise at least one virtual machine [7]. A transfer from A to A would be an effective countermeasure – because the intruder immediately loses any hijacked machine that is moved. To understand that, the reader must know that our approach does not effectively move a machine, it regenerates it. To move a machine means to launch a compensating machine unknown to the intruder and to terminate the former (hijacked) machine. Whenever an application is moved its virtual machines are regenerated. Moreover, this would effectively eliminate undetected hijacked machines.

However, attackers can run automated attacks against regenerated machines that will incorporate the same set of vulnerabilities. Therefore, this extended paper shows how we further can improve the regenerating security measure by

TABLE I. **Some popular open source elastic platforms**

Platform	Contributors	URL
Kubernetes	Cloud Native Found.	http://kubernetes.io
Swarm	Docker	https://docker.io
Mesos	Apache	http://mesos.apache.org/
Nomad	Hashicorp	https://nomadproject.io/

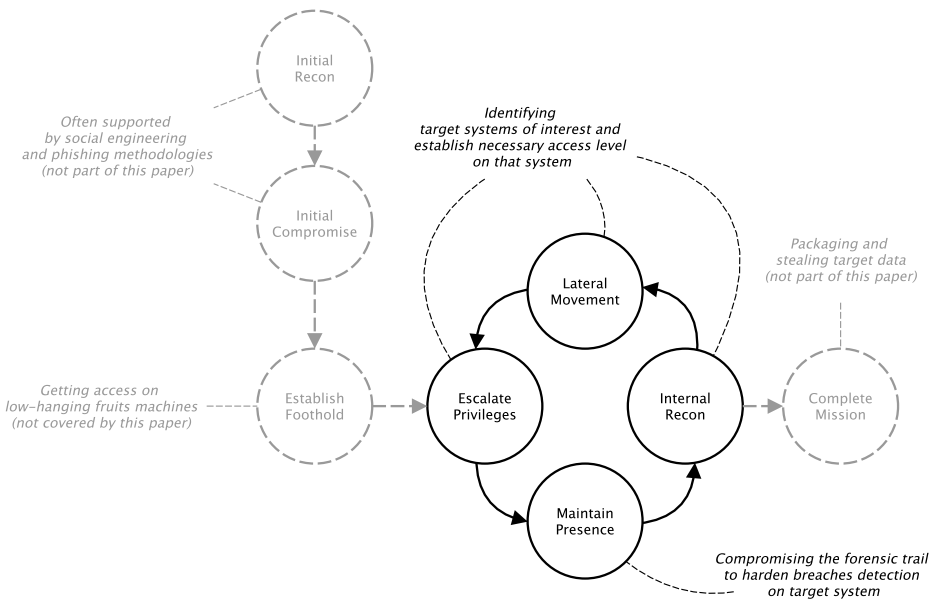

Initial Recon
Often supported by social engineering and phishing methodologies (not part of this paper)
Initial Compromise
Getting access on low-hanging fruits machines (not covered by this paper)
Establish Foothold
Identifying target systems of interest and establish necessary access level on that system
Lateral Movement
Escalate Privileges
Internal Recon
Maintain Presence
Compromising the forensic trail to harden breaches detection on target system
Packaging and stealing target data (not part of this paper)
Complete Mission

Figure 1. **The cyber attack life cycle model.** *Adapted from the cyber attack lifecycle used by the M-Trends reports, see Table II.*

employing MTD at the *application layer* [8] to change the attack surface of the application itself to let even automated and formerly successful attack scripts fail (at least partly). Primarily, this is achieved by diversifying the application in a way that its containerized components are dynamically transformed at *runtime*. The two abstraction layers that compose microservice applications (application layer and the container image layers) are dynamically changed by changing the programming languages of the applications, and consequently, the container images are built to conform to the requirements of the corresponding applications. This combined approach is enforced at runtime to transform the attack surface of cloud-native applications, thereby reducing the possibility of successful attacks.

The remaining of this paper is **outlined** as follows: Section II presents a cyber-attack lifecycle model to show where our approach intends to break the continuous workflow of security breaches. Section III presents an approach on how MTD can be applied on cloud runtime environment (infrastructure) level to regenerate the *"infrastructure cells"* of a system continuously, leveraging the inherent "designed-for-failure" capabilities of modern container platforms like Kubernetes, Swarm, or Mesos. This continuing regeneration will wipe out even undetected attackers in a system. However, attackers might recognize that they periodically loose foothold in a hijacked system and might try to automate their breaches. To overcome this, Section IV will present how even the attack surface of an application can be continuously changed and therefore extends our ideas shown in [1]. We have to consider that our approach has some limitations. We will discuss these limitations in Section V and present corresponding related work in Section VI. We conclude our findings in Section VII.

II. Cyber Attack Reference Model

Figure 1 shows the cyber attack life cycle model, which is used by the M-Trends Report [9] to present current developments in cyber attacks over the years. According to this model, an attacker passes through different stages to complete a cyber attack mission. It starts with initial reconnaissance and compromising of access means. Social engineering methodologies [10] and phishing attacks [11] very often supports these steps. Intruders aim to establish a foothold near the target. All these steps are not covered by this paper, because technical solutions are not able to harden the weakest point in security – the human being. The following steps of this model are more important for this paper. According to the life cycle model, the attacker's goal is to escalate privileges to get access to the target system. Because this leaves trails on the system, which could reveal a security breach, the attacker is motivated to erase this forensic trail. According to security reports, attackers increasingly employ counter-forensic measures to hide their presence and impair investigations. These reports refer to techniques used to clear event logs and securely delete arbitrary files. The technique is simple, but the intruders' knowledge of forensic artifacts demonstrate increased sophistication, as well as their intent to persist in the environment. With a barely detectable foothold, the internal reconnaissance of the victim's network is carried out to allow lateral movement to target systems. This process is a complex and lengthy process and may even take weeks. So, infiltrated machines and application components have worth for attackers and tend to be used for as long as possible. Table II shows the average period an intruder remains on a victim system undetected. So, basically there is the requirement, that **(1) an undetected attacker should lose access to compromised nodes of a system as fast as possible.** Furthermore, there is the requirement that it **(2) must be hard for an attacker to regain foothold in a system by automating successful attacks**. However, how?

Section III will deal with the **(1) requirement** showing that it is possible to regenerate possibly compromised infrastructure continuously even to get rid of undetected attackers. Section

TABLE II. **Undetected days on victim systems** *reported by M-Trends. External and internal discovery data is reported since 2015. No data could be found for 2011.*

Year	External notification	Internal discovery	Median
2010	-	-	416
2011	-	-	?
2012	-	-	243
2013	-	-	229
2014	-	-	205
2015	320	56	146
2016	107	80	99

IV will deal with the **(2) requirement** and demonstrate that it is possible to change attack surfaces of applications in a way that successful attacks cannot be repeated 1:1.

III. Moving Target Defense Mechanisms on the Container Runtime Environment Level

Our recent research dealt [12] mainly with vendor lock-in and the question how to design cloud-native applications that are transferable between different cloud service providers. One aspect that can be learned from this is that there is no common understanding of what a cloud-native application is. A kind of software that is *"intentionally designed for the cloud"* is an often heard but empty phrase. However, noteworthy similarities exist between various viewpoints on *cloud-native applications* (CNA) [5]. A common approach is to define maturity levels in order to categorize different kinds of cloud applications (see Table III). [13] proposed the IDEAL model for CNAs. A CNA should strive for an **isolated state**, is **distributed**, provides **elasticity** in a horizontal scaling way, and should be operated on **automated deployment machinery**. Finally, its components should be **loosely coupled**.

Balalaie et.al [14] stressed that these properties are addressed by cloud-specific architecture and infrastructure approaches like **Microservices** [15], **API-based collaboration**, adaption of **cloud-focused patterns** [13], and **self-service elastic platforms** that are used to deploy and operate these microservices via self-contained deployment units (containers). Table I lists some of these platforms that provide additional operational capabilities on top of IaaS infrastructures like automated and on-demand scaling of application instances, application health management, dynamic routing and load balancing as well as aggregation of logs and metrics [5].

A. Regenerating cloud application runtime environments continuously

If the reader understands and accepts the commonality that cloud-native applications are operated (more and more often) on elastic – often container-based – platforms, it is an obvious idea to delegate the responsibility to immunize cloud applications to these platforms. Recent research showed that the operation of these elastic container platforms and the design of applications running on top of them should be handled as two different engineering problems. This point of view often solves several issues in modern cloud-native application engineering [4]. Also, that is not just true for the transferability problem but might be an option to tackle zero-day exploits. These kinds of platforms could be an essential part of the immune system of modern cloud-native applications.

Furthermore, **self-service elastic platforms** are really "bulletproofed" [17]. *Apache Mesos* [18] has been successfully operated for years by companies like Twitter or Netflix to consolidate hundreds of thousands of compute nodes. Elastic container platforms are **designed for failure** and provide self-healing capabilities via auto-placement, auto-restart, auto-replication, and auto-scaling features. They will identify lost containers (for whatever reasons, e.g., process failure or node unavailability) and will restart containers and place them on remaining nodes. These features are necessary to operate large-scale distributed systems resiliently. However, the same features can be used intentionally to **purge "compromised nodes"**.

In [3], a software prototype that provides the control process shown in Figure 2 and Figure 3 was presented. This process relies on an *intended state* ρ and a *current state* σ of a container cluster. If the intended state differs from the current state ($\rho \neq \sigma$), necessary adaption actions are deduced (creation and attachment/detachment of nodes, creation and termination of security groups) and processed by an execution pipeline fully automatically (see Figure 3) to reach the *intended state* ρ. With this kind of control process, a cluster can be simply resized by changing the intended amount of nodes in the cluster. If the cluster is shrinking and nodes have to be terminated, affected containers of running applications will be rescheduled to other available nodes.

The downside of this approach is that this will only work for Level 2 (cloud resilient) or Level 3 (cloud-native) applications, (see Table III) which by design, can tolerate dependent service failures (due to node failures and container rescheduling). However, for that kind of Level 2 or Level 3 application, we can use the same control process to regenerate nodes of the container cluster. The reader shall consider a cluster with $\sigma = N$ nodes. If we want to regenerate one node, we change the intended state to $\rho = N + 1$ nodes, which will add one new node to the cluster ($\sigma' = N + 1$). Moreover, in a second step, we will decrease the predetermined size of the cluster to $\rho' = N$ again, which affects that one node of the cluster is terminated ($\sigma'' = N$). So, a node is regenerated simply by adding one node and deleting one node. We could even regenerate the complete cluster by changing the cluster size in the following way: $\sigma = N \mapsto \sigma' = 2N \mapsto \sigma'' = N$. However, this would consume much more resources because the cluster would double its size for a limited amount of time.

TABLE III. **Cloud Application Maturity Model**, adapted from *OPEN DATA CENTER ALLIANCE Best Practices [16]*

Level	Maturity	Criteria
3	Cloud native	- Transferable across infrastructure providers at runtime and without interruption of service. - Automatically scale out/in based on stimuli.
2	Cloud resilient	- State is isolated in a minimum of services. - Unaffected by dependent service failures. - Infrastructure agnostic.
1	Cloud friendly	- Composed of loosely coupled services. - Services are discoverable by name. - Components are designed to cloud patterns. - Compute and storage are separated.
0	Cloud ready	- Operated on virtualized infrastructure. - Instantiateable from image or script.

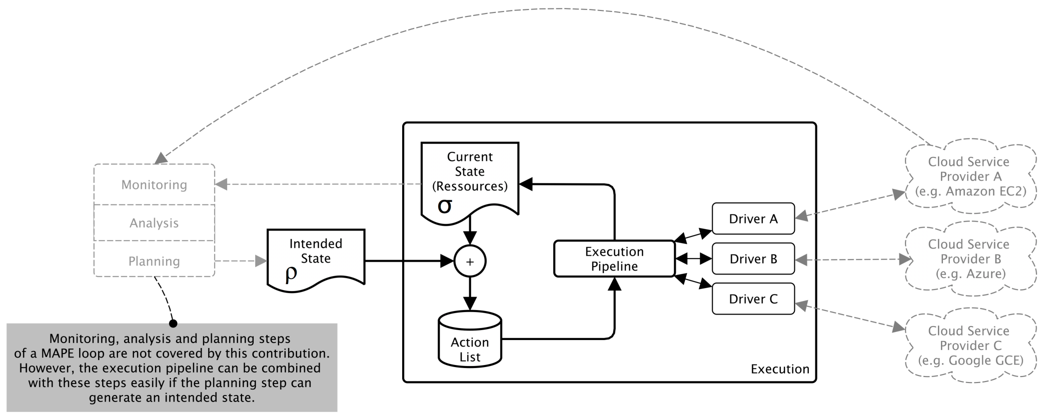

Monitoring
Analysis
Planning
Intended State ρ
Current State (Ressources) σ
+
Action List
Execution Pipeline
Driver A
Driver B
Driver C
Execution
Cloud Service Provider A (e.g. Amazon EC2)
Cloud Service Provider B (e.g. Azure)
Cloud Service Provider C (e.g. Google GCE)
Monitoring, analysis and planning steps of a MAPE loop are not covered by this contribution. However, the execution pipeline can be combined with these steps easily if the planning step can generate an intended state.

Figure 2. **The control theory inspired execution control loop** *compares the intended state ρ of an elastic container platform with the current state σ and derives necessary scaling actions. These actions are processed by the execution pipeline explained in Figure 3. So, platforms can be operated elastically in a set of synchronized IaaS infrastructures. Explained in details by [3].*

A more resource efficient way would be to regenerate the cluster in N steps: $\sigma = N \mapsto \sigma' = N+1 \mapsto \sigma'' = N \mapsto ... \mapsto \sigma^{2N-1} = N+1 \mapsto \sigma^{2N} = N$. The reader is referred to [4] for more details, especially if the reader is interested in the multi-cloud capabilities that are not covered by this paper due to page limitations.

Whenever such regeneration is triggered, all – even undetected – hijacked machines would be terminated and replaced by other machines, but the applications would be unaffected. For an attacker, this means losing their foothold in the system entirely. Imagine this would be done once a day or even more frequently?

B. Evaluation

The execution pipeline presented in Figure 3 was evaluated by operating and transferring two elastic platforms (*Swarm Mode of Docker 17.06* and *Kubernetes 1.7*). The platforms operated a reference "sock-shop" application being one of the most complete reference applications for microservices architecture research [19]. Table IV lists the machine types that show a high similarity across different providers [20].

The evaluation of [4] demonstrated that most time is spent on the IaaS level (creation and termination of nodes and security groups) and not on the elastic platform level (joining, draining nodes). The measured differences on infrastructures provided by different providers are shown in Figure 4. For the current use case, the reader can ignore the times to create and delete a security group (because that is a one time action). However, there will be many node creations and terminations. According to our execution pipeline shown in Figure 3, a node creation ($\sigma = N \mapsto \sigma' = N+1$) involves the durations to **create a node** (request of the virtual machine including all installation and configuration steps), to **adjust security groups** the cluster is operated in and to **join the new node** into the cluster. The shutdown of a node ($\sigma = N \mapsto \sigma' = N-1$) involves the **termination of the node** (this includes the platform draining and deregistering of the node and the request to terminate the virtual machine) and the necessary **adjustment of the security group**. So, for a complete regeneration of a node ($\sigma = N \mapsto \sigma' = N+1 \mapsto \sigma'' = N$) we have to add these runtimes. Table V lists these values per infrastructure.

Even on the "slowest" infrastructure, a node can be regenerated in about 10 minutes. In other words, one can regenerate six nodes every hour or up to 144 nodes a day or a cluster of 432 nodes every 72h (which is the reporting time requested by the EU General Data Protection Regulation). If the reader compares a 72h regeneration time of a more than 400 node cluster (most systems are not so large) with the median value of 99 days that attackers were present on a victim system in 2016 (see Table II) the benefit of the proposed approach should become apparent.

TABLE IV. **Used machine types and regions for evaluation**

Provider	Region	Master type	Worker type
AWS	eu-west-1	m4.xlarge	m4.large
GCE	europe-west1	n1-standard-4	n1-standard-2
Azure	europewest	Standard_A3	Standard_A2
OS	*own datacenter*	m1.large	m1.medium

TABLE V. **Durations to regenerate a node (median values)**

Provider	Creation	Secgroup	Joining	Term.	Total
AWS	70 s	1 s	7 s	2 s	**81 s**
GCE	100 s	8 s	9 s	50 s	**175 s**
Azure	380 s	17 s	7 s	180 s	**600 s**
OS	110 s	2 s	7 s	5 s	**126 s**

IV. Moving Target Defense Mechanisms on the Microservice Architecture Level

MTD techniques introduce methods for improving the security of protected assets by applying *security-by-diversity* tactics and *security diversification* concepts. While most MTD techniques do not have formal requirements for diversifying, i.e., when, how and why to diversify, we employ a *cyber risk-based* technique as the primary diversification decision making factor on the application level [8]. Our motivation for this is to overcome the high number of vulnerability infection among container images as shown by several recent researchers[21], [22]. Therefore, our MTD techniques are designed to improve this *state of insecurity* by reducing the *window of vulnerability*

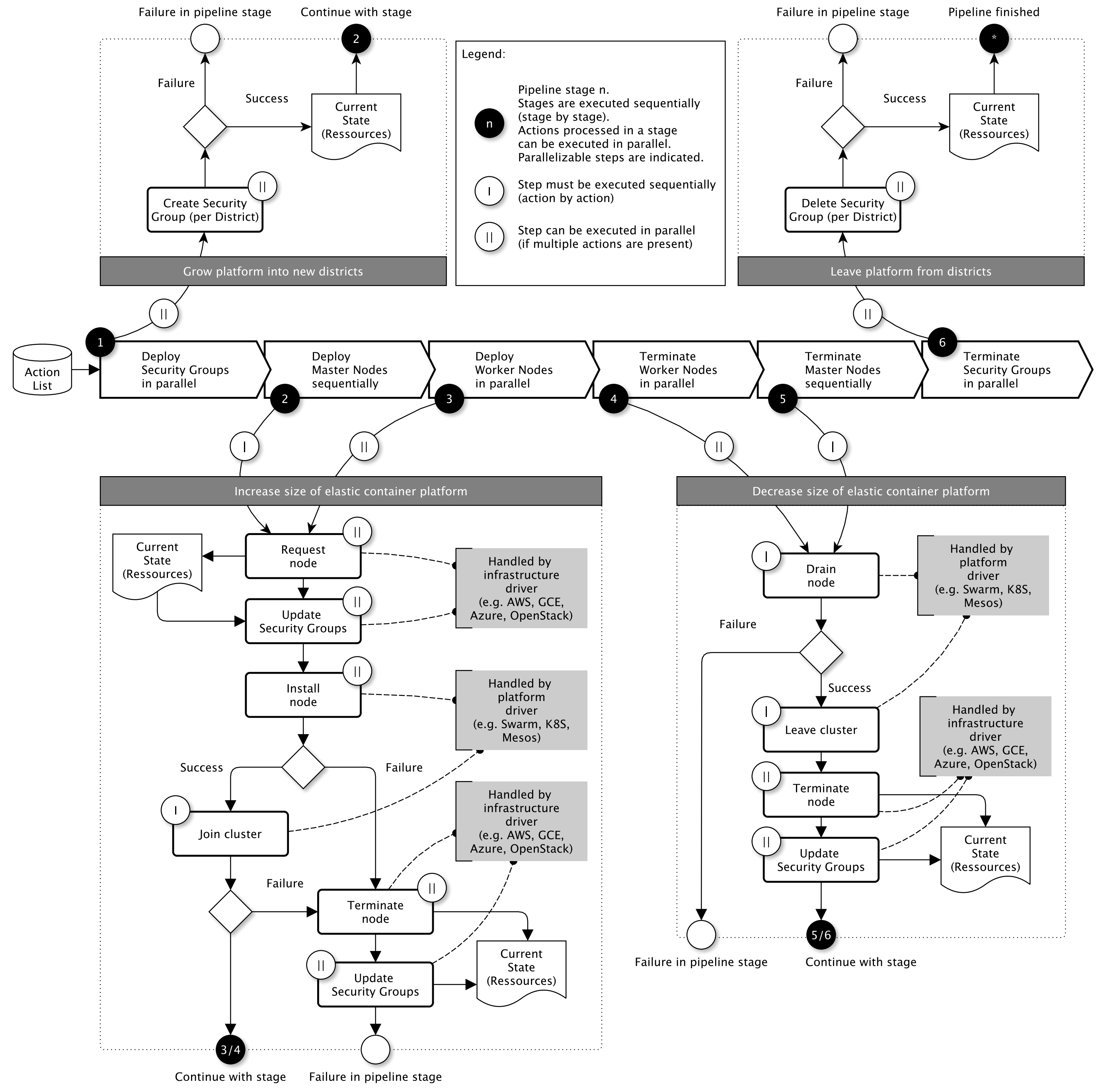

Failure in pipeline stage
Continue with stage
2
Failure
Success
Current State (Ressources)
Create Security Group (per District)
||
Grow platform into new districts
Legend:
n
Pipeline stage n. Stages are executed sequentially (stage by stage). Actions processed in a stage can be executed in parallel. Parallelizable steps are indicated.
|
Step must be executed sequentially (action by action)
||
Step can be executed in parallel (if multiple actions are present)
Failure in pipeline stage
Pipeline finished
*
Failure
Success
Current State (Ressources)
Delete Security Group (per District)
||
Leave platform from districts
Action List
1
Deploy Security Groups in parallel
2
Deploy Master Nodes sequentially
3
Deploy Worker Nodes in parallel
4
Terminate Worker Nodes in parallel
5
Terminate Master Nodes sequentially
6
Terminate Security Groups in parallel
Increase size of elastic container platform
Current State (Ressources)
Request node
Update Security Groups
Install node
Handled by infrastructure driver (e.g. AWS, GCE, Azure, OpenStack)
Handled by platform driver (e.g. Swarm, K8S, Mesos)
Success
Failure
Join cluster
Failure
Terminate node
Update Security Groups
Handled by infrastructure driver (e.g. AWS, GCE, Azure, OpenStack)
Current State (Ressources)
3/4
Continue with stage
Failure in pipeline stage
Decrease size of elastic container platform
Drain node
Handled by platform driver (e.g. Swarm, K8S, Mesos)
Failure
Success
Leave cluster
Handled by infrastructure driver (e.g. AWS, GCE, Azure, OpenStack)
Terminate node
Update Security Groups
Current State (Ressources)
5/6
Failure in pipeline stage
Continue with stage

Figure 3. **The execution pipeline** *processes necessary actions to transfer the current state σ into the intended state ρ. See [4] for more details.*

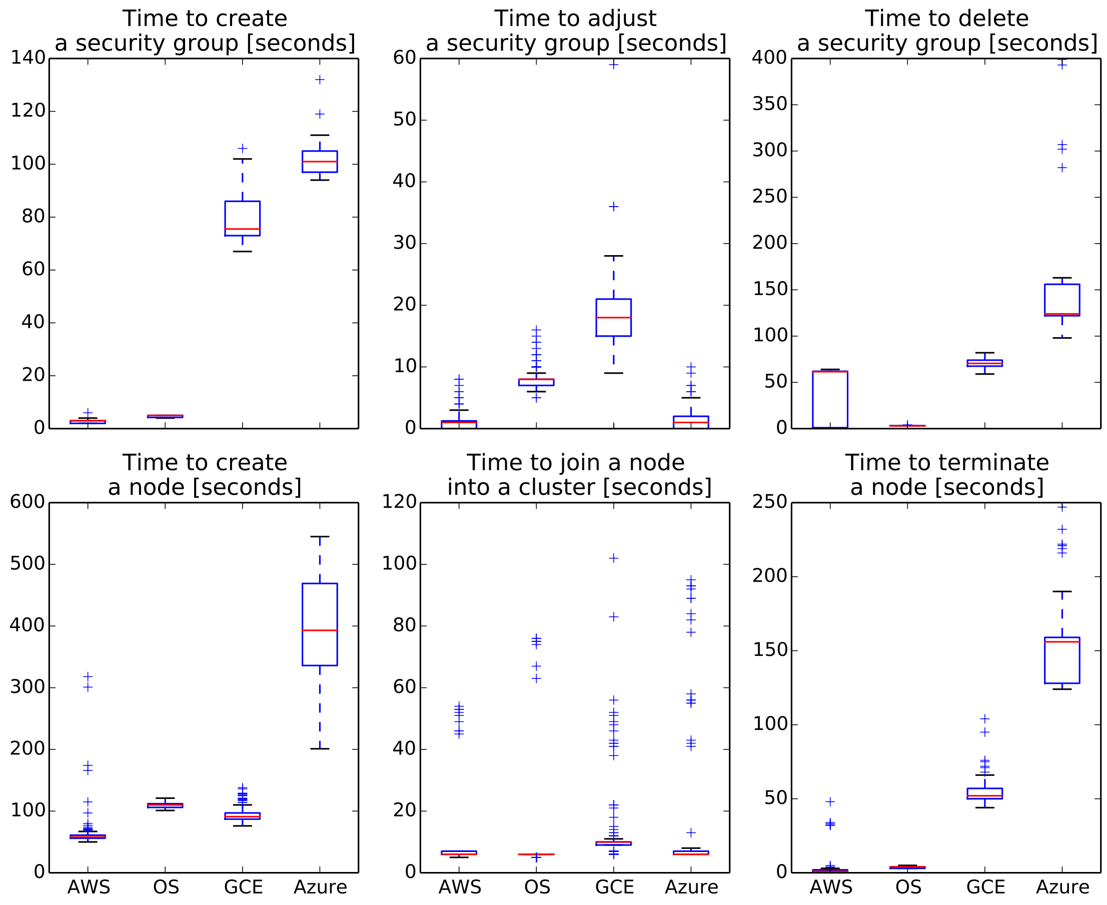


Figure 4. **Infrastructure specific runtimes of IaaS operations** *see [4].*

exposure via diversification and commensurate attack surface randomization.

A. Cyber Risk Analysis for Microservice Diversification

Larsen et al. [23] assert that a common challenge when employing diversification strategies is deciding on *when*, *how* and *where* to diversify. We present a cyber risk procedure to support decision making or satisfy the afore-mentioned requirements. We leverage security metrics to design a cyber risk-based mechanism, and security metrics are useful tools for risk assessment. These metrics are computed by deriving security risks per microservice and after that employing *vulnerability prioritization* such that diversification is a function of microservice risk assessment, i.e., microservices are diversified in order risk severity. We introduce the notion of **Diversification Index -** D_i as an expression of the depth of diversification to be implemented. D_i defines if microservices are to be globally or selectively diversified e.g., diversifying 2 out of 4 microservices can be expressed as *2:4*. We formally define D_i as:

$$D_i = \frac{m_d}{m} \tag{1}$$

where,

m_d = number of microservices to be diversified,

m = total number of microservices in the application.

In order to decide on the value of D_i, it is important to analyze the security state of the microservice application. There are several techniques for arriving at what may be defined as the *security state* of a microservice application. Traditionally, several security metrics have been defined to evaluate the security state of computing environments, therefore we employ security risk techniques, following two approaches:

1) Risk Analysis Using CVSS: The Common Vulnerability Scoring System CVSS [24] is a widely adopted vulnerability metrics standard. It provides vulnerability *base scores*, which express the severity of damage the referred vulnerability might impact upon a system if exploited. In order to derive the microservice security state (Security Risk - SR), base scores of all the vulnerabilities detected can be summed and averaged as expressed below:

$$SR = \frac{1}{N}\sum_{i=1}^{N} V_i \tag{2}$$

where SR is the Security Risk, V_i is the CVSS base score of vulnerability i, and N is the total number of vulnerabilities detected in microservice m. However, averaging vulnerabilities to obtain a single metric to signify a system's security state is not optimal. Derived values are not sufficiently representative of other factors such as the public availability of exploits. Therefore, we employ another scoring technique called *shrink-*

age estimator, an approach, which has been popularly used for online rating systems, e.g., IMDB. The shrinkage estimator considers the average rating and the number of votes. Hence, it provides a more precise value for SR, than mere averaging (Equation (2)). Therefore, leveraging the shrinkage estimator, we can derive a more precise SR as follows:

$$SR = \frac{v}{v+a}R + \frac{a}{v+a}C \qquad (3)$$

where,

v = the total number of vulnerabilities detected in a microservices,

a = minimum number of vulnerabilities to be detected in a microservice assessment before it added in the risk analysis,

C = the mean severity score of vulnerabilities detected in a microservice

R = the average severity score of all vulnerabilities infecting a microservice-based application

The Pearson's correlation coefficient is derived to determine the dependence relationship between the microservices.

2) Risk Analysis Using OWASP Risk Rating Methodology: The risk assessment method described in the previous subsection is limited to vulnerabilities contained in the Common Vulnerability Enumeration (CVE) dictionary. CVE is a public dictionary for publishing known vulnerabilities. These vulnerabilities are analyzed and assigned vulnerability security metrics using the CVSS. However, the CVE contains only a handful of web application vulnerabilities. Thus, we need to derive another risk assessment methodology for application layer vulnerabilities. This additional step is necessary since microservices are essentially web/REST-based applications. We opt for the OWASP Risk Rating Methodology (ORRM), which is specifically designed for web applications [25]. This methodology is based on two core risk components: *Likelihood* and *Impact* formally expressed as:

$$Risk = Likelihood * Impact \qquad (4)$$

In order to derive these metrics, risk assessors are required to consider the threat vector, attacks to be used and the impacts of successful attacks.

B. Dissecting Microservice Attack Surfaces

An important aspect of our *security-by-diversity* tactics is to manipulate microservice attack surfaces against possible attackers through random architectural transformations. Therefore, the attack surfaces are altered by randomizing the entry and exit points, which are commonly used for identifying attack surfaces [27], [28]. A detailed understanding of these attack surfaces is imperative. Therefore, we categorize microservice attack surface into: *horizontal* and *vertical* attack surfaces and thereafter employ *vulnerability correlation* to identify vulnerability similarities.

1) Horizontal Vulnerability Correlation: The objective of correlating vulnerabilities horizontally is to analyze the relationship of vulnerabilities along the horizontal attack surface, i.e., the parts of the applications users directly interact with. Figure 5 illustrates the multi-layered attack surface of the PetClinic application [26]. The application layer horizontal attack surface consists of the interactions and exit/entry points

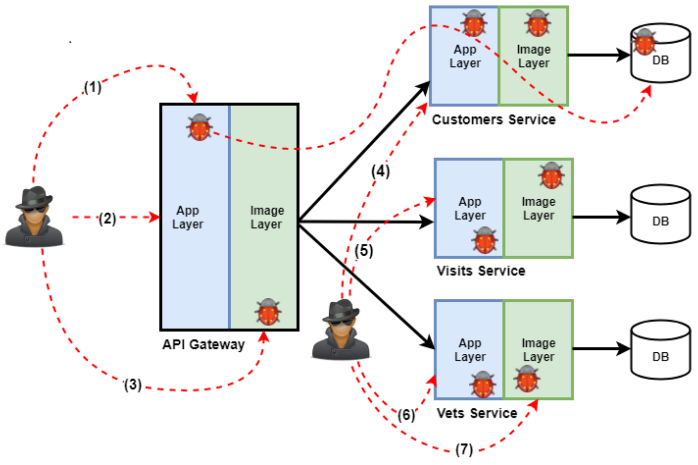


Figure 5. Typical Microservice Attack Surfaces illustrated with the PetClinic Application [26]

from the API gateway to the Vets, Visits and Customer services application layers. Requests and responses are transversed along this layer, providing attack opportunities for attackers. The vulnerability correlation process is similar to *security event correlation* techniques [29], though rather than clustering similar attributes, e.g., malicious IP addresses, we focus on Common Weakness Enumeration (CWE) Ids. The CWE is a standardized classification system for application weaknesses [30]. For example, CWE 89 categorizes all vulnerabilities related to *Improper Neutralization of Special Elements used in an SQL Command (SQL Injection)* [31] and can be mapped to several CVEs, e.g., *CVE-2016-6652*[32], a SQL injection vulnerability in Spring Data JPA. If this vulnerability exists in all PetClinic's microservices, an attacker could easily conduct a correlated attack (*Attack Paths 2, 4, 5, and 6* of Figure 5) resulting to correlated failures and eventual application failure since each microservice works ultimately to the successful functioning of the PetClinic application.

2) Vertical Vulnerability Correlation: The vertical correlation technique is similar to the horizontal correlation. However, the interactions across application-image layers are analyzed. This analysis, therefore, employs security-by-design tactics across the vertical attack surface. Attack Path 1 illustrates the exploitation of vulnerability across the vertical attack surface, and the attacker initiated an attack against the API Gateway of the PetClinic application, from the application layer to the image layer. From there, another attack is launched to the Customers service application layer, across the image layer and finally, the database is compromised. The same attack can be repeated against the other microservices if affected by the vulnerabilities. Hence we need to express such casual relationships in vulnerability correlation matrices.

Correlated vulnerabilities can be represented with correlation matrices, more specifically referred to as *microservices vulnerability correlation matrix*. Therefore, we are influenced by [33] to define the microservices vulnerability correlation matrix as *a mapping of vulnerabilities to microservice instances in a microservice-based application*. The *microservices vulnerability correlation matrix* presents a view of vulnerabilities that concurrently affect multiple microservices. An

$$\begin{array}{c|cccc} & V_1 & V_2 & \dots & V_n \\ \hline M_1 & 1 & 1 & \cdots & \dots \\ M_2 & 1 & 0 & \cdots & \dots \\ & \vdots & \vdots & \ddots & \vdots \\ M_n & 1 & 1 & \cdots & \dots \end{array}$$

Figure 6. Microservice Vulnerability Correlation Matrix

TABLE VI. Vulnerabilities Detected in PetClinic App-Layer

CWE-ID	API-GATEWAY	CUSTOMERS-SERVICE	VETS-SERVICE	VISITS-SERVICE
CWE-16	31	4	2	2
CWE-524	48	17	6	11
CWE-79	0	3	0	1
CWE-425	0	0	20	0
CWE-200	14	6	0	0
CWE-22	0	1	0	0
CWE-933	1	0	0	0
TOTAL	94	31	28	14

example of the microservice correlation matrix is Figure 6, where the microservices M_1 and M_2 will have a correlated failure under an attack that exploits vulnerability V_1 since they share the same vulnerability. However, an attack that exploits V_2 can only affect M_1, while M_2 remains unaffected.

C. Evaluation

The PetClinic application was used for our evaluation. PetClinic is a Spring application that demonstrates cloud-native application capabilities. It has been used for demonstration purposes in several industry and academic scenarios [19]. However, we were forced to modify the original PetClinic by adding OpenAPI support. Two experiments have been conducted: (1) Security risk comparison to verify the efficiency of our security-by-diversity tactics (2) Attack surface analysis to evaluate the improvement in the horizontal and vertical attack surfaces.

In order to perform Security Risk analysis, we leveraged the Cloud Aware Vulnerability Assessment System (CAVAS) [34]. The vulnerability scanners integrated into CAVAS (Anchore and OWASP ZAP), are used for launching vulnerability scans against PetClinic images and microservice instances respectively. The detected vulnerabilities were persisted in the Security Reports and CMDB. First, the diversification index

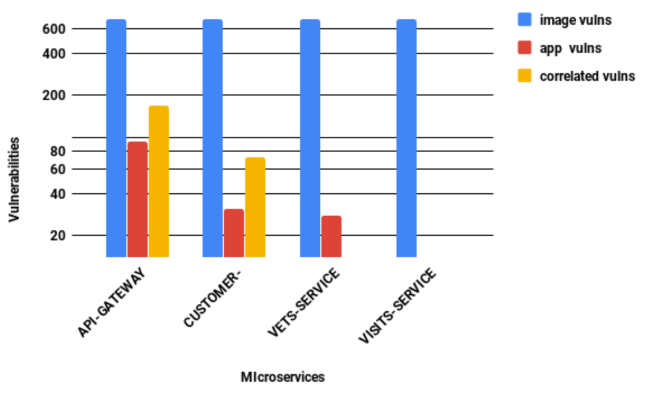


Figure 7. Homogeneous PetClinic Vulnerabilities

TABLE VII. Risk Scores By CWE

CWE-ID	OWASP T10 Risk Category	Risk Score
CWE-16	A6 - Security Misconfiguration	6.0
CWE-524	Not Listed	3.0
CWE-79	A6 - Security Misconfiguration	6.0
CWE-425	Not Listed	3.0
CWE-200	A3 - Sensitive Data Exposure	7.0
CWE-22	A5 - Broken Access Control	6.0
CWE-933	Not Listed	3.0

is derived by computing risks per PetClinic microservices to obtain the *Security Risk* - SR. Hence, we inspect the results for the image vulnerability scan and notice that the vulnerabilities are too similar (Figure 7). Therefore, SR will be too similar for meaningful vulnerability prioritization. Since the prioritization step is imperative for ranking microservices in order of risk severity, we compute SR using the ORRM (Section IV-A2). The application layer scan results are retrieved from the database and analyzed. Scores are assigned to the detected vulnerabilities based on the risk scores for OWASP Top-10 2017 web vulnerabilities. This is a reasonable approach given OWASP uses ORRM for deriving the Top-10 web application vulnerability scores. Also, this affords objective assignment of scores [35], which are publicly verifiable. Table VI is the distribution of detected vulnerabilities, while a subset of the mapping between CWE-Ids and OWASP Top-10 is on Table VII. From Table VII, it is obvious that the API-Gateway has the most severe risks followed by the Customer, Vets, and Visits microservices. Therefore, we apply diversification based on this result using a *diversification index of 3:4*, i.e., three out of four microservices. The diversified PetClinic is *retested* and the results are shown in Figure 8. We observe that the diversified PetClinic application layer vulnerabilities are reduced with about 53.3 %. However, the image vulnerabilities increased especially for the Customer and Vets service, which are transformed to NodeJS and Ruby respectively. Importantly, the microservices are no longer homogeneous, and the possibilities for correlated attacks have been eliminated. Also, the vulnerabilities in the API Gateway's image are drastically reduced from 696 to 6, while the application layer vulnerabilities reduced from 94 to 24. The reduction is due to reduced code base size, a distinct characteristic of Python programming model. The API Gateway is the most important microservice since it presents the most vulnerable and sensitive attack surface of the application, therefore consider the security of PetClinic improved, our results mean that out of 94 opportunities for attacking the API Gateway, only 24 were left.

D. Attack Surface Analysis

Here we analyze the attack surfaces of the homogeneous and diversified PetClinic versions. We consider direct and indirect attack surfaces, i.e., vulnerabilities that directly/ indirectly lead to attacks respectively. From the vulnerability scan reports, each detected vulnerability is counted as an attack surface *unit* (*attack opportunities concept* [36], [37]). Figure 9 compares the horizontal app layer attack surface for both PetClinic apps. Notice a reduced attack surface in the diversified version, showing better security. Essentially, the attackability of PetClinic has been reduced. However, the results for the vertical attack surface are different. This attack surface portrays attacks transversing the app-image layer

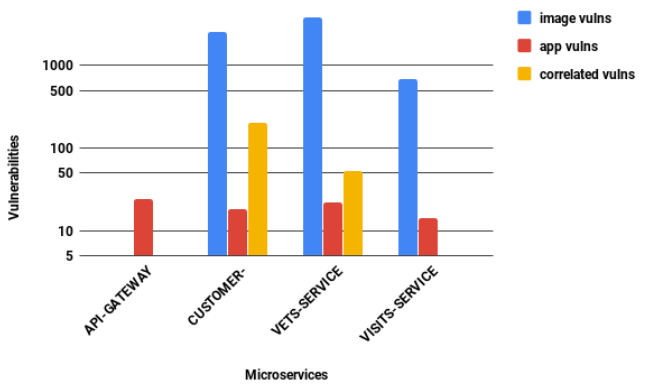


Figure 8. Diversified PetClinic Vulnerabilities

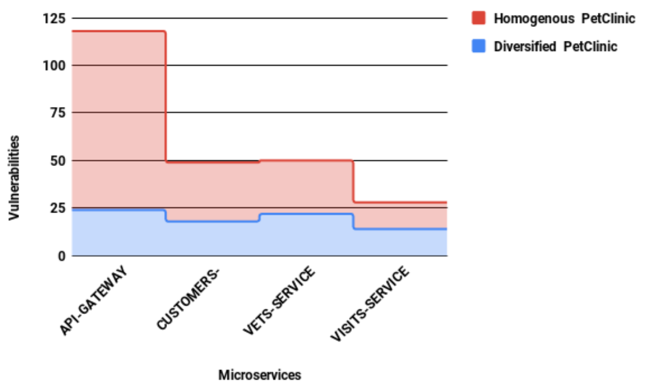


Figure 9. Horizontal Attack Surface Analysis

(Figure 5). While there are fewer correlated vulnerabilities in the diversified API-Gateway, correlated vulnerabilities in the Customers and Vets Services have increased. This increment is due to the corresponding increase of image vulnerabilities. However, the attackability due to homogeneity is reduced. We want to emphasize that intruders would observe this approach as permanently changing attack surfaces increasing dramatically the effort to breach the system.

V. Critical Discussion

The idea presented in Section III of an immune system like approach to remove undetected intruders in virtual machines seems to a lot of experts intriguing. Nevertheless, according to the state of the art, this is currently not done. There might be reasons for that and open questions the reader should consider.

It is often remarked that the proposal can be compared with the approach to restart periodically virtual machines that have memory leak issues and has apparently nothing to do with security concerns, and could be applied to traditional (non-cloud) systems as well. So, the approach may have even a broader focus than presented (which is not a bad thing).

Another question is how to detect "infected" nodes? The presented approach selects nodes simply at random and will hit every node at some time. The same could be done using a round-robin approach, but a round-robin strategy would be better predictable for an attacker. However, both strategies will create a lot of additional regenerations, and that leaves room for improvements. It seems obvious to search for solutions like presented by [38], [39] to provide some "intelligence" for the identification of "suspicious" nodes. Such a kind of intelligence would limit regenerations to likely "infected" nodes. In all cases, it is essential for anomaly detection approaches to secure the forensic trail [40], [41].

Furthermore, to regenerate nodes periodically or even randomly is likely nontrivial in practice and depends on the state management requirements for the affected nodes. Therefore, this paper proposes the approach only as a promising solution for Level 2 or 3 cloud applications (see Table III) that are operated on elastic container platforms. These kinds of applications have desirable state management characteristics. However, this is a limitation to applications following the microservice architecture approach.

One could be further concerned about exploits that are adaptable to bio-inspired systems. Stealthy resident worms dating back to the old PC era would be an example. This concern might be especially valid for the often encountered case of not entirely stateless services when data-as-code dependencies or code-injection vulnerabilities exist. Furthermore, attackers could shift their focus to the platform itself in order to disable the regeneration mechanism as a first step. On the other hand, this could be easily detected – but there could exist more sophisticated attacks. In order to efficiently employ this strategy, efficient *real-time* security monitoring is required. This could be achieved via two major approaches, the first requires log aggregation and analysis using either machine learning practices or other anomaly detection techniques. Otherwise, it is also possible to deploy run-time security monitoring agents in the cloud as recommended by the NIST Application Container Security Guide [42]. For example, Falco [43] is an open-source behavioral activity monitor for detecting anomalous activities in containers. When deployed, it can trigger the regeneration of new cells when malicious activities are detected, however the system has to automate the management of traffic (requests and responses) in a manner that evades service disruption.

The "immunization" results on the infrastructure level (see Section III) are impressive but should be combined with secure coding practices in development pipelines, i.e., coordinated with continuous security assessments. We presented how to automate security in CNA development environments [34]. In these cases, detected web vulnerabilities, e.g., *X-Content-Type-Options Header Missing*, can be resolved by appending appropriate *headers*, as described and advised in CAVAS reports. Furthermore, image vulnerabilities can be reduced by using more secure container images. For example, *Alpine Linux* images can replace Ubuntu images as base images due to smaller footprint, which equals smaller attack surfaces [44].

Of course, this is in line with the current trend of moving security of microservices *leftwards*, i.e., integrating security in the development pipelines [45]. Similarly, it is imperative to implement continuous security monitoring techniques for deployed containers to detect vulnerabilities discovered after the deployment of containers in production environments. This approach provides an efficient possibility of patching vulnerable images and redeploying the appropriate containers. However, it adds an overhead for the development pipelines since it requires additional deployment, management and involvement of security teams/architects.

Our MTD approach presented in Section IV leverages automatic code generation techniques on the application level via Swagger CodeGen library. We discovered that over 150 companies/projects use Swagger CodeGen in production [46], hence the library is mature and capable of transforming large microservice applications. Nevertheless, in this work a basic application has been used to introduce the concepts, more complex applications will be tested in the future. There are however several obstacles to the realization of the proposed MTD, our techniques can be applied only to OpenAPI compatible microservices. This implies that the target microservices have to be compatible with the OpenAPI standard, this is achievable via several programming frameworks, and also a positive addition since OpenAPI seems to be the leading API exchange standard. Also, Swagger Codegen currently supports about 30 programming languages/frameworks and this might be a limitation in terms for possible combinations (entropy), although more languages can be added via customization. There might be a need for manual efforts to check if the transformation output is functionally compatible especially for complex applications. A possible challenge for real deployments might be the need to have development teams that are proficient in multiple programming languages.

An event-based technique might interestingly enhance our MTD technique by detecting attacks and triggering commensurate diversification. Conventionally, Web Application Firewalls (WAF) are deployed in front of web applications to detect and stop malicious traffic (which might also indicate an ongoing attack). Hence WAF can be deployed at the API Gateway and configured with *attack thresholds*. Once a threshold is breached, the WAF would trigger the diversification of the entire microservice application or only the endangered microservices. A scheduled diversification routine might support this methodology. These techniques can comfortably be applied across cloud platforms using orchestration technologies, e.g., Kubernetes.

VI. Related Work

To the best of the authors' knowledge, there are currently no approaches making intentional use of virtual machine regeneration for security purposes neither on the infrastructure nor on the application level. However, the proposed approach is derived from multi-cloud scenarios and their increased requirements on security. Moreover, several promising approaches are dealing with multi-cloud scenarios. So, all of them could show equal opportunities. However, often, these approaches come along with much inherent complexity. A container-based approach seems to handle this kind of complexity better. There are some good survey papers on this [47], [48], [49], [50].

MTD via software diversity was first introduced by Forest et al. [51], since then the concept has been applied at different abstraction levels. Baudry et al. [52] introduced *sosiefication*, a diversification method, which transforms software programs by generating corresponding replicas through statement deletion, addition or replacement operators. These variants still exhibit the same functionality but are computationally diverse. Williams et al. [53] presented *Genesis*, a VM-based dynamic diversification system. Genesis employed the *Strata* VM to distribute software components such that every version became unique, hence difficult to attack. A detailed comparison of automated diversification techniques was presented in [23]. The authors have not found a prior work that applied MTD concepts to microservices.

VII. Conclusion

There is still no such thing as an impenetrable system. Once attackers successfully breach a system, there is little to prevent them from doing arbitrary harm but we can reduce the available time for the intruder to do this. Moreover, we can make it harder to replay a successful attack. The presented approach evolved mainly from transferability research questions for cloud-native applications. Therefore, it is limited to microservice-based application architectures but provides some unusual characteristics for thinking about security in general.

Basically we proposed an "immune system" inspired approach to tackle zero-day exploits. The founding cells are continuously regenerated. The primary intent is to reduce the time for an attacker acting undetected massively. Therefore, this paper proposed to regenerate virtual machines (the cells of an IT-system) with a much higher frequency than usual to purge even undetected intruders. Evaluations on infrastructures provided by AWS, GCE, Azure, and OpenStack showed that a virtual machine could be regenerated between two minutes (AWS) and 10 minutes (Azure). The reader should compare these times with recent cybersecurity reports. In 2016 an attacker was undetected on a victim system for about 100 days. The presented approach means for intruders that their undetected time on victim systems is not measured in months or days any-more, it would be measured in minutes.

However, regenerated virtual machines will incorporate the same set of application vulnerabilities. So, a reasonable approach for intruders would be to script their attacks and rerun it merely. Although they might lose their foothold within minutes in a system, they can regain it automatically within seconds. Therefore, we propose to alter the attack surface of applications by randomizing the entry and exit points, which are commonly used for identifying attack surfaces [27], [28]. Based on horizontal and vertical microservice attack surfaces we demonstrated how to employ a *vulnerability correlation* to identify vulnerability similarities on the application layer and how to adapt the attack surface accordingly. This attack surface modification would let even automated and formerly successful attack scripts fail (at least partly). We propose and demonstrate the feasibility to diversify the application via dynamic transformations of its containerized components at *runtime*. In our presented use cases, we could show that it is possible to change the attack surface of a reference application incorporating over 600 container image vulnerabilities and approximately 80 application vulnerabilities to a surface with no image vulnerabilities and only 24 application vulnerabilities anymore. That is a reduction of almost 98%. What is more, the surface of the application can be changed continuously resulting that scripted attacks fail with each surface change. That is a nightmare from an intruders point of view.

The critical discussion in Section V showed that there is a need for additional evaluation and room for more in-depth research on both levels: continuously infrastructure regeneration and application surface modifying. However, several reviewers remarked independently that the basic idea is so "intriguing", that it should be considered more consequently.

ACKNOWLEDGMENT

This research is partly funded by the Cloud TRANSIT project (13FH021PX4, German Federal Ministry of Education and Research). The authors would like to thank Bob Duncan from the University of Aberdeen for his inspiring thoughts on cloud security challenges.

REFERENCES

[1] N. Kratzke, “About an Immune System Understanding for Cloud-native Applications - Biology Inspired Thoughts to Immunize the Cloud Forensic Trail,” in Proc. of the 9th Int. Conf. on Cloud Computing, GRIDS, and Virtualization (CLOUD COMPUTING 2018, Barcelona, Spain), 2018.

[2] B. Duncan and M. Whittington, “Compliance with standards, assurance and audit: does this equal security?” in Proc. 7th Int. Conf. Secur. Inf. Networks - SIN ’14. Glasgow: ACM, 2014, pp. 77–84, [Accessed 10 February 2019]. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2659651.2659711

[3] N. Kratzke, “Smuggling Multi-Cloud Support into Cloud-native Applications using Elastic Container Platforms,” in Proc. of the 7th Int. Conf. on Cloud Computing and Services Science (CLOSER 2017), 2017.

[4] ——, “About the complexity to transfer cloud applications at runtime and how container platforms can contribute?” in Cloud Computing and Service Sciences: 7th International Conference, CLOSER 2017, Revised Selected Papers, Communications in Computer and Information Science (CCIS). Springer International Publishing, 2018, to be published.

[5] N. Kratzke and P.-C. Quint, “Understanding Cloud-native Applications after 10 Years of Cloud Computing - A Systematic Mapping Study,” Journal of Systems and Software, vol. 126, no. April, 2017.

[6] N. Kratzke and R. Peinl, “ClouNS - a Cloud-Native Application Reference Model for Enterprise Architects,” in 2016 IEEE 20th Int. Enterprise Distributed Object Computing Workshop (EDOCW), September 2016.

[7] L. Bilge and T. Dumitras, “Before we knew it: an empirical study of zero-day attacks in the real world,” in ACM Conference on Computer and Communications Security, 2012.

[8] K. A. Torkura, M. I. Sukmana, and A. V. Kayem, “A cyber risk based moving target defense mechanism for microservice architectures,” in 2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom). IEEE, 2018, pp. 932–939.

[9] FireEye, “M-trends 2019 report,” [Accessed 07 November 2017]. [Online]. Available: https://www.fireeye.com/current-threats/annual-threat-report.html

[10] K. Krombholz, H. Hobel, M. Huber, and E. Weippl, “Advanced social engineering attacks,” Journal of Information Security and Applications, vol. 22, 2015.

[11] S. Gupta, A. Singhal, and A. Kapoor, “A literature survey on social engineering attacks: Phishing attack,” 2016 International Conference on Computing, Communication and Automation (ICCCA), 2016, pp. 537–540.

[12] N. Kratzke and P.-C. Quint, “Technical Report of the Project Cloud-TRANSIT - Transfer Cloud-native Applications at Runtime,” October 2018, technical report.

[13] C. Fehling, F. Leymann, R. Retter, W. Schupeck, and P. Arbitter, Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications. Springer Publishing Company, Incorporated, 2014.

[14] A. Balalaie, A. Heydarnoori, and P. Jamshidi, “Migrating to Cloud-Native Architectures Using Microservices: An Experience Report,” in 1st Int. Workshop on Cloud Adoption and Migration (CloudWay), Taormina, Italy, 2015.

[15] S. Newman, Building Microservices. O’Reilly Media, Incorporated, 2015.

[16] S. Ashtikar, C. Barker, B. Clem, P. Fichadia, V. Krupin, K. Louie, G. Malhotra, D. Nielsen, N. Simpson, and C. Spence, “Open Data Center Alliance Best Practices: Architecting Cloud-Aware Applications Rev. 1.0,” 2014, [Accessed 10 February 2019]. [Online]. Available: https://www.opendatacenteralliance.org/docs/architecting_cloud_aware_applications.pdf

[17] M. Stine, Migrating to Cloud-Native Application Architectures. O’Reilly, 2015.

[18] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. H. Katz, S. Shenker, and I. Stoica, “Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center.” in 8th USENIX Conf. on Networked systems design and implementation (NSDI’11), vol. 11, 2011.

[19] C. M. Aderaldo, N. C. Mendonça, C. Pahl, and P. Jamshidi, “Benchmark requirements for microservices architecture research,” in Proc. of the 1st Int. Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering, ser. ECASE ’17. Piscataway, NJ, USA: IEEE Press, 2017.

[20] N. Kratzke and P.-C. Quint, “About Automatic Benchmarking of IaaS Cloud Service Providers for a World of Container Clusters,” Journal of Cloud Computing Research, vol. 1, no. 1, 2015.

[21] J. Gummaraju, T. Desikan, and Y. Turner, “Over 30% of official images in docker hub contain high priority security vulnerabilities,” BanyanOps, Tech. Rep., 2015.

[22] R. Shu, X. Gu, and W. Enck, “A study of security vulnerabilities on docker hub,” in Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, 2017.

[23] P. Larsen, S. Brunthaler, L. Davi, A.-R. Sadeghi, and M. Franz, “Automated software diversity,” Synthesis Lectures on Information Security, Privacy, & Trust, vol. 10, no. 2, 2015, pp. 1–88.

[24] P. Mell, K. Scarfone, and S. Romanosky, “Common vulnerability scoring system,” IEEE Security & Privacy, 2006.

[25] OWASP, “Owasp risk rating methodology,” [Accessed 08 January 2019]. [Online]. Available: https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology

[26] Pivotal, “Distributed version of spring petclinic built with spring cloud,” [Accessed 08 January 2019]. [Online]. Available: https://github.com/spring-petclinic/spring-petclinic-microservices

[27] A. Younis, Y. K. Malaiya, and I. Ray, “Assessing vulnerability exploitability risk using software properties,” Software Quality Journal, 2016.

[28] P. K. Manadhata, Y. Karabulut, and J. M. Wing, “Report: Measuring the attack surfaces of enterprise software.” ESSoS, vol. 9, 2009, pp. 91–100.

[29] M. Ficco, “Security event correlation approach for cloud computing,” International Journal of High Performance Computing and Networking 1, 2013.

[30] S. Christey, J. Kenderdine, J. Mazella, and B. Miles, “Common weakness enumeration,” Mitre Corporation, 2013.

[31] M. Corporation., “Cwe-89: Improper neutralization of special elements used in an sql command (’sql injection’),” [Accessed 16 May 2019]. [Online]. Available: https://cwe.mitre.org/data/definitions/89.html

[32] NIST, “Cve-2016-6652 details,” [Accessed 16 May 2019]. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2016-6652

[33] P.-Y. Chen, G. Kataria, and R. Krishnan, “Correlated failures, diversification, and information security risk management,” MIS quarterly, 2011, pp. 397–422.

[34] K. A. Torkura, M. I. Sukmana, and C. Meinel, “Cavas: Neutralizing application and container security vulnerabilities in the cloud native era (to appear),” in 14th EAI International Conference on Security and Privacy in Communication Networks. Springer, 2018.

[35] OWASP, “Top 10-2017 details about risk factors,” 2017, [Accessed 07 January 2019]. [Online]. Available: https://www.owasp.org/index.php/Top_10-2017_Details_About_Risk_Factors

[36] M. Howard, J. Pincus, and J. M. Wing, “Measuring relative attack surfaces,” in Computer security in the 21st century. Springer, 2005, pp. 109–137.

[37] OWASP, “Attack surface analysis cheat sheet,” [Accessed 08 January 2019]. [Online]. Available: https://www.owasp.org/index.php/Attack_Surface_Analysis_Cheat_Sheet

[38] Q. Fu, J.-G. Lou, Y. Wang, and J. Li, “Execution Anomaly Detection in Distributed Systems through Unstructured Log Analysis,” in 2009 Ninth IEEE Int. Conf. on Data Mining, 2009.

[39] M. Wurzenberger, F. Skopik, R. Fiedler, and W. Kastner, “Applying High-Performance Bioinformatics Tools for Outlier Detection in Log Data,” in CYBCONF, 2017.

[40] B. Duncan and M. Whittington, “Cloud cyber-security: Empowering the audit trail,” Int. J. Adv. Secur., vol. 9, no. 3 & 4, 2016, pp. 169–183.

[41] ——, “Creating an Immutable Database for Secure Cloud Audit Trail and System Logging,” in Cloud Comput. 2017 8th Int. Conf. Cloud Comput. GRIDs, Virtualization. Athens, Greece: IARIA, ISBN: 978-1-61208-529-6, 2016, pp. 54–59.

[42] M. Souppaya, J. Morello, and K. Scarfone, “Application container security guide,” 2017. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-190

[43] FalcoSecurity, “Falco: Container native runtime security,” [Accessed 08 January 2019]. [Online]. Available: https://github.com/falcosecurity/falco

[44] H. Gantikow, C. Reich, M. Knahl, and N. Clarke, “Providing security in container-based hpc runtime environments,” in International Conference on High Performance Computing. Springer, 2016.

[45] H. Myrbakken and R. Colomo-Palacios, “Devsecops: a multivocal literature review,” in International Conference on Software Process Improvement and Capability Determination. Springer, 2017, pp. 17–29.

[46] SmartBear, “Swagger codegen repository,” [Accessed 08 January 2019]. [Online]. Available: https://github.com/swagger-api/swagger-codegen

[47] A. Barker, B. Varghese, and L. Thai, “Cloud Services Brokerage: A Survey and Research Roadmap,” in 2015 IEEE 8th International Conference on Cloud Computing. IEEE, June 2015.

[48] D. Petcu and A. V. Vasilakos, “Portability in clouds: approaches and research opportunities,” Scalable Computing: Practice and Experience, vol. 15, no. 3, October 2014.

[49] A. N. Toosi, R. N. Calheiros, and R. Buyya, “Interconnected Cloud Computing Environments,” ACM Computing Surveys, vol. 47, no. 1, May 2014.

[50] N. Grozev and R. Buyya, “Inter-Cloud architectures and application brokering: taxonomy and survey,” Software: Practice and Experience, vol. 44, no. 3, March 2014.

[51] S. Forrest, A. Somayaji, and D. H. Ackley, “Building diverse computer systems,” in Operating Systems, 1997., The Sixth Workshop on Hot Topics in. IEEE, 1997, pp. 67–72.

[52] B. Baudry, S. Allier, and M. Monperrus, “Tailored source code transformations to synthesize computationally diverse program variants,” in Proceedings of the 2014 International Symposium on Software Testing and Analysis. ACM, 2014.

[53] D. Williams, W. Hu, J. W. Davidson, J. D. Hiser, J. C. Knight, and A. Nguyen-Tuong, “Security through diversity: Leveraging virtual machine technology,” IEEE Security & Privacy, 2009.