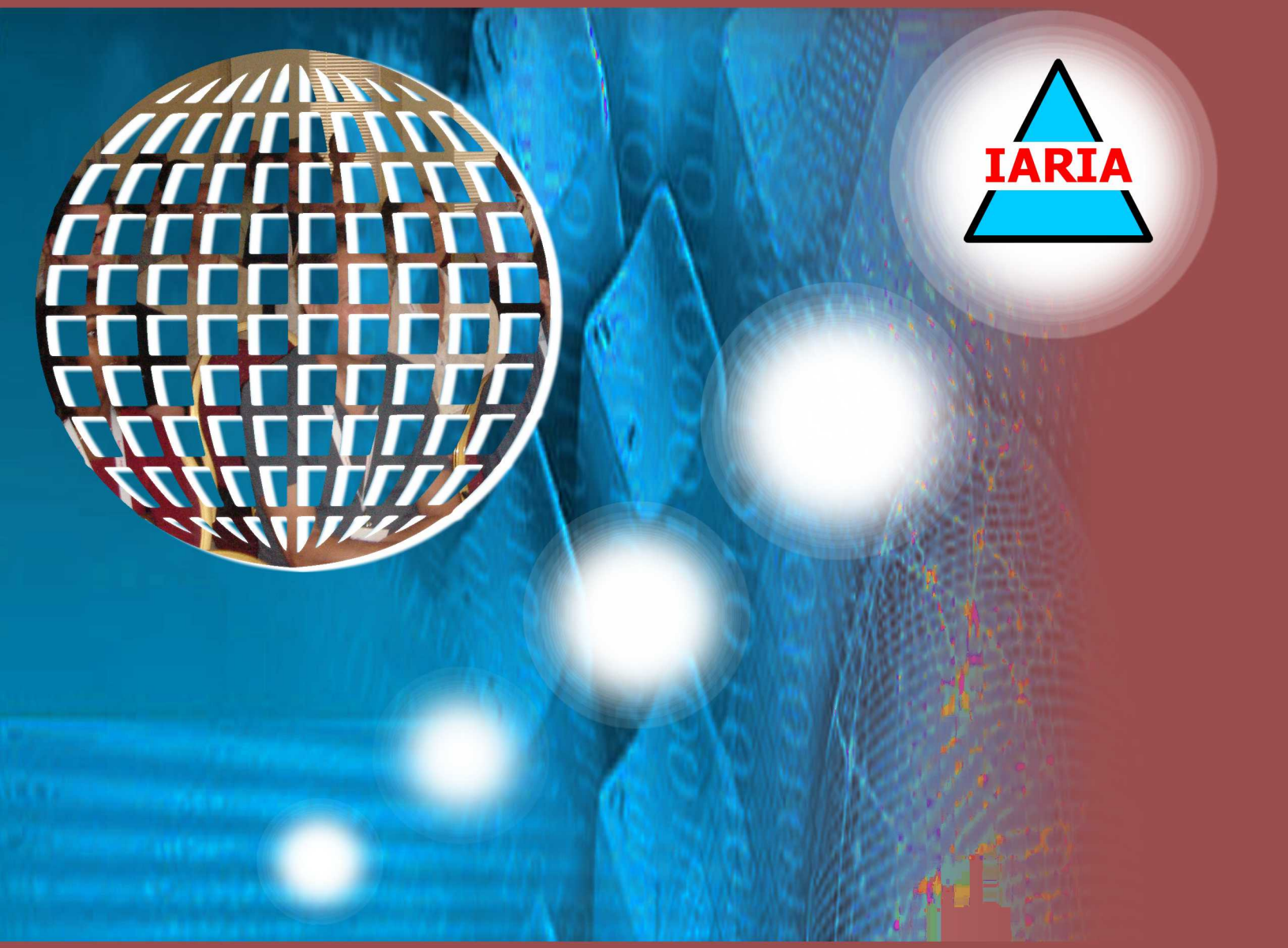
International Journal on Advances in Security



2021 vol. 14 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

Sponsored by IARIA
www.iaria.org

International Journal on Advances in Security
Volume 14, Number 1 & 2, 2021

Hyunsung Kim, Kyungil University, Korea
Geir M. Køien, University of Agder, Norway
Ah-Lian Kor, Leeds Metropolitan University, UK
Evangelos Kranakis, Carleton University - Ottawa, Canada
Lam-for Kwok, City University of Hong Kong, Hong Kong
Jean-Francois Lalande, ENSI de Bourges, France
Gyungho Lee, Korea University, South Korea
Clement Leung, Hong Kong Baptist University, Kowloon, Hong Kong
Diego Liberati, Italian National Research Council, Italy
Giovanni Livraga, Università degli Studi di Milano, Italy
Gui Lu Long, Tsinghua University, China
Jia-Ning Luo, Ming Chuan University, Taiwan
Thomas Margoni, University of Western Ontario, Canada
Rivalino Matias Jr ., Federal University of Uberlandia, Brazil
Manuel Mazzara, UNU-IIST, Macau / Newcastle University, UK
Catherine Meadows, Naval Research Laboratory - Washington DC, USA
Carla Merkle Westphall, Federal University of Santa Catarina (UFSC), Brazil
Ajaz H. Mir, National Institute of Technology, Srinagar, India
Jose Manuel Moya, Technical University of Madrid, Spain
Leonardo Mostarda, Middlesex University, UK
Jogesh K. Muppala, The Hong Kong University of Science and Technology, Hong Kong
Syed Naqvi, CETIC (Centre d'Excellence en Technologies de l'Information et de la Communication),Belgium
Sarmistha Neogy, Jadavpur University, India
Mats Neovius, Åbo Akademi University, Finland
Jason R.C. Nurse, University of Oxford, UK
Peter Parycek, Donau-Universität Krems, Austria
Konstantinos Patsakis, Rovira i Virgili University, Spain
João Paulo Barraca, University of Aveiro, Portugal
Sergio Pozo Hidalgo, University of Seville, Spain
Yong Man Ro, KAIST (Korea advanced Institute of Science and Technology), Korea
Rodrigo Roman Castro, University of Malaga, Spain
Heiko Roßnagel, Fraunhofer Institute for Industrial Engineering IAO, Germany
Claus-Peter Rückemann, Leibniz Universität Hannover / Westfälische Wilhelms-Universität Münster / North-German Supercomputing Alliance, Germany
Antonio Ruiz Martinez, University of Murcia, Spain
Paul Sant, University of Bedfordshire, UK
Peter Schartner, University of Klagenfurt, Austria
Alireza Shameli Sendi, Ecole Polytechnique de Montreal, Canada
Dimitrios Serpanos, Univ. of Patras and ISI/RC ATHENA, Greece
Pedro Sousa, University of Minho, Portugal
George Spanoudakis, City University London, UK
Vladimir Stantchev, Institute of Information Systems, SRH University Berlin, Germany
Lars Strand, Nofas, Norway
Young-Joo Suh, Pohang University of Science and Technology (POSTECH), Korea
Jani Suomalainen, VTT Technical Research Centre of Finland, Finland
Enrico Thomae, Ruhr-University Bochum, Germany

Tony Thomas, Indian Institute of Information Technology and Management - Kerala, India
Panagiotis Trimintzios, ENISA, EU
Peter Tröger, Hasso Plattner Institute, University of Potsdam, Germany
Simon Tsang, Applied Communication Sciences, USA
Marco Vallini, Politecnico di Torino, Italy
Bruno Vavala, Carnegie Mellon University, USA
Mthulisi Velempini, North-West University, South Africa
Miroslav Velev, Aries Design Automation, USA
Salvador E. Venegas-Andraca, Tecnológico de Monterrey / Texia, SA de CV, Mexico
Szu-Chi Wang, National Cheng Kung University, Tainan City, Taiwan R.O.C.
Steffen Wendzel, Fraunhofer FKIE, Bonn, Germany
Piyi Yang, University of Shanghai for Science and Technology, P. R. China
Rong Yang, Western Kentucky University , USA
Hee Yong Youn, Sungkyunkwan University, Korea
Bruno Bogaz Zarpelao, State University of Londrina (UEL), Brazil
Wenbing Zhao, Cleveland State University, USA

International Journal on Advances in Security
Volume 14, Numbers 1 & 2, 2021

CONTENTS

Ronald Beaubrun, Université Laval, Canada
Alejandro Quintero, Polytechnique Montreal, Canada

Fraud Detection Using Multilayer Perceptron and Convolutional Neural Network

Omoyele Odeniyi
Department of Cyber Security
Federal University of Technology
Akure, Nigeria
eoodeniyi@futa.edu.ng

Oghenerukvwe Oyinloye
Department of Computer Science
Ekiti State University
Ado Ekiti, Nigeria
oghenerukevwe.oyinloye@eksu.edu.ng

Aderonke Thompson
Department of Cyber Security
Federal University of Technology
Akure, Nigeria
afthompson@futa.edu.ng

Abstract— **In recent time, precarious transaction activities have attained a systematic daily occurrence, imbuing landed, personal and intangible properties. Of all these, credit card fraud is the most catastrophic if not detected on time for easy retrieval from the perpetrator. So, the threat actor gains unauthorized access in order to obtain money. Machine learning and data science has revolutionized and enhanced prompt discovery of expedient hidden information in data. Therefore, in this study, we develop an efficient fraud detection framework using non-rule-based approach of Multi-layer perceptron (MLP) on a given financial transaction dataset. Frauds were correctly predicted and detected. The algorithms on the datasets evaluate their respective effectiveness vis-à-vis fraud detection in bank transactions. The results are compared and evaluated using various evaluation metrics. In addition, we explored a 1D-Convolutional Neural Network, leveraging on its strength of less computational resource requirement. Observation from the experimental result revealed a desired gradual high accuracy.**

Keywords- fraud; credit cards; Multi-layer perceptron*; *1D-Convolutional Neural Network, Big Data.

I. INTRODUCTION

Recent information technology (IT) proliferation deployed in major financial services by Nigerian banking institutions has led to an increase in threats posed to these systems. A real time analysis is of utmost importance to the finance sector, enhancing its operational mode and outcome in a short time frame of fraud occurrence [1]. Debit/Credit cards are one of the most common payment methods used over the Internet. It was asserted that financial fraud can be viewed as an act intended for deception involving financial transactions for personal gain purpose [2]. Fraudsters have it easier as most transactions do not require the presence of a bank account/card holder; stealing relevant customers' details or perform identity theft by posing as the customer at point of payments is all that is vital to perpetrating their acts. This includes phishing and unsuspecting customers, redirection to malicious websites with a hidden act of harvesting customers' banking details and information. Credit card fraud is equally viewed as a type of theft and fraud done using a payment card, as a fraudulent fund source in a transaction. Some security issues are mostly faced by banks everywhere, but the prevention of card fraud attracts high priority, and this is set to grow with the exponential rate of Internet awareness and transactions. Increase in online purchases has made criminals take advantage of various weak authentication checks to commit credit card fraud [3].

Models provide a way to mitigate these occurrences, protect clients' transactions and play an essential role in payment service providers' profitability and sustainability. All the aforementioned can be achieved using a fraud detection system (FDS). FDS is computational analysis fraud detection techniques via fraud identification or anomaly transactions in swift and proven techniques of machine learning as presented in [4]. Modeling of past credit card transactions has to do with detecting fraudulent transactions via the existing knowledge fraud. This model is then used to identify whether a new transaction is fraudulent or not in the two major existing fraud methods of physical and virtual frauds. Physical fraud is done by stealing a card and using it for the payment or purchasing while virtual fraud is committed by using someone's card details through the internet for transactions. Further classification of credit card fraud is given in Figure 1. Section I deals with the introduction of various acts of fraud. A guide to available credit card fraud is presented in Section II, while Section III gives a detail of related study in the fraud detection domain. In Section IV, multilayer perceptron methodology approach to fraud detection is extensively discussed. Implementation of a feed forward Artificial Neural Network for the machine learning approach is presented in Section V in addition to Section VI which further shows the implementation with

various parameters. Observations from the proposed model and evaluation are given in Section VII with the performance of the Logistic regression study based on the same dataset.

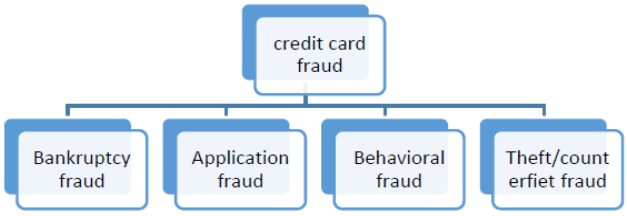


Figure 1. Classification of Credit Card Fraud

II. MAJOR METHODS USED TO MITIGATE CREDIT CARD

There are basically two major forms of mitigating credit card fraud; it could be in the preventive or detective mode. The preventive mode involves blocking fraudulent transaction at the point of transaction. Such as passwords, pin and blocked cards; while the detective mode identifies successful fraud transaction through predictive models with machine learning approach.

Traditionally, fraud resolution process usually involves: fraud detection, investigation, confirmation, and prevention. Therefore, a self-learning computer program automates the above processes using various methods. Signature based detection method detects fraud traces through the signature technology using known patterns or byte sequence; it is efficient for known frauds. However, fraudsters have continued to manipulate the system by finding creative ways to beat signature strings. The anomaly detection method comes with the ability to detect both known and novel frauds; although, this method is limited by false positive error, that is, previously unknown legitimate transactions. Consequently, this paper exploits machine learning (see Section IV) to detect fraudulent activities as well as measuring its performance.

III. LITERATURE REVIEW

Financial fraud had been a major challenge for corporate organizations, government and most specifically businesses that utilize information technology. Financial fraud is defined as an intentional act of deception involving financial transactions for personal purpose gain. Another definition for financial fraud is "to take advantage over another by false representations" which include "surprise, trickery, cunning and unfair ways through which another is cheated" [2]. Globally, fraud costs some financial industry approximately $80 billion annually while the United States' credit and debit card issuers alone lost $2.4 billion.

The financial fraud occurrence in any organization undermines both the effort and prospects. Financial fraud brings about losses owing to theft, distrust in transaction, and litigation. These losses owing to fraud are grossly detrimental to institutions in which they occur. As advances in cloud technology plums and cyber-security measures are not commensurate, there exists high possibility of financial fraud bound to threaten businesses worldwide. Detection of financial fraud had not come so easy; it is mostly at a high cost and time. The cost of financial fraud reported is about $1 million per incident, occupational fraud costs $150 to $200,000 per incident while losses due to fraud costs an average of 5% of gross profit and take around 24 to 36 months to discover - usually via a tip (40%), by accident (20%), or during an audit (10%). Some motivations for committing financial fraud have been reported and identified by senior management to be most responsible for most fraud [5].

The authors in [5] argued that meeting external forecasts emerged as the primary motivation and it was conceptualized that three elements common among all fraud is called the fraud triangle. These elements include a perceived pressure, a perceived opportunity, and a rationalization of the fraud act. in addition to the trio, is motivation for need, greed and addictions (or vices). This is with the assertion that the motivation for greed in turn feeds the motivation for vices. Capping it all, these motivations become a vicious cycle leading to fraud. thus, financial fraud is categorized mainly into three areas: bank fraud, corporate fraud and insurance fraud. Bank fraud is subdivided into credit card fraud, mortgage fraud and money laundering fraud [6].

Fraud modelling is one important tool in addressing financial fraud. It expands in importance as corporate organizations and government determine which type of models to use and continuous update in order to protect against evolving threats. In the past, traditional fraud models are used to automatically detect unauthorized transactions such as determining when a card has been used without the owner's consent. Most card issuers use fraud models to identify fraudulent card usage in order to maintain the integrity and security of their network as it is core to earning trust in online business world. However, diverse range of payment services offered by organizations and businesses to clients also presents higher opportunities for fraud occurrence. Consequently, fraud models provide a way to mitigate these occurrences, protect clients' transactions and play an essential role in payment service providers' profitability and sustainability with attributes of a given transaction as variables used in fraud models. Thereafter, it classifies or attempts to label the transaction fraudulent or legitimate (see Sections V-VII). Some extensive models label the type or category of fraud. Some of the common attributes used by fraud models include: Merchant (the business charging the transaction), transaction location, amount, type (online or offline), volume, account history, transaction history, and so on, depending on the amount of attribute information captured in a transaction. The five basic fields, which describe type, time (hours, minutes), location, amount, and date (week days) of a transaction were used in the fraud model. While 16 significant ratios out of 29 financial ratios were used in

detection of fraud in the financial statements of banks, which were categorized into asset quality ratios, earnings and profitability ratios, liquidity/solvency ratios, long term solvency/leverage ratio, capital adequacy ratio, cash flow analysis and trends. These fraud models utilized 29 variables of which 24 are financial variables while 5 are non-financial variables as it proved that model tools based on financial numbers, linguistic behaviour, and non-verbal vocal cues have each demonstrated the potential for detecting financial fraud. Fifty-one (51) financial ratios were utilized in detecting fraud in financial statements by means of financial ratios [7].

Notable fraud detection models are mainly categorized as rule-based models and algorithmic (or machine learning) models. Rule-based models are collection of rules used to detect fraudulent transactions with a single rule containing as a set of conditions that, when present, labels a transaction either as fraudulent or not. Rule-based models are made up of an expert knowledge base. In addition, new rules evolve from time to time because of inference action on streams of time changing data. However, one major limitation of rule-based fraud models is time complexity in handling big data. Algorithmic models make use of machine-learning methods to classify a transaction as either fraudulent or legitimate. Algorithmic models are more complex than rule-based models; this is dependent on the type of algorithm used. These models are computationally complex than rule-based models but achieve high performance. They are far better at detecting complex relationships between variables than the rule-based models. Machine-learning methods also require a pre-requisite of having many variables to implement and ensure learning. Therefore, when there is limited number of variables usage, the benefit of algorithmic methods over rule-based models is diminished.

The review on financial accounting fraud detection based on data mining techniques was motivated by the idea that the failure of internal auditing system of the organization in identifying the accounting frauds has led to the use of specialized procedures to detect financial accounting fraud. The findings of this review showed that data mining techniques such as logistic models, neural networks, Bayesian belief network, and decision trees have been applied most extensively to provide primary solutions to the problems inherent in the detection and classification of fraudulent data. In [7], financial fraud detection using vocal, linguistic and financial cues is presented and observed that these methods for automating financial fraud detection (FFD) have mainly relied on financial statistics; although, some recent studies have suggested that linguistic or vocal cues may also be useful indicators of deception. The hypothesis investigated in the study is that an improved tool (based on financial numbers, linguistic behaviour, and non-verbal vocal cues) could be developed if specific attributes from these feature categories were analysed concurrently. A set of 1,572 public company quarterly earnings conference call audio file samples was used in the study. The authors re-affirmed that earnings from conference calls are ideal for investigation because they involved corporate executives publicly discussing financial information, thereby simultaneously providing financial, linguistic and vocal cues. The study proved that tools based on financial numbers, linguistic behaviour, and non-verbal vocal cues have each demonstrated the potential for detecting financial fraud. However, it is quite tasking (and computationally intensive) to concurrently source and compute large amount of vocal and linguistic data [8].

In another study, a difference between precision-recall and Receiver Operator Characteristic (ROC) curves for evaluating the performance of credit card fraud detection models was motivated by the need to solve the problem of fraudulent transactions detection with use of machine learning for legitimate or fraudulent the credit card transactions classification. In order to solve this problem, the precision-recall curves are described as an approach. Weighted logistic regression is used as an algorithm level technique and random under-sampling is proposed as data-level technique to build credit card fraud classifier. Performance evaluation of these approaches adopted the ROC curves, which showed the variance of the number of correctly classified positive examples with the number of incorrectly classified negative examples. However, ROC curves present an overly optimistic performance view. It established that precision-recall curves have more advantages than ROC curves in dealing with credit card fraud detection. Nevertheless, the study was limited by inability to find the best solution to the problem of imbalanced data in the dataset [9].

In the same vein, a study on "Combatting Financial Fraud: A Co-evolutionary Anomaly Detection Approach" evolved around the motivation of the major difficulty in anomaly detection which lies in discovering boundaries between normal and anomalous behaviour. The objective was to present a co-evolutionary algorithm which tackles the anomaly detection problem and discover the boundary between normal and abnormal behaviour. The co-evolutionary algorithm was used to provide a competitive interaction between different populations which minimize detection errors and the adaptive evolutionary environment accelerated by the process of finding good solution. The authors implemented the algorithm using anonymized transactional data from a real financial institution. The data set contains two-year Automated Bank Machine (ABM) and Point of Sale (POS) fraud-free transaction history. The research has contributed to knowledge by using concept of evolution to detect anomalies in fraudulent transactions only it was not applied to realistic data [10].

IV. METHODOLOGY

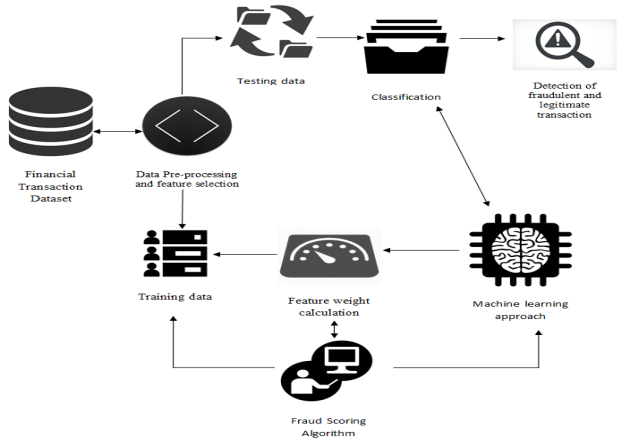
The study deploys multilayer perceptron approach to detect fraud using financial datasets. Each transaction by a customer on card contains the transaction API, which is stripped into attributes. The attributes (model variables)

from the API include; Source IP address, Destination IP address, Card pan, Location of transaction, Item bought, Unit of items bought, Amount of transaction and the Date and Time of transaction. The model architectural design is depicted in Figure 2. The Architecture is divided into 3 major parts, namely:

i. Data preprocessing & Feature Selection
ii. Data Training & Learning
iii. Classification

Financial credit card datasets were selected (Dataset 1 and Dataset 2) were obtained from "Kaggle Data Repository" [19] which are publicly available containing anonymized real-life credit card transactions with an evident presence of fraudulent cases. Dataset 1 was obtained from Kaggle Data Repository, and contains anonymized data to protect user's vital information. Data was from Credit Card Transactions for users in Europe in 2013. It has 284,808 entries. It has 31 attributes with class labels The Dataset 1 sample is shown in Table 1.

Dataset 2 contains anonymized data to protect users' vital information, Data contains credit card transactions. It has 151,113 entries. It has 11 attributes with class labels, partitioned into testing set and training set. Training set contained 105,778 records and testing set had 45,335 records. Sample records of the Dataset 2 are shown in Table II.



2. Architectural design of the model

The data pre-processing and preparation was carried out on the raw financial dataset to remove outliers using max-min normalization technique. As shown in equation (1)

$$Normalized\_Value = \frac{(f_{value} - f_{\min})}{(f_{\max} - f_{\min})} \quad (1)$$

where f_{value}, is the feature value to be normalized, f_{min} is the minimum feature value and f_{max} is the maximum feature value respectively.

Feature selection was performed by computing feature importance. This is done using Information gain calculation. Thus, given a set of financial transaction dataset S_c

$$E(F) = \sum_{j=1}^{c} \frac{S1_j + \ldots + Sc_j}{S} * I(si_j, \ldots, sc_j) \quad (2)$$

where (I = information, S = total number of financial transaction data instances, c = total classes (i.e., fraudulent and legitimate classes, F = Features)
The information gain, G(F) is defined as:

$$G(F) = I(s_1, s_2, \ldots, s_c) - E(F) \quad (3)$$

Features with high information gain are selected for model development while the others are removed.

TABLE I. SAMPLE OF DATASET 1

Time	V1	V2	V3	V4	V5	V6	V7	V8
0	-1.35981	-0.07278	2.536347	1.378155	-0.33832	0.462388	0.239599	0.098698
0	1.191857	0.266151	0.16648	0.448154	0.060018	-0.08236	-0.0788	0.085102
1	-1.35835	-1.34016	1.773209	0.37978	-0.5032	1.800499	0.791461	0.247676
1	-0.96627	-0.18523	1.792993	-0.86329	-0.01031	1.247203	0.237609	0.377436
2	-1.15823	0.877737	1.548718	0.403034	-0.40719	0.095921	0.592941	-0.27053
2	-0.42597	0.960523	1.141109	-0.16825	0.420987	-0.02973	0.476201	0.260314
4	1.229658	0.141004	0.045371	1.202613	0.191881	0.272708	-0.00516	0.081213
7	-0.64427	1.417964	1.07438	-0.4922	0.948934	0.428118	1.120631	-3.80786
7	-0.89429	0.286157	-0.11319	-0.27153	2.669599	3.721818	0.370145	0.851084
9	-0.33826	1.119593	1.044367	-0.22219	0.499361	-0.24676	0.651583	0.069539
10	1.449044	-1.17634	0.91386	-1.37567	-1.97138	-0.62915	-1.42324	0.048456
10	0.384978	0.616109	-0.8743	-0.09402	2.924584	3.317027	0.470455	0.538247
10	1.249999	-1.22164	0.38393	-1.2349	-1.48542	-0.75323	-0.6894	-0.22749
11	1.069374	0.287722	0.828613	2.71252	-0.1784	0.337544	-0.09672	0.115982
12	-2.79185	-0.32777	1.64175	1.767473	-0.13659	0.807596	-0.42291	-1.90711
12	-0.75242	0.345485	2.057323	-1.46864	-1.15839	-0.07785	-0.60858	0.003603
12	1.103215	-0.0403	1.267332	1.289091	-0.736	0.288069	-0.58606	0.18938
13	-0.43691	0.918966	0.924591	-0.72722	0.915679	-0.12787	0.707642	0.087962
14	-5.40126	-5.45015	1.186305	1.736239	3.049106	-1.76341	-1.55974	0.160842

creditcard

TABLE II: SAMPLE OF DATASET 2

user_id	signup_time	purchase_time	purchase_	device_id	source	browser	sex	age	ip_address	class
22058	2/24/2015 22:55	4/18/2015 2:47	34	QVPSPJUC	SEO	Chrome	M	39	732758368.8	0
333320	6/7/2015 20:39	6/8/2015 1:38	16	EOGFQPIZ	Ads	Chrome	F	53	350311387.9	0
1359	1/1/2015 18:52	1/1/2015 18:52	15	YSSKYOSJ	SEO	Opera	M	53	2621473820	1
150084	4/28/2015 21:13	5/4/2015 13:54	44	ATGTXKYK	SEO	Safari	M	41	3840542444	0
221365	7/21/2015 7:09	9/9/2015 18:40	39	NAUITBZF	Ads	Safari	M	45	415583117.5	0
159135	5/21/2015 6:03	7/9/2015 8:05	42	ALEYXFXI	Ads	Chrome	M	18	2809315200	0
50116	8/1/2015 22:40	8/27/2015 3:37	11	IWKVZHJC	Ads	Chrome	F	19	3987484329	0
360585	4/6/2015 7:35	5/25/2015 17:21	27	HPUCUYLI	Ads	Opera	M	34	1692458728	0
159045	4/21/2015 23:38	6/2/2015 14:01	30	ILXYDOZI	SEO	IE	F	43	3719094257	0
182338	1/25/2015 17:49	3/23/2015 23:05	62	NRFFPPHZ	Ads	IE	M	31	341674739.6	0
199700	7/11/2015 18:26	10/28/2015 21:59	13	TEPSJVVX	Ads	Safari	F	35	1819008578	0
73884	5/29/2015 16:22	6/16/2015 5:45	58	ZTZZJUCR	Direct	Chrome	M	32	4038284553	0
79203	6/16/2015 21:19	6/21/2015 3:29	18	IBPNKSMC	SEO	Safari	M	33	4161540927	0
299320	3/3/2015 19:17	4/5/2015 12:32	50	RMKQNVE	Direct	Safari	M	38	3178510015	0
82931	2/16/2015 2:50	4/16/2015 0:56	15	XKIFNYUZ	SEO	IE	M	24	4203487754	0
31383	2/1/2015 1:06	3/24/2015 10:17	58	UNUAVQX	SEO	Safari	F	24	995732779	0
78986	5/15/2015 3:52	8/11/2015 2:29	57	TGHVAWE	SEO	FireFox	M	23	3503883392	0
119824	3/20/2015 0:31	4/5/2015 7:31	55	WFIIFCPI	Ads	Safari	M	38	131423.789	0
357386	2/3/2015 0:48	3/24/2015 18:27	40	NWSVDOI	Ads	FireFox	M	24	3037372279	0
289172	7/17/2015 5:48	11/12/2015 22:08	46	KFZGQIWI	Direct	FireFox	F	53	1044590098	0

V. MULTI LAYER PERCEPTRON (MLP)

The implementation is a feed-forward artificial neural networks; MLP consists of the input layer, output layer, and one or more hidden layers. Each layer of MLP includes one or more neurons directionally linked with the neurons from the previous and the next layer. Figure 3 represents a 3-layer perceptron having three inputs, two outputs, and the hidden layer including five neurons.

The values retrieved from the previous layer are summed up with certain weights, individual for each neuron, plus the bias term [11]. The sum is transformed using the activation function.

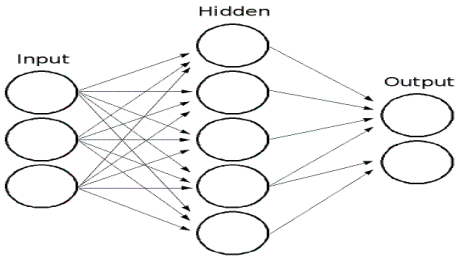


Figure 3. A Multi-Layer perceptron

The perceptron computes a single output from multiple real-valued inputs by forming a linear combination according to its input weights and then putting the output through some nonlinear activation function:

Given output (u_i)

$$u_i = \sum_{j=1}^{n} (w_{i,j}\ x_j + b_i) \qquad (4)$$

With the activation function (φ) applied, mathematically the MLP can be written as:

$$y_i = \varphi\left(\sum_{j=1}^{n} (w_{i,j}\ x_j + b_i)\right) \qquad (5)$$

where w = weight going to the hidden unit layer

x= Input to hidden unit

b= bias input

φ= Activation function

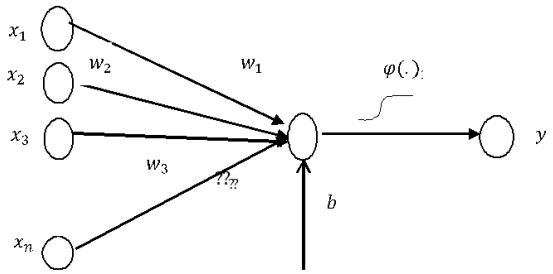


Figure 4. Representation of the MLP equation

A. Learning Algorithm

The MLP uses a backpropagation algorithm to learn and train from the dataset.

The back-propagation algorithm is in 2 phases:

- The forward pass phase- computes 'functional signal', feed forward propagation of input pattern signals through network.
- Backward pass phase- computes 'error signal', *propagates* the error *backwards* through network starting at output units (where the error is the difference between actual and desired output values).

Forward pass Algorithm

- Step 1: Initialize weights at random, choose a learning rate η
- Until network is trained:
- For each training example i.e., input pattern and target output(s):
- Step 2: Do forward pass-through net (with fixed weights) to produce output(s)
  - i.e., in Forward Direction, layer by layer:
    - Inputs applied
    - Multiplied by weights
    - Summed
    - 'Squashed' by sigmoid activation function
    - Output passed to each neuron in next layer
  - Repeat above until network output(s) produced

Backward pass /Back propagation of error

- Compute error (delta or local gradient) for each
- output unit δ *k*
- Layer-by-layer, compute error (delta or local
- gradient) for each hidden unit δ *j* by backpropagating
- errors (as shown previously)
- Next, update all the weights Δ*wij*
- By gradient descent, and go back to Step 2

The overall MLP learning algorithm, involving forward pass and backpropagation of error (until the network training completion), is known as the Generalized Delta Rule (GDR), or more commonly, the Back Propagation (BP) algorithm.

VI. MLP IMPLEMENTATION

The MLP model was implemented on a Personal Computer with 2.30 GHz and 8GB of RAM in Microsoft Windows 10 Operating system platform and Microsoft Excel 2013 with Python Programing Language. The MLP training was defined with parameters epochs = 20, dim_size = 15, num_seq = 30, batch_size = 200, activation function = Sigmoid.

Due to the high imbalance in the datasets, the data were synthetically balanced using the smote method, the datasets 1 and dataset 2 stored in csv format were loaded into python 3.6 IDLE via a read_csv () command. The datasets were divided into two parts (Input and Output). The input data are those with the attributes while the output data contain the target class ('Fraudulent' and 'Normal').

A. *Evaluation Metrics*

The evaluation of the model was carried out using the various evaluation metrics such as Accuracy, Precision, F1-score, Recall and False alarm rate.

Accuracy: is defined as the number of correct predictions made by the model. It is the proportion of the total number of correct predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (6)$$

False Alarm Rate (FAR)/False Positive rate: is a ratio of wrongly classified normal instances.

$$False\ Alarm\ Rate = \frac{FP}{TN + FP} \quad (7)$$

Precision: defines the results classified as positive by the model, how many were actually positive. It is the number of items correctly identified as positive out of total true positives.

$$\text{Precision} = \frac{\text{true positive}}{\text{true positive + false positives}} \quad (8)$$

Recall: It is the number of items correctly identified as positive out of the total items classified as positive.

$$\text{Recall} = \frac{\text{true positive}}{\text{true positive + false negatives}} \quad (9)$$

F1-Score: is the weighted average of the precision and the recall, it takes both false negatives and positives into the account and gives a better outlook especially in an uneven class distribution it is given as:

$$\text{F1 Score} = 2\left(\frac{\text{Precision} * \text{recall}}{\text{Precision} + \text{recall}}\right) \quad (10)$$

where True positive (TP) represents data detected as fraudulent, True negative (TN) represents data detected as legitimate, False positive (FP) represents normal data detected as fraudulent, and False Negative (FN) is denoted as fraud data detected as normal.

VII. RESULTS

In this section, an evaluation of the study with some metrics is presented with the two datasets. Dataset I reveals the significance of dataset that is characterized with minimum missing data. This is presented in Tables III and IV. The graphical representation of these datasets is presented in Figure 5.

TABLE III: EVALUATION RESULT ON DATASET I

Model	*Accuracy (%)*	*F1 score (%)*	*Precision (%)*	*Recall (%)*	*False Alarm rate (%)*
Multi-Layer Perceptron	96.4	96.3	99.1	93.6	0.001

TABLE IV: EVALUATION RESULT ON DATASET 2

Model	*Accuracy (%)*	*F1 score (%)*	*Precision (%)*	*Recall (%)*	*False Alarm rate (%)*
Multi-Layer Perceptron	77.4	71.4	96.9	56.5	0.002

From Figure 5 we can conclude that the proposed model performed appreciably better with dataset using the evaluation metrics.

B *Performance of Dataset 1 and Datset 2 Using MLP*

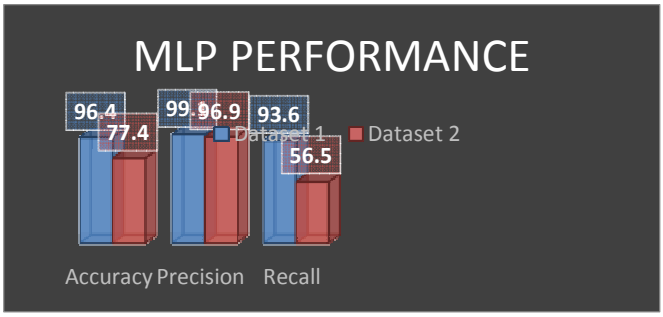


Figure 5. Performance of Dataset 1 and Dataset 2 using MLP

C *Comparative Evaluation*

The results of this model were thereafter compared with the results of a work that was implemented using Logistic regression machine learning approach with the same dataset 1 is the result.

TABLE V: COMPARATIVE EVALUATION OF MLP AND LOGISTIC REGRESSION

Model	Accuracy (%)	Precision (%)	Recall (%)
Multilayer Perceptron	96.4	99.1	93.6
Logistic Regression	Not given	71	64

This model performed impressively against the performance of the Logistic regression study with the same dataset. Weighted logistic regression was used as an algorithm level technique and random under-sampling was

used as data-level technique to build credit card fraud classifier. The classification used in the study was Logistic Regression and the performance metrics are Recall and Precision. A graphical evaluation report of the two models is illustrated in Figure 6.

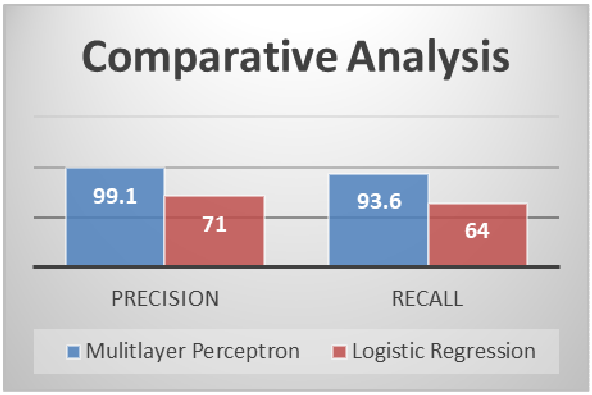


Figure 6. Comparative Analysis of Our Model (MLP) and Logistic Regression

VIII. CONVOLUTIONAL NUERAL NETWOK (CNN)

CNN is a type of deep neural networks that works best with image recognition [12]. CNN networks have been used in video and image applications such as objects/image detection [13]. It is based on the convolution of images and extraction of salient features based on filters that are learned by the network during training phase [14]. Aside the input layer, the stacked layers of Convolutional neural network include: convolution layer, activation layer, pooling layer, and fully-connected layer [12]. A typical sample layers of CNN is presented in Figure 7.

a) Input layer: the input to this layer are usually image pixels (either gray-scale or RGB)

b) Convolution layer: This is the heart of CNN network. It is based on convolutional filtering such that during training filter weights are learned. In order to extract more complex features from image input, several filters are used, and this determined the depth of the convolution layer. The filter is also referred to as the kernel, and it has height and width in a matrix form (e.g., a filter size of 3x3 will have nine weights). An important component of this layer is the stride: it determines the number of pixels that a kernel window will slide through.

c) Activation layer: CNN generally uses Rectified Linear Units (ReLu) activation function. The ReLU adds non-linearity into the network and at the same time provides non-saturating gradients for positive net inputs. It changes the output of a neuron to zero when the net input of a neuron is less than zero ($y = \max(0, w^T x + b)$) [14].

d) Pooling layer: This layer reduces the spatial dimension of an image pixel size [15]. The layer can either be a Max pooling or an Average pooling. In max pooling, the maximum pixel intensity of a locality (window size) is taken as representative of that locality, while in average pooling the average is taking instead of the maximum.

e) Fully-connected layer: each neuron in this layer is connected to all neuron of the previous layer. More so, there are no weight sharing but neuron(s) receives different set of weights form preceding layers.

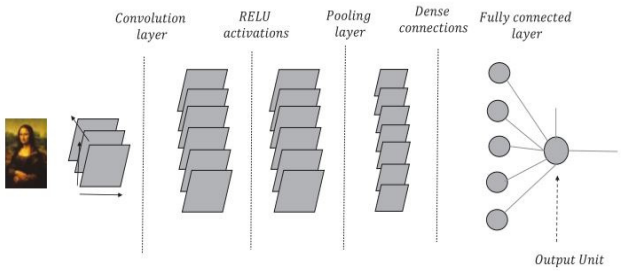


Figure 7. Sample layers of CNN [14]

Commonly used CNN forms are 2D-CNN and 1D-CNN. The two CNNs both share the same characteristics and approach but they differ in their respective filters opereations as it moves across data and in the structure of their input dimensions. 1D convolutional neural network has been used in analysis of a time series for sensor data, mechanical or aerospace [16], audio recording, Fault detection [17], patient ECG [18]. Authors in [16] emphasized on the advantages of 1D-CNN over 2D-CNN has having lesser computational complexity, shallow architecture with potential to learn complex features, required less computational resources (CPU rather than GPU), and well suited for real-time and low-cost applications on hand-held devices.

In recent times, there has been increase in fraud, which has resulted to loss of money and lack of trust in financial systems worldwide. In the financial systems of various countries of the world, there exist several techniques for fraud detection, which has also evolved over time. Fraud detection encompasses the observation of the activities of users so as to avoid, perceive and estimate unwelcomed behavior which include delinquency, fraud, intrusion, and account defaulting [20]. Credit card fraud can be described as any unauthorized account activity by an unauthorized person for which the account was unintended for; thus, action is engaged to halt the abuse and adopt risk management practices to secure imminent fraud actions [20]. Although, credit card has become dominant in the world's financial system similarly, fraud is increasing globally.

- CNN Fraud Architecture

Authors in [22] proposed a Convolutional Neural Network based framework for detecting surreptitious fraud patterns in credit card transactions. The authors transformed transaction data into a feature matrix for each record, by which the inherent relations and interactions in time series was revealed for the CNN model. They combined the cost-based sampling method in characteristic space, the extremely imbalanced sample sets are alleviated, yielding a superior performance of fraud detection. The proposed framework is shown in Figure 8.

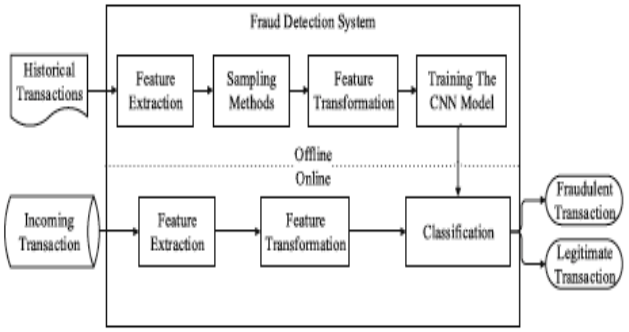


Figure 8. Credit Card Illustration Fraud Detection System [22]

In addition to the online and offline segments of the proposed framework, the proposed system adopts trading entropy to a group of traditional features in order to model a more complicated consuming behavior. As regards to data mining, the model was structured after feature engineering and it was realized that credit card data was imbalanced, which resulted to proposing and adoption of a cost-based sampling method for the generation of synthetic frauds and thus transform features into a feature matrix so as to fit the model. Subsequently, the proposed model was simulated and evaluated alongside other industry-based models (SVM, RF and NN). Simulation Results proved that the cost-based sampling method uses additional legitimate data and improves the imbalanced problem, hence the CNN model when simulated on various sample sets, attains the best performance.

Research outcome in [21] posited that the widely adoption of CNN architecture is due to its flexibility structure and obtains the feature automatically, thereby resolving so many classification problems and in an exact specification situation, the structural feature settings of CNN could be modified to achieve optimum performance. The authors proposed a three Convolutional Neural Network models to resolve Fraud Account Detection. The models proposed include the Network Topological Data (NTD-CNN), Time Series Data (TTD) tagged as TTD-CNN model and a CNN model that combines the two kinds of Heterogenous Data Features (HDF), which are extracted from the former two kinds of data, tagged HDF-CNN model. They further proposed a wholistic account transaction network mathematical model, which was used as the basis of learning network vector of accounts. The network comprises of the transaction relationships cum timestamp data, this represented account's historical trading behavior.

The study adopted a DirectedWalk algorithm was used to learn the account's network vector; this quantified the network local topological arrangements of transaction network into high dimensional vectors. The research explored data set from the Department of Economic Investigation, which avails transaction data of real bank accounts and subsequently subjected it to simulation. The experimental result on real data set revealed that HDF-CNN achieved improvements when compared with other proposed CNN models in classification performance.

Furthermore, the gain of Neural networks and deep learning is its ability to estimate complex nonlinear relationships, fault tolerance, robustness and find the best solutions at a very high speed is asserted in [23], hence, proven to portray a unique performance in video processing, natural language processing and image recognition. Conversely, for structured data particularly online transaction data, neural and deep learning models have displayed poor performance since the available dimensions of the transaction data are limited. The authors proposed CNN based on feature sequencing to ameliorate fraud detection in online transactions as presented in Figure 9.

CNN was applied to directly use low dimensional raw features as the input into the model, in order to enhance the sequence of features, thus a feature sequencing layer is added automatically. The proposed approach saves variable derived time, learns derivative features that benefit the classification results and reduces human interference. The architecture is divided into two segments, which include transaction detection and training segments.

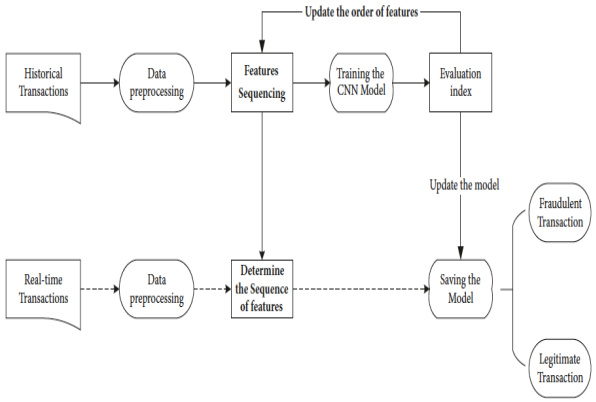


Figure 9. A Fraud Detection model [23]

The training segment was further divided into feature sequencing layer and CNN. The transactions features were optimized using feature sequencing, historical data was cleaned up and inputted into feature sequencing layer. The

proposed model was simulated by training the CNN framework, and the feature sequence order is modified by the effect feedback. Upon simulation, the results revealed that the proposed the CNN architecture based on feature rearrangement entrenched in the research had an outstanding experimental implementation with good stability.

A hybrid model for Fraud detection in credit card is presented in [24], it comprises CNN and K-Nearest Neighbors (KNN) Classification. The proposed system adopts Machine Learning techniques such as KNN-Classification and Convolutional Neural Network. These techniques are implemented on data-features such as Customer ID, gender, Merchant ID, age, Merchant type of customers. The system also adopts a serialized approach in fraud detection and the model trained such that it feeds the output of the CNN model into the training set of the KNN.

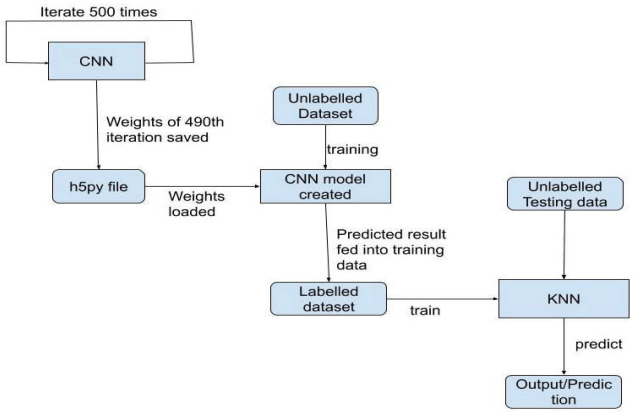


Figure 10. A proposed Fraud Detection hybrid model [24]

In the proposed model, CNN and Long short-term memory (LSTM) algorithms were applied to the first layer of the model to enhance the detection of fraud and induced the model towards identifying fraudulent transaction attempts. The analysis of the sequence of data and memory checking was enforced by LSTM. The output of this layer is also stored as the classification label for the training set, to feed into the KNN model. The KNN layer is used to quickly classify through the resultant set, making the model faster and more accurate.

Experimental results revealed that the accuracy of CNN upon training the data for 490 repetitions had 87.79% and a logarithmic loss of 3.90. The K-Nearest Neighbor classification had 90.5%. Upon hybridization of the two techniques, the resultant CCFDS model had an accuracy of 98% with a logarithmic loss of 0.647. This accuracy of the CNN is amplified by 10% when imputed into the hybrid model with KNN, and can only increase if trained over a bigger balanced dataset.

Authors in [25] used deep learning techniques to detect fraud in mobile communications. Dataset from a mobile communication network was used for experiment purposes and learning features extracted and grouped into fraudulent and non-fraudulent activity; as presented in Figure 11.

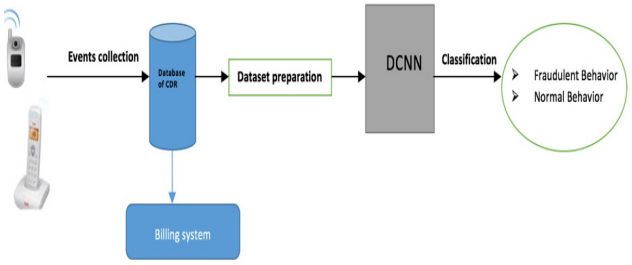


Figure 11. General Fraud Detection Framework [25]

Datasets were subjected to Simulations based on the proposed model and results showed that the performance of Deep Convolution Neural Networks (DCNN) method superseded that of Gradient Boosting Classifier, Random Forest and Support Vector Machines as regards to training and accuracy.

The multiple benchmarked machine learning techniques such as SVM, KNN and RF and deep learning methods such as autoencoders, CNN, RMB and DBN is presented in [26]. The authors sourced for datasets from European Union, Australia and Germany. The study adopted three evaluation metrics, which include the Area Under the ROC Curve (AUC), Matthews Correlation Coefficient (MCC) and Cost of failure. Simulation Findings revealed that for larger datasets, the best technique to adopt is SVM in combination with CNN to maximize performance while for small datasets, a combination of KNN, RF and SVM provides good enhancement and Convolutional Neural Networks has the best performance when compared with DBN, Autoencoders and RBM.

IX. RESULT WITH 1D-CNN

We experimented our proposed 1D-CNN approach with financial credit card datasets from kaggle data repository. The dataset has 284,808 entries with 31 attributes. We split the dataset into 80% training and 20% testing sets.

From Figure 12, the 1D-CNN architecture consists of two convolutional layers with filter size of 128 each, which are preceded by a max pooling layer and a batch normalization. Also, two convolutional layers with filter size of 256 and 512 respectively were added along with a batch normalization layer. Furthermore, we included another convolution layer preceded by a max pooling layer that has its output forwarded into the third batch normalization layer. Finally, we included two fully-connected layers (dense layers) such that the second dense layer has an output of 2 classes. Each convolutional layer used ReLU activation function while in the last dense layer we applied a SoftMax activation function.

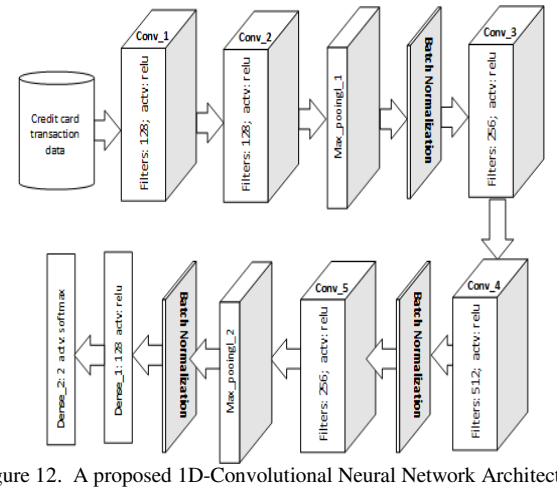


Figure 12. A proposed 1D-Convolutional Neural Network Architecture (CNN)

The essence of the batched normalization layer is to help overcome overfitting problem in our dataset due to unbalance class ratio and also to enhance our network accuracy. During the network training, we set our batch size to 1400, epoch to 150 and learning rate to 0.001. We applied Adam optimizer to optimize the loss gotten from cross-entropy loss function we applied in the 1D-CNN. As a result, our network was able to achieve a training accuracy of 99.53% after 150 epochs, so that both the train and test set accuracy rose gradually after 30 epochs, hence at 90 epochs the accuracies have surpass 90% illustrated in Figure 13. The impact of batch normalization techniques and the learning rate, facilitates the gradual increase in the network accuracy.

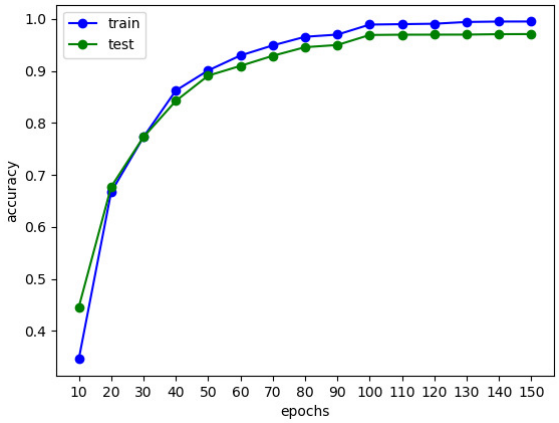


Figure 13. 1D-CNN train and test data accuracy

X. COMPARATIVE EVALUATION OF MLP AND 1D-CNN

A further comparison of MLP and 1D-CNN models on the test data resulted in 96.4% and 99% accuracy with the same dataset is the result as illustrated in Table VI.

TABLE VI: COMPARATIVE EVALUATION OF MLP AND 1D-CNN

Model	Accuracy (%)
Multilayer Perceptron	96.4
1D-CNN	>98

XI. CONCLUSION

In conclusion, the multilayer perceptron which used information gain method as feature selection technique for obtaining the most relevant features of the dataset was found to be effective in fraud detection; this is hopeful to be of high importance to the financial sector. This study established a fraud detection framework that is capable of unmasking real-time fraudulent transactions. The prediction of the MLP and ID-CNN proposed frameworks record high level of accuracy, precision, recall, good F1-score and very low false alarm rate. In addition, it is observed that the larger dataset, which is Dataset I, with MLP and 1D-CNN, yielded high evaluation values than Dataset II (a smaller dataset). This corroborates facts from literatures on the prediction accuracy in big data. Future work will be extended to Association Rule mining by improved apriori principles as well as hybridized approach with focus on computational complexities will be studied for suitability with big data.

REFERENCES

[1] A. Thompson, O. Oyinloye, L Aborisade, and E. Odeniyi, "A Fraud Detection Framework using Machine Learning Approach," Cyber 2019 Conference, Porto, Portugal.

[2] W. S. Albrecht, C. O. Albrecht, C. C. Albrecht, and M. F Zimbelman, "Fraud examination," 5th Edition, Cengage Learning, 2014.

[3] F. N. Ogwueleka, "Data mining application in credit card fraud detection system," *Journal of Engineering Science and Technology*, 2014, 6(3):311-322.

[4] H. Shao, H. Zhao, and G. Chang, "Applying data mining to detect fraud behaviour in customs declaration," *Proc. of 1st International Conference on Machine Learning and Cybernetics*, 2015, 1241-1244

[5] N. M. Brennan and M. McGrath, "Financial statement fraud: incidents, methods and motives," *Australian Accounting Review*, 2007, 17(2):49-61.

[6] P. L. Clifton, Vincent, S. Kate, and G. Ross, "A comprehensive survey of data mining-based fraud detection research," School of Business Systems, Faculty of Information Technology, Monash University, Clayton campus, Wellington Road, Clayton, Victoria 3800, Australia, 2012.

[7] C. S. Throckmorton, V. Mohan, J. M. William, and C. Leslie, "Financial fraud detection using vocal, linguistic and financial cues," 2018.

[8] F. Chowdhury and M. S. Ferdous, "Modelling cyber-attacks," International Journal of Network Security & Its Applications 9(4):13-31, July 2017.

[9] K. Dinaesh, "Cyber defense mathematical modeling and simulation," International Journal of Applied Physics and Mathematics, Vol. 2, No. 5, September 2012.

[10] P. Laerte, D. Marcelo, M. Bernardo, F. G. David, and D. T. Deus, "A Formal classification of internet banking

attacks and vulnerabilities in combatting financial fraud: A coevolutionary anomaly detection approach," International Journal of Computer Science & Information Technology (IJCSIT), vol. 3, 2015.

[11] S. Maes, K. Tuyls, B. Vanschoenwinkel, and B. Manderick, "Credit card fraud detection using Bayesian and neural networks," In Proceedings of the 1st International naiso congress on neuro fuzzy technologies 2002, pp. 261-270.

[12] T. A. Adesuyi, B. M. Kim, and Y. S. Shin, "A Brief on Snoring Data and Classification Methods," International Journal of Advanced Trends in Computer Science and Engineering, vol. 9(1), pp. 426-432, 2020.

[13] A. S Mohamed, N. Marbukhari, and H. Habibah, "A Deep Learning Approach in Robot-Assisted Behavioral Therapy for Autistic Children," International Journal of Advanced Trends in Computer Science and Engineering, vol. 8(1.6), pp. 437-443, 2019. https://doi.org/10.30534/ijatcse/2019/6381.62019.

[14] S. Pattanayak, "Pro Deep Learning with Tensorflow: A Mathematical Approach to Advanced Artificial Intelligence in Python," In: Pro Deep Learning with TensorFlow, 2017 eBook ISBN 978-1-4842-3096-1, DOI 10.1007/978-1-4842-3096-1

[15] H. Chabanne, A. D. Wargny, J. Milgram, C. Morel, and E. Prouff, "Privacy-Preserving Classification on Deep Neural Network," Sefran Identity & Security, Cryptology ePrint Archive, 2017. Intelligence in Python," Apress Media, pp. 178, 2017.

[16] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1D convolutional neural networks and applications- a survey," arXiv:1905.03554, 2019.

[17] T. Ince, S. Kiranyaz, L. Eren, M. Askar, and M. Gabbouj, "Real-time motor fault detection by 1-D convolutional neural networks," IEEE Transactions on Industrial Electronics, vol. 63(11), 2016.

[18] X. Zhou, X. Zhu, K. Nakamura, and N. Mahito, "ECG quality assessment using 1D-convolutional neural network," In Proc. IEEE ICSP, pp. 780-784, 2018.

[19] Kaggle Data Repository retrieved from kaggle.com.

[20] A. Choudhury, "Credit Card Fraud Detection by Neural network in Keras Framework," 2019. Retrieved from https://blog.usejournal.com/credit-card-fraud-detection-by-neural-network-in-keras-4bd81cc9e7fe.

[21] F. Live, W. Wang, Y. Wei, Y. Sun, J. Huang, and B. Wang, "Detecting Fraudulent Bank Account Based on Convolutional Neural Network with Heterogeneous Data," Mathematical Problems in Engineering, Volume 2019, Article ID 3759607, 11 pages. DIO: https://doi.org/10.1155/2019/3759607.

[22] K. Fu, D. Cheng, Y. Tu, and L. Zhang, "Credit Card Fraud Detection Using Convolutional Neural Networks," 2016. Retrieved from https://twin.scihub.se/6056/3e1860653f88184c2e1233617611748/fu2016.pdf#view=FitH

[23] X. Zhou, X. Zhang, L. Wang, and P. Wang, "A Model Based on Convolutional Neural Network for Online Transaction Fraud Detection," Security and Communication Networks, Volume 2018, Article ID 5680264, 9 pages. https://doi.org/10.1155/2018/5680264.

[24] G. Nancy, S. Kumar, S. Veena, N. Vinoth, and M. Bandyopadhyay, "Fraud detection in credit card transaction using hybrid model," AIP Conference Proceedings 2277, 130010, Nov. 2020. https://doi.org/10.1063/5.0025561

[25] A. Chouiekh and E. Haj, "ConvNets for Fraud Detection analysis," The First International Conference on Intelligent Computing in Data Sciences, Procedia Computer Science, vol. 127, pp. 133-138, 2018.

[26] P. Raghavan and N. Gayar, "Fraud Detection using Machine Learning and Deep Learning," International Conference on Computational Intelligence and Knowledge Economy (ICCIKE), Amity University Dubai, UAE, December 11-12, 2019, pp. 334-345.

ADAM - An Adversary-Driven Attack Modelling Framework for Model-Based Security Testing

Tina Volkersdorfer, Hans-Joachim Hof
Security in Mobility
CARISSMA Institute of Electric, Connected, and Secure Mobility (C-ECOS)
Technische Hochschule Ingolstadt, Germany
tina.volkersdorfer@carissma.eu, hans-joachim.hof@thi.de

***Abstract*—ADAM (Adversary-Driven Attack Modelling) is a framework for model-based security testing. It is the foundation of a systematic and holistic attack modelling to support consistent and comprehensible penetration tests on model level. ADAM can be used for the automation of security testing in the early phases of software engineering (e.g., manual security reviews) as well as providing attack information for testing activities in later phases of the development lifecycle (e.g., penetration tests). By using ADAM, it is possible to continuously and consistently address security in software development, even if no running code is available. This paper focuses on the presentation of the concept of ADAM, describing the necessary components, their use and giving an insight into how the ADAM framework can be used in the context of a simulation environment. ADAM captures different perspectives of an attack, by the simulation of an adversary that executes multiple attacks to reach a given goal. Thus, ADAM supports not only the automation of model-based security tests but the whole security testing on model level, e.g., including test case generation. Our preliminary evaluation shows that it is possible to use ADAM in a wide range of domains and that there is potential reuse of modelled elements.**

***Index Terms*—attack model; adversary model; model-based testing; security testing; penetration test.**

I. Introduction

In this extended paper, we present the Adversary-Driven Attack Modelling (ADAM) framework. This work expands the basic idea of a holistic attack modelling framework to support the model-based security testing presented in [1] by the following aspects:

- Specifying the adversary model utilising attributes for the adversary characteristic and the adversary goal.
- Giving an insight into how the Adversary-Driven Attack Modelling (ADAM) framework can be used and integrated, in the context of a simulation environment from the automotive domain, which represents the target model.

The ADAM framework is being developed in the research project "Modellbasierte Absicherung von Security und Safety für umfeldbasierte Fahrzeugfunktionen (MASSiF)" that addresses model-based safety and security testing in the automotive software domain. In the automotive domain, software engineers thoroughly use model-based safety engineering and model-based safety testing, e.g., to develop advanced driver-assistance systems [2]. However, to our best knowledge, there are currently no approaches for holistic attack-model-based security testing. This argument is also supported by [3]. Depending on what the use case requires, a suitable attack model of the existing multitude of isolated solutions is used. If the use case changes or new questions arise, the applied model may have to be updated, or further models may have to be used, e.g., the MITRE ATT&CK Framework [4] (used for details of a specific adversary profile) in contrast to attack trees [5] (focusing on the system security on identifying security improvements). Using different models or the constant development of new models is time-consuming and causes security to be inconsistent and untraceable, which in turn may have a negative impact on the quality of security testing. To close this gap, [1] introduced the basic idea of a holistic attack modelling framework to support model-based security testing. The concept provides an adversary-based and target-based foundation to automate security testing on model level.

Penetration tests can be used in different domains [6][7]. We will show that the idea of penetration testing can be applied in the early stages of software development even if there is not yet a running code. Instead of testing an implemented system ADAM tests a model of the system.

Penetration testing is a common means to evaluate implemented security controls [6]. However, penetration testing usually only takes place in the late phases of software development, when it is already very expensive in comparison to security fixes in the early phases of software development. Also, the effectiveness and efficiency of penetration tests depend on the skills of the tester [6]. Vulnerabilities could go unnoticed, hence the coverage of penetration tests is unclear.

In contrast, a holistic attack model that provides automatable mechanisms for generating and simulating an adversary's attacking procedure could be applied in the early design phase. For example, in the automotive domain in contrast to other domains like web-application programming, executable system models are already used in the engineering process (e.g., for simulations of complex assistance systems [2]). ADAM can reuse these models. Hence, it mitigates some of the shortcomings of penetration testing by applying the idea of penetration testing already at model level. The automatable test execution is more cost-efficient, and weaknesses can be

This work is supported by the German Federal Ministry of Education and Research (BMBF) under the KMUinnovative program (Grant No. 16KIS0946).

detected earlier than in common penetration testing. However, our approach is a complement for penetration tests. The execution of attacks on models does not replace a necessary penetration test on the implemented system in later phases. In summary, the contribution by ADAM is

- Foundation for early, automatable security testing on model level (adversary-driven, step-by-step attempts to reach a specific goal) before running code is available.
- Domain-agnostic, holistic attack information basis supporting automatable security testing on model level.
- Potential for reuse of the framework elements, e.g., modelled attacks.

The primary focus of this extended paper is to provide more details on the necessary components of the ADAM framework.

The rest of this paper is structured as follows: Section II discusses related work on attack modelling. Section III states the requirements for the ADAM framework as a holistic attack modelling framework for security testing. Section IV presents our approach to attack modelling. This is followed by the presentation of the four main components of ADAM, adversary model in Section V, target model in Section VI, attack base in Section VII and the attack modelling from the process perspective in Section VIII. Section IX shows the preliminary evaluation of the ADAM framework. Section X concludes the paper.

II. Related Work

Several adversary and attack models exist. Depending on the perspective of the attack, there are various modelling concepts [8].

The process modelling approach focuses on representing the attack based on phases. For example, the Lockheed Martin Cyber Kill Chain [9] defines an attack with seven phases that have to be passed through by the adversary. The kill chain model intends to model advanced persistent threats and malware behaviour. Hence, an attack is seen as a linear process, and it does not represent information about the attack surface that is provided for an adversary. Testing requires exploring multiple attack techniques, so bare process modelling approaches are typically not sufficient for testing. Another standard method is graph-based modelling that uses attack graphs to represent various attack opportunities. Kaynar [10] presents examples of this class of adversary and attack models in the domain network security.

A specific graph representation of attacks is the attack tree [5]. An attack tree focuses on the primary goal of an adversary. This primary objective represents the root of the attack tree, the elementary attack steps to are the leaves, and the various associated subgoals link these nodes. Existing attack trees can easily be reused or combined to form more comprehensive attack trees for threat and risk analysis. Attack trees incorporate multiple paths adversaries may take, but they do not include any characteristics of an adversary or about an adversary's decision on the next steps in an attack. Efficient testing requires an approach that also takes into consideration realistic assumptions about attack paths. Our work uses tree structures in combination with adversary modelling and target modelling to overcome the shortcomings of attack trees.

Classification modelling approaches model attacks on different abstraction levels. For example, the MITRE Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) framework [4] enables attack modelling based on the adversary's perspective. Tactics, techniques, and procedures define adversary behaviour. MITRE ATT&CK can be used both to derive behaviour-based adversary scenarios and to establish relevant adversary profiles for an implemented system. It is suitable for testing and verifying the security of an existing software product. However, the MITRE ATT&CK framework is not designed for use in the early design phase to support model-based security testing based on a specific adversary strategy. Our work closes this gap.

However, some domains have specific requirements, that have to be met, e.g., in the automotive domain. Thus, [11] shows another classification modelling approach regarding attacks. Ponikwar et al. [11] focus on defining realistic adversaries for vehicular networking applications. Defined adversary characteristics provide a template that can be used to categorise adversaries. In the context of various attack scenarios, adversaries with different levels of strength can be considered, depending on the specific, targeted vehicular networking applications. The adversary profiles in [11] can be easily reused. However, it is designed as a foundation for threat and risk analysis, and for making implementation decisions of security controls, rather than to support the security testing process. Thus, ADAM intends to take advantage of an adversary characteristic, focusing on the rational behaviour of adversaries in keeping with a goal-oriented security testing procedure.

Another considerable approach regarding classification shows [8]. Sommer, Dürrwang, and Kriesten [8] define a uniform taxonomy to describe automotive security attacks by classification. However, the attacks are just descriptions (tabular form, listing) by the taxonomy, which are not linked to a model of the targeted system for automatable security testing. Sommer, Dürrwang, and Kriesten already compare attacks with successful penetration tests that can consist of several steps. The consideration of attacks into its steps has the main advantage that the attack's detailed information is taken into account, e.g., to recombine steps for unknown attack paths. This requires a collection of possible attack paths. However, [8] provides no method to find attack paths or to recombine steps for new paths. ADAM aims to close this gap. It considers the adversary's decision-making, including his opportunities for achieving the prime goal, that results in the identification of attack paths. Sommer, Dürrwang, and Kriesten mention that in further work, attack paths could be automatically generated based on this taxonomy. This is also the intention of ADAM, where the model-based approach (not limited to the automotive domain) is followed instead of representing attacks as a description (tabular form, listing).

There are also combined approaches to attack aspects shown above. Adepu and Mathur [3] present unified adversary and

attack models with a focus on both security and safety aspects in the context of cyber physical systems. The relevant system information is part of an attack domain model. However, Adepu and Mathur limit the proposed framework by not considering the characterisation of an adversary, e.g., the adversary's current knowledge about the target. However, realistic assumptions about an adversary are necessary for comprehensible modelling of the strategic and tactical attack actions of the adversary.

The Security Abstraction Model (SAM) [7] and ADversary VIew Security Evaluation (ADVISE) [12][13] are the works most similar to our approach.

Zoppelt and Kolagari [7] provide SAM, a foundation for the early analysis and design phases of software architecture development. It focuses on the integration of a security model into the common system model of the automotive domain, under consideration the associated security challenges. SAM presents a structured approach to identify and categorise attacks. The resulting overview of the overall attack situation assists the collaboration of various experts in the automotive domain with deriving security requirements and with the decision-making of security controls. However, SAM supports security by design and does not focus on automatable, early security testing by considering the strategical adversary's perspective of attack-decision-making. ADAM intends to automate early security testing (adversary-driven, step-by-step).

The protection of a software architecture always requires compromises for defenders. Not only defenders but also purposeful adversaries trade effort against benefit. Hence, ADVISE addresses the structured and goal-oriented procedure of an adversary [12][13]. This method is based on an executable security model on the system level to generate security metrics. It can already be applied in the design process of a system. The application of ADVISE is neither limited to a specific domain nor a certain level of detail. ADVISE is proposed for repeatable usage in security engineering to support the evaluation of system security. However, this security analysis method is not designed to automate security testing on model level based on a specific adversary. The adversary's decision function of ADVISE for the simulated attack procedure does not include the different aspects of designing and launching an attack, e.g., reconnaissance actions. In contrast, the main aspect of ADAM is the adversary-driven approach that drives the attack modelling step-by-step, including actions to launch an attack for the automation of security testing on model level.

III. Requirements

We propose ADAM, a concept for holistic attack modelling to support the model-based security testing by simulating the strategic actions of an adversary in terms of traceability. The general basis for the requirements engineering is [14] by interpreting security testing or attacking as business processes. Concerning the modelling of dynamic behaviour, there are analogies between model-based testing, attack strategies, and tactics and models for business processes [8][14][15]. Using models, complex scenarios can be simulated. ADAM is intended to be used to decide on the next steps during attacking activities that can be interpreted as test steps, e.g., modelling the structured use of existing hacking tools (or penetration testing tools from the tester's point of view). Hence, relevant requirements for the design of our approach can be derived from [14]:

a) Model-based: The expectation is that applying a model-based perspective to an attack presents a suitable basis for formalisation similar to the formalisation of the software development process in IT that came with the introduction of model-based software engineering [16].
b) Expressive: The purpose is to model as many attacks as possible by ADAM. A generic attack modelling framework should express all necessary information regarding attack, adversary, and target. As already shown in Section II, most attack models only incorporate certain aspects of an adversary and the target. The area of application is a relevant factor for the choice of an attack model. Using ADAM, this holistic attack model for multiple use cases can involve less effort than the application of several different attack modelling techniques, and it offers widespread usage.
c) Reusable: ADAM should consist of reusable components to reduce the time-consuming modelling of new attacks. For example, already modelled elements of ADAM should be reusable for as many different use cases as possible (e.g., change of target, adversary, or attack). The requirements a) model-based and b) expressive support this requirement reusable.
d) Systematic: The proposed ADAM framework should ensure a systematic and continuous (re-)use of attack information in as many phases as possible of the software engineering process. From the experience of the authors, today's software development often lacks such a systematic and continuous reuse of information about attacks.
e) Consistent: ADAM should be consistent. A consistent model can be verified and validated. Formalisation and automated tool support require a consistent model.
f) Visualisable: ADAM should use visual means to model attacks. An appropriate visual graphic representation of attacks facilitates readability and understandability, especially of complex attacks. Visual illustrations are more intuitive for humans than prose text [15]. The use of visual elements supports the formalisation as it is missing the ambiguity and inaccuracy of prose. The aim is to achieve a concise expressiveness of ADAM. The connection of the individual attack modelling components in ADAM should be easily identifiable, such that security can be consistently verified and software quality increases.
g) Understandable: Software engineers that are no security experts should be able to use ADAM throughout the software development process. Hence, ADAM should be understandable, easy to learn and uncomplicated to use. Complex models tend to be difficult to understand [16]. This disadvantage should be avoided.

IV. Design of the ADAM framework

The ADAM framework provides an adversary-based and target-based foundation to automate security testing on model level, including test case generation. Aspects of the adversary as well as aspects of the target under attack affect the attack path. Accordingly, these aspects will be considered for a holistic attack modelling to drive the security testing on the model level. We postulate the following scope for this work. Future work will probably leverage some of these restrictions:

- The ADAM framework focuses on attacks on modelled systems. Therefore, an adversary can theoretically influence any software-enabled technology in various ways [17]. Attacks targeting humans (e.g., Social Engineering) are out of scope.
- Our model is limited to adversaries that follow a rational goal [11]. Random attacks are out of the scope of this work. The ADAM framework is limited to goal-oriented adversaries.
- The attack modelling is limited to the information that is defined in an attack base. Therefore, no exploits can be considered in the systematic attack modelling by ADAM, which are not included in the attack base. But by updates of a suitable attack base, the Zero-Day exploits can be quickly available for security tests.
- The focus of this paper is to identify the necessary conceptional elements for the suitable, holistic ADAM framework and giving an idea of how it can be used. Completeness, detailed specification and implementation of these elements are out of the scope of this paper.

A. Overview

By means of ADAM, we associate each attack with an adversary and the system under attack (target). An adversary plans, develops, and executes attacks against the target by using specific resources. The adversary attempts actions step-by-step in a certain order to reach the primary adversary goal. The target may provide one or more access points (AcPs). An AcP is a point of the target, that provides the adversary with the opportunity of executing attacks [8][18] (see Section VI). After each step, the adversary can gain new information about the target and expands his perspective in this regard. He has certain AcPs at his disposal. Different aspects, such as his skills (e.g., hacking skills), knowledge of the target domain (e.g., the ISO 26021 [19] for road vehicles), or his connection to the target (e.g., only remote) reduce the selection of available AcPs for the current attack attempt. Both for the initialisation of the ADAM framework (see the "for" section of the loop in Figure 1) and its application (see the "while" and "do" section of the loop in Figure 1), it requires this basis of the content in the context of attacks.

Figure 1 gives an overview of the ADAM framework. The framework consists of the adversary model, target model, and attack model. The adversary model characterises a specific adversary. Each adversary is defined by a set of descriptive attributes, the goal of his attack, and his current perspective of the target (called adversary perspective model in Figure 1). Based on [13], a modelled adversary is called adversary profile, consisting of the defined goal, initial perspective and determined characteristic. The target model simulates the system under attack and all necessary associated components of the environment to simulate the attack path of the adversary. Based on [8], the attack model is called attack base and includes the technical aspects, potential ways and means to attack, e.g., the possible actions with the associated target.

ADAM links all the necessary information of these models to consider the attack from a process perspective (see "Attack Modelling" in Figure 1). At the beginning (see "Initialise Elements" in Figure 1), the necessary elements (adversary model, attack model and target model) have to be initialised. To achieve the prime adversary goal, the so-called elementary attack iteration (EAI) is iteratively called in the attack simulation. The EAI consists of defined steps. With each iteration, the adversary must decide on which action to attempt next that is goal-directed. For example, depending on the current adversary's perspective, and skills, he attempts to exploit the target by an available AcP.

For this purpose, all necessary information from the attack base and adversary profile is used. The attack simulation on the target model provides the effect of the attack on the target. The use of a target model allows executing attacks on systems that do not yet exist. Each iteration step ends with an update of the adversary profile. For example, when the adversary reached his primary goal, the simulation terminates. Otherwise, see "Attack continues" in Figure 1, e.g., the next iteration starts. The outcome of the ADAM framework in the context of an attack simulation is an attack path. Based on [8][20], an attack path is one action or a set of sequential actions taken by an adversary for the adversary's goal achievement. In the simplest case, the adversary can theoretically achieve the prime goal by selecting and executing one action, i.e., the attack path consists of one action. Otherwise, the adversary's goal achievement is composed of several actions, i.e., the attack path is a sequence of several actions in a certain order.

ADAM can be used for model-based security testing. It provides a structured approach to simulate the strategical behaviour of a given adversary to attack a particular target. It takes into consideration the properties of the adversary as well as the perspective the adversary has of the target at a given time. The systematic attack modelling process of ADAM identifies actions that can be interpreted as test steps on model level [8]. Thus, ADAM provides a basis to automate security testing on model level, including test case generation. Hence, ADAM supports holistic model-based security testing in the early phases of the software development process (no running code available) [1]. Besides, e.g., the generated attack path by ADAM can support as guidance for tests in later phases of the software development lifecycle (running code available) [1].

V. Adversary Model

The adversary model provides the goal-oriented adversary for attack modelling. In the context of ADAM, the following

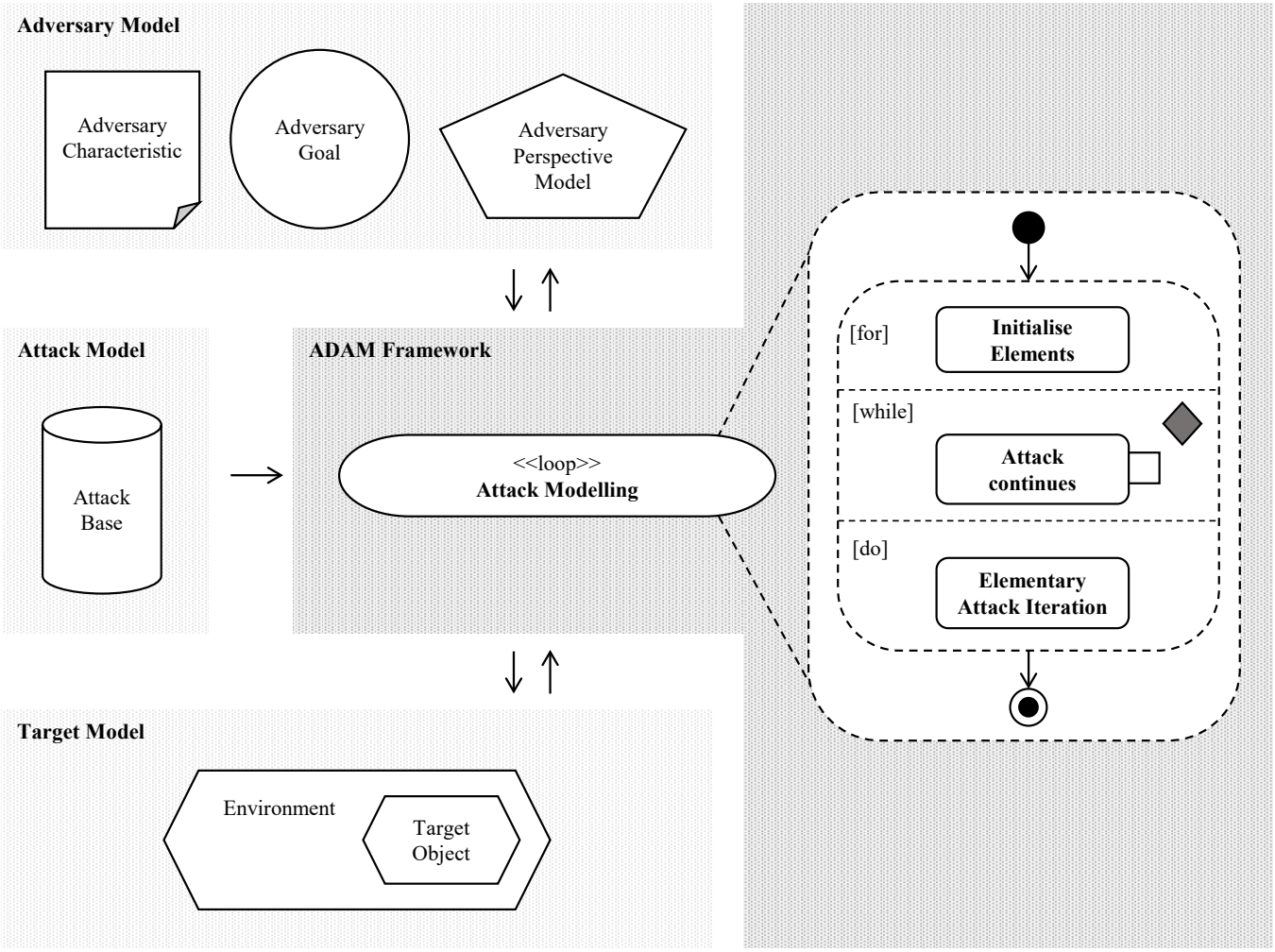


Fig. 1. Conceptual context of the ADAM framework based on [1].

requirements arise for the adversary model:

- An adversary should be modelled using defined, qualitative characteristics so that various realistic adversaries can be considered, independent of the target domain.
- The adversary model should state the prime goal of an adversary. The adversary goal is necessary to decide if the attack modelling is finished because the adversary goal is achieved. It is used for modelling the goal-oriented decision-making of a given adversary.
- The adversary model should represent the dynamic adversary's perspective of the targeted system, i.e., all information the adversary currently knows about the target that can be used to attack. This consideration enables representing an attack step-by-step, based on the adversary's preliminary, changing view of the target.
- A modelled adversary should be used for the process of attack modelling to generate the attack path.

Therefore, the adversary model consists of the three elements described in the following: adversary characteristic, adversary goal and the adversary perspective model [1] (see Figure 1).

A. Adversary Characteristic

The adversary is characterised by static attributes. This means, that the characteristic is initialised at the setup of the attack modelling (see "Initialise Elements" in Figure 1) and is not changed during its further use in modelling. Changing the characteristic means defining a different (type of) adversary for another simulation scenario. The total attributes represent the power of the modelled adversary and affect the adversary's attack path. It is assumed, e.g., an adversary with the expertise of general computer sciences, as the only skill, will search alternatives to actions of sophisticated hacking, unlike an adversary with professional security and hacking skills.

Figure 2 shows an exemplary collection of attributes for the adversary characteristic in support of ADAM. It can be expanded by arbitrary, which become necessary for the adversary's decision-making in the attack modelling regarding of future work, e.g., with the application of ADAM in further domains.

A unique ID (see characteristicID in Figure 2) is necessary to provide the individual identification of the modelled adver-

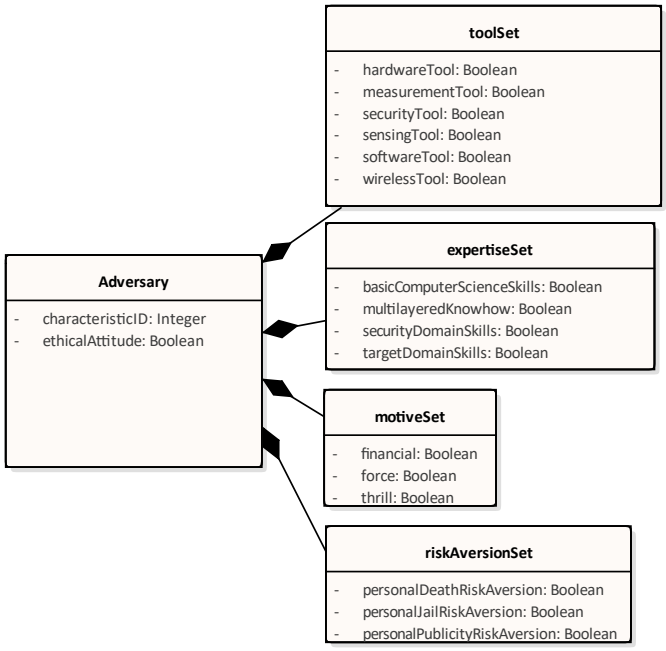


Fig. 2. Attributes for the Adversary Characteristic based on [8][12][13][21].

sary characteristic and thus its deliberate reuse in combination with another prime goal or initial adversary's perspective about the target.

The attribute ethicalAttitude in Figure 2 defines whether the adversary rejects personal injury as a matter of principle on the way to achieving his ultimate goal. For example, it is assumed that a penetration tester or hobbyist will not take actions that will clearly harm human lives to achieve their prime goal. Instead, these adversaries will seek other alternatives or give up. Thus, ethicalAttribute can be used for the adversary's decision-making on the next action of attacking out of available actions.

The adversary's power depends on tools. The adversary is able to choose the attacking actions associated with them. Adapted from [8] we use the attributes securityTool, softwareTool, hardwareTool, sensingTool, measurementTool and wirelessTool (see Figure 2). The attribute securityTool defines whether the adversary owns security tools, e.g., a reverse engineering tool to be able to choose information gathering actions. The attribute hardwareTool defines whether the adversary owns hardware tools, e.g., a laptop to be able to collaborate with different other tools. The attribute softwareTool defines whether the adversary owns software tools, e.g., a debugger to be able to choose information gathering actions. The attribute wirelessTool defines whether the adversary owns wireless tools, e.g., a cellular tool to be able to take remote actions. The attribute sensingTool defines whether the adversary owns sensing tools, e.g., a radar tool to provide input for systems that expect radar data. The attribute measurementTool defines whether the adversary owns measurement tools, e.g., a logic analyser to be able to choose information gathering actions. This insight shows that different attacking actions require different tools.

In addition to the consideration of tools, an adversary can be endowed with helpful skills. A distinction is made between the adversary's skills (as a statically determined resource) and the current perspective of the target that the adversary (dynamically) expands during the attack until the achievement of the adversary's goal. Based on [12][13], we distinguish the skills between basicComputerScienceSkills, securityDomainSkills, targetDomainSkills and the attribute multilayeredKnowhow (see Figure 2). By means of these attributes, the adversary's capabilities are determined, which affects his attack path. For example, just because the targeted system provides the opportunity of, e.g., actions regarding timing attacks, an adversary without the necessary skills will take alternative actions. The attribute basicComputerScienceSkills defines whether the adversary has basic expertise in computer science, for example, expertise in the internet protocol family. The attribute securityDomainSkills defines whether the adversary has specific expertise in the domain of security and hacking, e.g., as a professional penetration tester [6]. The attribute targetDomainSkills defines whether the adversary has specific expertise in the target domain, for example, in the interface technology of controller area network (CAN) in the automotive domain [8]. Besides the various areas of expertise, the attribute multilayeredKnowhow defines, whether the adversary has extensive knowledge on various topics that can be required for the achievement of time-consuming or complex attacks [5][11]. For example, a group of experts of different topics can be defined as an adversary compared to an individual adversary who is limited to its skills. It is used to distinguish organised groups with different experts as an adversary in ADAM to less broadly based groups or individuals. It is assumed that organised groups, like advanced persistent threat groups, have access to a wide range of different experts and are well networked with each other in the group.

The reason for a particular behaviour is determined by the motive [21]. The adversary's motives affect his decisions during an attack, independent of the current prime goal. Based on [8], and adjusted for the adversary model, the motives of the adversary are defined by the following set of attributes: force, thrill and financial (see Figure 2). The attribute force defines whether the adversary's willingness to act is driven by the purpose of influencing, e.g., as a state actor [22]. The attribute thrill defines that the adversary's willingness to act is based on thrill, e.g., as a hobbyist or script kiddie [22]. The attribute financial defines whether the adversary has financial reasons that purposefully affect his willingness to act, e.g., as a criminal, or as a paid professional penetration tester [22].

The adversary's decisions are determined by his risk aversion [5]. Each adversary is willing to take a certain risk to achieve the individual ultimate goal. During the ongoing attack, in step 3 (see Section VIII) of the EAI, the adversary has to choose one action out of diverse options, whereby each action has a specific risk. If an action is associated with

a risk that the adversary has an aversion to, it is assumed that the adversary prefers alternative actions or surrenders. Based on [5], the set of attributes for risk aversion consists of personalPublicityRiskAversion, personalJailRiskAversion and personalDeathRiskAversion (see Figure 2). The attribute personalPublicityRiskAversion defines whether the adversary has an aversion to the risk of publicity, e.g., a current employee [22] of the business of the targeted object (not to be confused with an angry or former employee). The attribute personalJailRiskAversion defines whether the adversary has an aversion to the risk of jail time, e.g., a hobbyist [22]. The attribute personalDeathRiskAversion defines whether the adversary has an aversion to the risk of death, e.g., an activist [22] who wants to call attention to a grievance.

This collection of attributes in Figure 2 exemplifies that various realistic adversaries can be presented using the adversary characteristic in the ADAM framework. The adversary can be characterised independently of the specific attack goal, and the current adversary's perspective of the target. Once an adversary is modelled by these attributes, the adversary characteristic can be reused in the context of, e.g., different adversary goals, initial adversary's perspective, targets, or new attack possibilities.

B. Adversary Goal

Another component of adversary modelling is the adversary goal (see Figure 1). The adversary goal is the primary goal to decide if the attack modelling is finished because this goal is achieved. It is used for modelling the goal-oriented decision-making of a given adversary. For example, the primary goal of an adversary may be the gain of financial data from a financial data transfer system. As well as the adversary characteristic, the adversary goal is static. This means, that the ultimate goal is initialised at the setup of ADAM (see "Initialise Elements" in Figure 1) and is not changed during its further use in the ADAM framework.

The adversary goal determines "what" is the attack goal. "How" the adversary goal is attempted to be achieved, by a certain sequence of actions (attack path) is only apparent from the application of ADAM (see attack modelling, Section VIII). Hence, the adversary's goal is used to derive the attack path of an adversary during an attack simulation. In the example above, the goal derives attractive data stores as target objects. The adversary tries to navigate from any AcPs available to the adversary to these data stores.

The following attributes illustrate an exemplary adversary goal modelling in support of the ADAM framework. A unique ID (goalID) is necessary to provide the individual identification of the modelled adversary goal and thus its deliberately reuse in combination with another adversary characteristic or initial adversary's perspective about the target. The attribute goalTarget defines the concrete target object of the adversary goal, e.g., a certain data set, data storage, interface, sub-function, function, or a complete system [8]. The attribute goalMotivation defines what is to be achieved concerning the goalTarget [8]. The goalMotivation always refers to the goalTarget. Based on [23], the goalMotivation represents the urge to act to achieve something by choosing useful actions. The following motivations can be distinguished to specify the adversary goal:

- Gaining (regarding information, financial or material) [7][8][24]
- Obtaining access [24]
- Affecting
- Controlling (in sense of the wittingly and goal-oriented utilisation)
- Modification or tuning (in sense of optimisation) [7][8]
- Damaging to property [8][24]

The attribute goalUndercover defines whether the adversary goal is associated with an as covert and unnoticed approach as possible [25]. It is assumed, e.g., to achieve the prime goal with a focus on industrial espionage, an adversary prefers to act as unnoticed and covertly as possible, i.e., the focus during the attack is also on actions to cover traces. The attribute goalAggressive defines whether the adversary goal is associated with an aggressive approach. For example, to achieve the ultimate goal with a focus on only finding vulnerabilities of a certain system (i.e., being not aggressive), an adversary prefers to act very carefully, not to cause any damage to the system. This scenario can be assigned to a typical penetration testing use case [25]. Whereas it is assumed, that, e.g., to achieve a prime goal with a focus on causing as much damage as possible (i.e., being aggressive), an adversary prefers choosing actions that cause harmful consequences [11].

These goal attributes exemplify how the adversary goal can be represented for the ADAM framework. The modelled adversary goal can be (re-)used in the context of, e.g., different adversary characteristics, adversary's perspective, targets, or new attack possibilities. The goal drives the step-by-step decision-making of a given adversary in ADAM and thus also affects the test case generation for model-based security testing.

C. Adversary Perspective Model

The third component of the adversary model for the ADAM framework is the adversary perspective model. It represents the adversary's state in the context of the progressing attack at a given time [12][13], from the beginning to the end of the attack simulation.

At the beginning of the attack modelling, the adversary perspective model shows the adversary's initial state of knowledge of the targeted object and its environment, including AcPs. An adversary can have already gained knowledge, e.g., from public sources, such as forums, manuals, standards, or as a result of previously executed attacks. For example, regarding the automotive domain, an adversary has already remote access to the CAN bus on a vehicle over cellular AcPs, which is demonstrated in [26]. Which initial adversary perspective is defined depends on what is the focus of the user of ADAM. If a tester uses ADAM to test a single function within a system, it can be assumed that the modelled adversary has already access to the system (defined as initial adversary perspective),

in contrast to the test case where an entire system is to be attacked. Thus, ADAM can influence the security test case generation on model level with the help of the adversary perspective model. The possibility of individually defining the initial adversary situation enables a flexible setting of the focus of the attack modelling. For example, if the adversary already has remote access to the victim's laptop, the focus is on actions regarding access on credentials instead of actions to gain access on the laptop anyway.

During the attack simulation, each EAI increases the adversary's knowledge, thus updating the adversary perspective model. For example, the adversary may get access to further AcPs after the first attack iteration, which he can use for an attack attempt in the next attack iteration. As long as the adversary's current knowledge is not sufficient to achieve his primary goal, the adversary tries to expand his knowledge in the appropriate direction through further attack attempts. This means that not only the adversary's initial perspective (by the adversary perspective model) but also each successive state during the attack attempt is relevant for attack modelling with ADAM. The adversary has a certain perspective of the target at a given time. Using the adversary perspective model, the results of the attack by the adversary is logged. Regardless of whether the simulated action was successful for the adversary or not, each result is a new takeaway. Hence, the initial perspective of the adversary is updated with each selected and simulated action during the attack simulation. It is assumed, that the adversary may get a wrong idea of the target. For example, the adversary could be fooled by means of security controls of the target. Thus, the adversary perspective model may keep incomplete or blurry details on the target. It represents the current, preliminary view an adversary usually has on the target. In contrast to the adversary perspective model, the target model (see Section VI) holds only correct information.

The adversary perspective model is necessary for the adversary-driven procedure of the step-by-step attack modelling. Hence, ADAM provides traceable attack attempts for the support of security testing automation on model level.

VI. Target Model

During an attack simulation with the ADAM framework, an adversary has to decide his next action for the achievement of his ultimate goal. Once the adversary has chosen an action, it is applied to the target model.

The target model represents the target, composed of both the target object and the environment of it, which an adversary wants to exploit and/or utilise for his attack attempt to achieve the ultimate goal. The target object corresponds to the concrete object, with which an adversary wants to act according to his ultimate goal (see the attribute goalTarget of the adversary goal). For example, the target object in the target model can be a certain data set that the adversary wants to delete. Depending on what is to be modelled or tested with a defined adversary, the target object is, e.g., data set, interface, subfunction, function or a complete system. The environment contains all necessary information to simulate the generated attack path by the EAI to get down to the target object, e.g., by means of exploited AcPs. From the set of all existing interfaces of the target object and its environment under consideration, the adversary has particular points available at a certain time, called AcPs. An AcP, based on [8][18], is an available point to the modelled adversary at the current time and provides him unintended access or unintended information disclosure. The AcP is either the target object or part of the surrounding environment of the target object. During an attack iteration, an adversary can utilise available AcPs to achieve his prime goal.

This outcome of the applied action to the target model influence the adversary's profile. For example, the output of a simulated action is a credential that is accordingly included in the current adversary perspective model of ADAM.

The attack simulation on a target model enables to show the attack and its effect on the target on model level. This implies that attacks can be tested in the early phases of the development of a product (target) that is not yet implemented. The target model simulates the target object and all relevant aspects of the environment of the target object that can be used for the attack attempts by the adversary.

VII. Attack Base

The attack base contains all necessary technical details for the attack modelling, e.g., all actions that an adversary could possibly execute during the attack simulation. Various works (see Section II) exist with a lot of information about known attacks. For example, based on known, past attacks [27][28], or from associated reports and analysis [29][30], actions can be derived to fill the attack base.

An action is the elementary element of the attack from the technical perspective. Based on [8], the action represents an elementary attack (e.g., modify the calibration update) that can be one specific step of a composite attack (e.g., controlling a certain electronic control unit (ECU)). As an intermediate step of a more comprehensive attack, the step does not necessarily have to be a stand-alone, typical attack action. For example, using search engines can be an action of the composite attack reconnaissance.

Several actions can be pooled to an AcP, which is also provided as information in the attack base. E.g., based on [28], the on-board diagnostics (OBD) as AcP (of the automotive domain) is associated with the following actions:

- Action of capturing CAN messages
- Action of replay CAN message
- Action of sending the standard command for disabling the CAN communication
- Action of firmware extraction of the telematics ECU
- Action of sending the standard command for flashing the telematics ECU

The actions can be linked to each other utilising preconditions and postconditions [8][10]. The action of capturing CAN messages has preconditions, e.g., that access to the CAN bus is

given and captured CAN messages as a postcondition, which in turn acts as a precondition for further actions in this regard.

The attack base includes the technical aspects of attacks. For the holistic attack consideration, the handling of this foundation of technical attack information is described in Section VIII. The attack modelling loop is responsible for selecting specific actions, step-by-step, from all available actions in the attack base.

VIII. Attack Modelling

The ADAM framework provides various perspectives of an attack as shown in Figure 3.

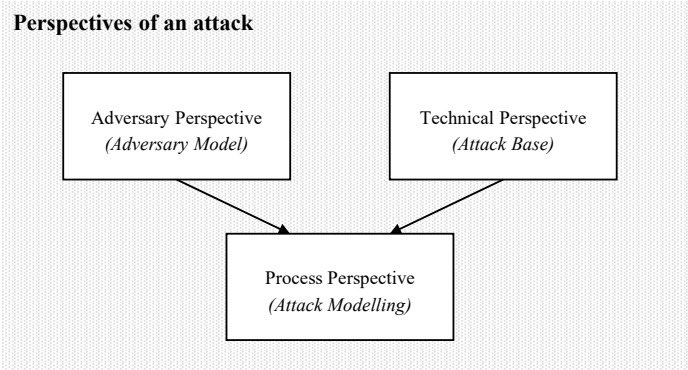


Fig. 3. Linking of the considered perspectives of an attack by the ADAM framework.

The adversary model provides the goal-oriented adversary perspective of attacks (see Section V). This perspective represents the scope of the attack modelling from the adversary's point of view, i.e., the adversary characteristic, the adversary goal and the adversary's current, dynamic knowledge about the target (adversary perspective model).

The technical perspective of attacks focuses on the actions provided by the attack base (see Section VII), which can lead to the adversary's goal achievement in proper order.

The process perspective focuses on the execution of an attack. By means of defined steps, the technical perspective of an attack is considered in a structured way, i.e., the attack modelling by the ADAM framework brings together both the goal-oriented adversary perspective of attacks and the technical perspective of attacks, as outlined in Figure 3.

The attack modelling loop simulates each attack. First, the necessary elements, i.e., the adversary model, attack model, and target model, have to be initialised (see "Initialise Elements" in Figure 1), which provide input for the attack modelling loop. As a result, a specific adversary profile, target model and attack base are available. Next, their content will be used in a structured way according to the attack modelling loop.

As long as the attack continues, for example, the adversary goal is not achieved and he does not surrender, the EAI will be called iteratively. We call such an iteration an elementary attack iteration (EAI), as shown in Figure 1, as it constitutes the smallest attack unit possible from a process perspective of attacks. Each EAI includes the five steps:

1) Identify available AcPs
2) Select an AcP
3) Select an action
4) Probe the target
5) Update the adversary profile

Within each iteration, the modelled adversary decides of which action to attempt next for achieving his primary goal. Thus, the adversary first chooses an AcP over which to execute the attack attempt. For this purpose, e.g., information about his current perspective of the target object and its environment is used (provided by the adversary perspective model). In addition to the prerequisites concerning the target (from the adversary's perspective), aspects regarding the adversary (characteristic) can also be presupposed, e.g., necessary tools and skills (see Subsection V-A) to select a certain AcP, and next, a specific action.

For example, the adversary wants to modify a function in a vehicle (adversary goal). Thus, he selects the AcP OBD connector because he has the necessary skills and tools for the automotive target domain and the OBD connector is target-aimed to the adversary goal. From all available actions associated with the AcP, the adversary chooses one goal-directed action based on the adversary's profile. In the example above, the adversary selects an action of code injection, because the adversary already knows, the OBD connector is active and vulnerable, and actions of override did not work (e.g., as results of a previous attack action, represented in the adversary perspective model) and the action is in compliance with the adversary characteristic. For example, he has targetDomainSkills and basicComputerScienceSkills as appropriate skills, and, e.g., due to the acceptance of the risk of death, the adversary selects and executes an action of code injection. The chosen action will be applied to the target model that provides the effect of the (attack) action on model level. Finally, the adversary profile will be updated, e.g., the adversary perspective will be expanded by the result of the action (successful, or not) and, as appropriate, by the recent AcPs that has been made available for the next EAI. To achieve the adversary's primary goal, usually, several elementary iterations are necessary. The chosen actions by the adversary profile during the attack modelling, by the sequential selection, provide the attack path as the output of the applied ADAM framework. These selected actions can be interpreted as test steps on model level [8]. Hence, ADAM supports not only a foundation to automate model-based security tests but the whole security testing on model level, including test case generation. Besides, the generated attack paths can give hints that are worth considering in other tests, like penetration tests in later phases of the software development process.

IX. Preliminary Evaluation

In this section, we evaluate if the ADAM framework meets the requirements of Section III and we give an idea of how ADAM can be integrated to support security testing. We

evaluated ADAM under the following restrictions (future work will leverage some of these restrictions):

- The application of the modelling is limited in each case to one attack iteration.
- The attack scenarios under consideration focus on the first actions of an attack, comparable to reconnaissance of the Lockheed Martin Cyber Kill Chain [9].
- The proposed ADAM framework is applied to two significant attack scenarios by way of example. The first example incorporates vulnerabilities of the Open Web Application Security Project (OWASP) Top Ten 2017 [31], hence is highly relevant in the domain of web application. In contrast, the second example stems from the automotive domain. We use the idea of UML class, object and activity diagrams [32] and attack tree [5] for our examples. We choose UML as it is common in many relevant application domains.
- The decision on which adversaries are relevant for modelling with the help of ADAM is not the focus of this paper. For the evaluation, we use the widespread method of threat and risk analysis in the context of the research project MASSiF. As a consequence, the relevant type of adversary is organised criminals.

A. Evaluation Criteria

The following criteria were identified for the evaluation:

1) Model-based: The criterion refers to the extent to which the ADAM framework is based on a model.
2) Relevant attacks: The criterion refers to the extent to which relevant attacks can be modelled using ADAM.
3) Application domain independence: The criterion refers to the ability to model different attacks independently of the application domain.
4) Reusable elements: The criterion refers to the extent to which the modelled contents and elements of ADAM can be easily reused in conjunction with other attack scenarios.
5) Systematic structures: The criterion refers to the extent to which there is a systematic approach to the structure and procedure of the proposed attack modelling concept so that an attack can be modelled in a comprehensible and repeatable way.
6) Visual elements: The criterion refers to the extent to which the ADAM framework has graphic elements or can be illustrated visually at a glance.

A consistent model is a requirement for the use of automatism [14] and thus, a suitable basis for supporting the automation of security testing on model level. Therefore, a detailed specification of the EAI, including a proper syntax and semantic for the necessary elements have to be defined. The specification of the individual elements of the proposed concept is out of the scope of this paper. Therefore, we omit the evaluation of the model consistency. Moreover, the specification of the EAI is required to make appropriate statements about the understandability. The understandability of a model helps to evaluate its usefulness. Also, we omit the evaluation of the requirement understandability. We will survey relevant stakeholders to assess the understandability of the model in the further course of the still running research project MASSiF.

B. Exemplary Application of ADAM

We iterated through the proposed ADAM framework, based on two exemplary attack scenarios, and integrated it in an exemplary way. For the sake of briefness, we only present an extract of exciting findings in the following Figure 6, Figure 8 and Figure 9, concerning the elementary attack step 3 of the EAI (see Section VIII).

In the first scenario, we model a criminal adversary who wants to steal an identity on a social media platform. This scenario incorporates attacks from the OWASP Top Ten 2017 [31]. The characteristic of the criminal is shown in Figure 4. The defined adversary accepts injury to people as a matter of principle on the way to achieving the ultimate goal (see "ethicalAttitude=false" in Figure 4). The adversary has an aversion to actions that are associated with a risk of the adversary's death (see "personalDeathRiskAversion=true" in Figure 4). For example, Ponikwar et al. [11] mention criminals who act goal-oriented, like a business. Hence, the criminal is assigned with the financial and force motive. He is well equipped with skills and tools. The adversary is defined as an organised criminal group, which is reflected in the attribute "multilayeredKnowhow=true" in Figure 4.

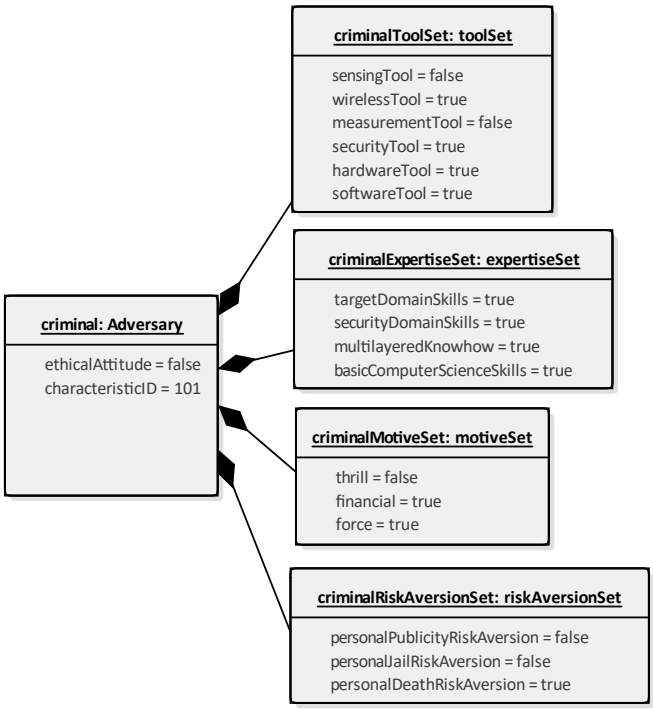


Fig. 4. Characteristic of the modelled criminal as an adversary.

Figure 5 describes the goal of the criminal for the first scenario. In this scenario, the criminal adversary not only

wants to gain identity information but also aims at personating someone else on a social media platform. In doing so, the goal is linked to acting as unnoticed as possible and not destroying the function of the social media platform.

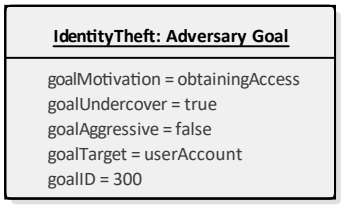


Fig. 5. Identity theft as an adversary goal.

The alternative actions for the criminal (regarding identity theft as the ultimate goal) from the AcP user input field of a web application are shown in Figure 6. Using tree structures, the result of the criminal's decision is visualised. The adversary chose an action in the context of Credential Stuffing.

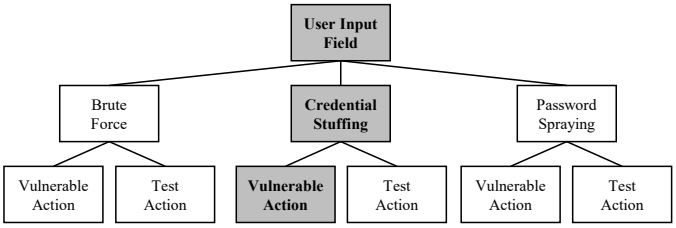


Fig. 6. Selected action based on the AcP "User Input Field".

The second scenario is taken from the research project MASSiF. The adversary characteristic from the first scenario is reused. In the second scenario, the defined criminal adversary with the characteristic as shown in Figure 4, attempts to disrupt the function of an ECU in a vehicle. In this use case, the "targetDomainSkills=true" refers to the automotive domain. For example, it is assumed the criminal knows the relevant standards of the automotive, such as the ISO 26021 ("Road vehicles – End-of-life activation of on-board pyrotechnic devices") [19]. For the goal achievement, the defined criminal could start an attempt by utilising actions based on his knowledge about the standards of automotive.

Figure 7 represents the adversary goal for the second scenario. The criminal wants to disturb an ECU functionality. Within the scope of this goal, the criminal does not care whether the victim notices the attack attempts (see "goalUndercover=false"). Attribute "goalAggressive=true" shows that the adversary accepts that the goal achievement may entail personal injuries.

Figure 8 illustrates the alternative actions for the criminal (with the prime goal of ECU functionality disturbance) from the standardised interface OBD connector in a vehicle. At the very beginning of the attack, the criminal needs information about the target. Thus, the adversary selected an action regarding extraction information using the OBD connector.

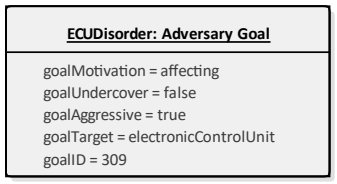


Fig. 7. ECU disturbance as adversary goal.

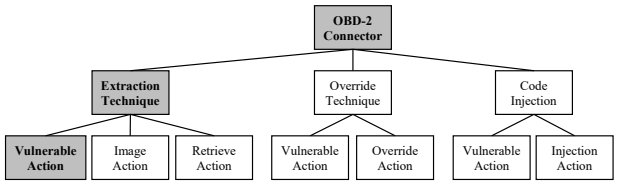


Fig. 8. Selected action based on the AcP "OBD connector".

The respective application of the actions in the scenarios leads to new information for the adversary, e.g., the specific AcP is vulnerable. During the next iteration, the adversary can select the next action based on the new information the adversary gained from the previous attack iteration.

In the context of the research project MASSiF, we integrated ADAM in an exemplary way, as shown in Figure 9. As MASSiF focuses on the automotive domain a model-in-the-loop (MiL) simulation environment is used to integrate. A MiL is common for safety testing of complex assistance systems in the early development phases [2]. In the MiL test, an executable model is the test object. The test object and an associated environment are integrated into a control loop to represent and test their interactions as realistically as possible [2].

The ADAM framework consists of the adversary and attack model, which are connected with the target model through the attack modelling loop (see Figure 1). The target model consists of the target object and its surrounding environment. Based on [15], a simulated environment model displays the boundary conditions and required interfaces of the environment for the system model. In this case, in Figure 9, the system model is the target object, e.g., the defined adversary wants to disturb a certain system, modelled as a system model.

The simulated environment provides required input data for the simulation of the system. For example, a system model processes the required input data and resends some output to another component in the simulated environment. But instead of a direct link between the simulated environment and the system model (target object), the data flows through the ADAM framework, as shown in Figure 9. During an EAI of the attack modelling loop, the defined adversary may or may not have certain possibilities (based on the adversary perspective model as well as his characteristic, e.g., skills) to take actions to affect the data flow from the simulated environment to the target object (system model) to achieve his ultimate attack goal. For example, the adversary chooses an action to replay manipulated sensor data to the system model (target object)

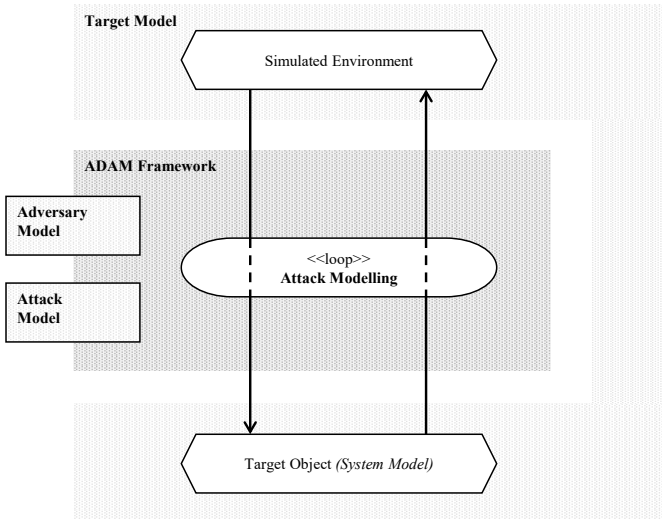


Fig. 9. Conceptional idea for the integration of the ADAM framework.

because he has already met the associated preconditions (such as successful access) as a result of previous actions. To get the result of the selected action, the data flow from the system model (target object) to the environment will be read out (as part of the EAI step 5 "Update the adversary profile", see Section VIII).

C. Interpretation and Discussion

Using the example of tree structures and UML class, object and activity diagrams, model-based elements can be used systematically for attack modelling. The criterion model-based can be confirmed insofar as ADAM provides a suitable foundation for different modelling approaches.

The criterion relevant attacks can be confirmed to the extent that we were able to model two representative examples from very different application domains. It is assumed, that the attacks have already been integrated as actions in the attack base to be considered using the ADAM framework. In our opinion, the proposed concept provides a suitable basis for modelling attacks, independent of the domain. Likewise, utilising the adversary model, different adversaries can be represented, for example, the adversary characteristic is extendable accordingly. Consequently, the ADAM framework accomplishes the criterion application domain independence. However, it is still an open question to what extent the specific characteristics of individual domains must or can be captured.

The defined actions of the attack base, as well as the EAI process itself, are exemplary representatives of reusable elements. Likewise, and regarding the criterion reusable elements, a modelled adversary (adversary profile) and its components (e.g., adversary goal), can be reused repeatedly and adapted, for example, with a different goal and/or initial adversary's perspective about the target for the attack modelling.

The EAI of the ADAM framework represents the basic, systematic, adversary-driven guideline for attack modelling from the process perspective of attacks. Likewise, the adversary model, target model and attack model represent a suitable foundation for a systematic deployment, representation and reuse of attack information. In this respect, the criterion systematic structures is accomplished.

The exemplary use of tree structures, UML activity, class and object diagrams shows that the attack modelling concept provides a suitable basis for the integration of graphical model elements. In this respect, the criterion of visual elements is accomplished.

The attributes of the adversary characteristic in their current form are not yet sufficient for the attack modelling by the ADAM framework. The Boolean expression, e.g., for the tool set and skill set of the adversary characteristic is not sufficient to filter out an action from a set of similar actions. For example, the actions of vulnerability scanning, fuzzing, and reverse engineering all require security tools. If securityTool (see adversary characteristic in Section V) is only expressed by a Boolean value, the filtering does not reduce the set of these available actions to one action. Another example is the attribute set of risk aversion (see adversary characteristic in Section V). For example, a dependency on the legal situation of the respective country is still important for choosing actions according to the personalJailRiskAversion.

The advantage of the conceptional idea in Figure 9 is that the integration of the ADAM framework can be used flexibly. In our integration example, the underlying MiL tests should be performed with or without the use of ADAM. If the ADAM framework is integrated, it passes the data onto the target model. It happens whether the modelled adversary took actions with this data or not. The data stream should be able to be passed through easily. Nevertheless, there can be a real-time requirement for the simulated environment regarding simulating safety-critical functions. Further work is required for this challenge.

We evaluated five out of seven requirements. In the context of the criteria, ADAM meets the requirements model-based, expressive, reusable, systematic, and visualisable. The requirements e) consistent and g) understandable can only be meaningfully evaluated in a later stage of the research project MASSiF. Hence, we did not evaluate these requirements.

It is being assumed, that further details and tighter definitions of the action, adversary goal, adversary characteristic and adversary perspective model are required because they are closely related to the specification of the EAI in future work. For example, a differentiated classification of the adversary characteristic values (compared to a simple Boolean consideration) may be necessary to enable a suitable cost-benefit analysis for implementing the adversary attacking decisions within the EAI.

By the integration of ADAM in the already existing work to model-based testing utilising environment model and system model [15], it is attempted to reduce effort and to use the already established knowledge. The extent to which domain-specific requirements can be met by the exemplary integration idea has to be considered in future work.

X. Conclusion and Future Work

In this extended paper, we present the ADAM framework for model-based security testing. The framework addresses security throughout the software engineering process. The main goal of ADAM is to provide a foundation to automate security testing in the early phases of software engineering (e.g., manual security reviews), which is the focus of this paper. Domains in which models are used at an early stage in software development will especially benefit from this approach. One example is the automotive industry, in which executable models are already used in the design phase, such as in MiL-methods for safety testing. ADAM can reuse these executable models. Besides, the results of ADAM can be used as a basis for assisting security testing activities in the later phases of the software development lifecycle, e.g., the generated attack paths can give hints that are worth considering in other tests like penetration tests. The central part of the ADAM framework is the attack modelling loop. During a loop, ADAM provides several perspectives on attacks. An attack is executed against a target simulation (target model). Using a system model as a part of the target model, in the context of the MiL-method, allows simulating attacks on software systems that are not implemented yet. The primary goal that the adversary wants to achieve drives the simulation and offers multiple paths of attacks. The preliminary evaluation of ADAM shows that the framework is expressive, reusable, systematic, visualisable and model-based. Future work will focus on detailed specification, implementation of the proposed elements, particularly the attack modelling loop, attack base and the testing part of the ADAM framework.

Acknowledgment

This work is part of the project "Modellbasierte Absicherung von Security und Safety für umfeldbasierte Fahrzeugfunktionen (MASSiF)". It is supported by the German Federal Ministry of Education and Research (BMBF) under the KMUinnovative program. Many thanks to our colleagues, Sanjana Biank and Matthias Meyer.

References

[1] T. Volkersdorfer and H.-J. Hof. "A Concept of an Attack Model for a Model-Based Security Testing Framework". In: *The Fourteenth International Conference on Emerging Security Information, Systems and Technologies Securware 2020* (Valencia, Spain, Nov. 21–25, 2020). IARIA, 2020, pp. 96–101. URL: https://www.thinkmind.org / articles / securware_2020_2_130_30046 . pdf (retrieved 11/18/2021).

[2] H. Winner, S. Hakuli, F. Lotz, and C. Singer. *Handbuch Fahrerassistenzsysteme. Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*. 3rd ed. Springer Verlag, 2015. Chap. 8, 33. DOI: 10.1007/978-3-658-05734-3.

[3] S. Adepu and A. Mathur. "Generalized Attacker and Attack Models for Cyber Physical Systems". In: *2016 IEEE 40th Annual Computer Software and Applications Conference. 10-14 June 2016, Atlanta, Georgia : proceedings*. 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC) (Atlanta, GA, USA, June 10–14, 2016). Ed. by S. Reisman. Piscataway, NJ: IEEE, 2016, pp. 283–292. ISBN: 978-1-4673-8845-0. DOI: 10.1109/COMPSAC.2016.122.

[4] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas. *MITRE ATT&CK. Design and Philosophy*. Ed. by The MITRE Corporation. The MITRE Corporation. 7515 Colshire Drive, McLean, VA 22102-7539, 2018. Chap. 1, 2, 3, 4. URL: https://www.mitre.org/sites/default/files/publications/pr-18-0944-11-mitre-attack-design-and-philosophy.pdf (retrieved 09/25/2021).

[5] B. Schneier. "Attack Trees". In: *Dr. Dobb's Journal* 24.12 (1999), pp. 21–29. URL: http://macs.citadel.edu/baniks/427/Homework/attacktrees.pdf (retrieved 10/18/2021).

[6] K. Scarfone, M. Souppaya, A. Cody, and A. Orebaugh. *Technnical Guide to Information Security Testing and Assessment*. Special Publication 800-115 800-115. Gaithersburg, MD 20899-8930: National Institue of Standards and Technology, Sept. 2008. URL: https://doi.org/10.6028/NIST.SP.800-115 (retrieved 10/05/2021).

[7] M. Zoppelt and R. T. Kolagari. "Reaching Grey Havens: Industrial Automotive Security Modeling with SAM". In: *International Journal on Advances in Security* 12.no. 3 & 4 (2019), pp. 223–235. ISSN: 1942-2636. URL: https://www.iariajournals.org/security/sec_v12_n34_2019_paged.pdf (retrieved 09/29/2021).

[8] F. Sommer, J. Dürrwang, and R. Kriesten. "Survey and Classification of Automotive Security Attacks". In: *Information* 10.4 (148 2019). URL: https://doi.org/10.3390/info10040148 (retrieved 11/15/2021).

[9] E. Hutchins, M. Cloppert, and R. Amin. "Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains". In: *Leading Issues in Information Warfare & Security Research* 1 (Jan. 2011). URL: https://www.researchgate.net/publication/266038451_Intelligence-Driven_Computer_Network_Defense_Informed_by_Analysis_of_Adversary_Campaigns_and_Intrusion_Kill_Chains (retrieved 04/01/2021).

[10] K. Kaynar. "A taxonomy for attack graph generation and usage in network security". In: *Journal of Information Security and Applications* 29 (2016), pp. 27–56.

ISSN: 2214-2126. DOI: https://doi.org/10.1016/j.jisa.2016.02.001.

[11] C. Ponikwar, H.-J. Hof, S. Gopinath, and L. Wischhof. "Beyond the Dolev-Yao Model: Realistic Application-Specific Attacker Models for Applications Using Vehicular Communication". In: *The Tenth International Conference on Emerging Security Information, Systems and Technologies (SECURWARE 2016)* (Nice, France). July 2016. URL: https://arxiv.org/abs/1607.08277v1 (retrieved 03/18/2020).

[12] E. LeMay, W. Unkenholz, D. Parks, C. Muehrcke, K. Keefe, and W. Sanders. "Adversary-Driven State-Based System Security Evaluation". In: *Proceedings of the 6th International Workshop on Security Measurements and Metrics* (Bolzano, Italy). MetriSec 10 Article 5. New York, NY, USA: Association for Computing Machinery, Oct. 2010, pp. 1–9. DOI: 10.1145/1853919.1853926.

[13] E. LeMay, M. D. Ford, K. Keefe, W. H. Sanders, and C. Muehrcke. "Model-based Security Metrics Using ADversary VIew Security Evaluation (ADVISE)". In: *2011 Eighth International Conference on Quantitative Evaluation of SysTems*. 2011, pp. 191–200. DOI: https://doi-org.thi.idm.oclc.org/10.1109/QEST.2011.34.

[14] A. Drescher, A. Koschmider, and A. Oberweis. *Modellierung und Analyse von Geschäftsprozessen. Grundlagen und Übungsaufgaben mit Lösungen*. ger. De Gruyter Studium. München, Wien: De Gruyter Oldenbourg, 2017. Chap. 2, 3. DOI: 10.1515/9783110494532.

[15] M. Winter, T. Roßner, C. Brandes, and H. Götz. *Basiswissen Modellbasierter Test. Aus- und Weiterbildung zum ISTQB Foundation Level - Certificated Model-based Tester*. 2nd ed. dpunkt.verlag GmbH, 2016. Chap. 3, 4. ISBN: 97833864902970.

[16] J. M. Borky and T. H. Bradley. *Effective Model-Based Systems Engineering*. Cham: Springer International Publishing, 2019. DOI: 10.1007/978-3-319-95669-5.

[17] The MITRE Corporation, ed. *CAPEC Glossary*. 2019. URL: https://capec.mitre.org/about/glossary.html (retrieved 06/10/2021).

[18] J. Bryans, L. S. Liew, H. N. Nguyen, G. Sabaliauskaite, S. Shaikh, and F. Zhou. "A Template-Based Method for the Generation of Attack Trees". In: *Information Security Theory and Practice*. IFIP International Federation for Information Processing 2020. Ed. by M. Laurent and T. Giannetsos. Cham: Springer International Publishing, 2020, pp. 155–165. DOI: 10.1007/978-3-030-41702-4_10.

[19] *Road vehicles - End-of-life activation of on-board pyrotechnic devices. Part 1: General information and use case definitions*. ISO 26021-2:2008(E). May 2008.

[20] S. Moskal, S. J. Yang, and M. E. Kuhl. "Cyber threat assessment via attack scenario simulation using an integrated adversary and network modeling approach". In: *The Journal of Defense Modeling and Simulation* 15.1 (2018), pp. 13–29. DOI: 10.1177/1548512917725408.

[21] G. W. Maier, F.-R. Esch, and M. Kirchgeorg. *Motiv*. Ed. by S.-V. GmbH. Feb. 16, 2018. URL: https://wirtschaftslexikon.gabler.de/definition/motiv-39694/version-263096 (retrieved 08/03/2021).

[22] Bundesamt für Sicherheit in der Informationstechnik - BSI, ed. *Register aktueller Cyber-Gefährdungen und -Angriffsformen*. Bonn, 2018. URL: https://www.allianz-fuer-cybersicherheit.de/SharedDocs/Downloads/Webs/ACS/DE/BSI-CS/BSI-CS_026.pdf?__blob=publicationFile&v=1 (retrieved 07/03/2021).

[23] G. W. Maier and M. Kirchgeorg. *Motivation*. Ed. by S.-V. GmbH. Feb. 19, 2018. URL: https://wirtschaftslexikon.gabler.de/definition/motivation-38456/version-261879 (retrieved 07/09/2021).

[24] J. D. Howard and T. A. Longstaff. *A Common Language for Computer Security Incidents*. Tech. rep. Oct. 1998. URL: https://www.osti.gov/biblio/751004 (retrieved 05/06/2021).

[25] *Durchführungskonzept für Penetrationstests*. Studie. Bonn: Bundesamt für Sicherheit in der Informationstechnik-BSI, 2003. URL: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/Penetrationstest/penetrationstest.pdf?__blob=publicationFile&v=2 (retrieved 09/23/2021).

[26] C. Miller and C. Valasek. "Remote Exploitation of an unaltered passenger vehicle". In: *Black Hat USA* (2015). URL: http://www.illmatics.com/Remote%20Car%20Hacking.pdf (retrieved 06/04/2021).

[27] The MITRE Corporation, ed. *CAPEC*. URL: https://capec.mitre.org/ (retrieved 08/11/2021).

[28] F. Sommer and J. Dürrwang. *IEEM-HsKA/AAD: Automotive Attack Database (AAD)*. 2019. URL: https://github.com/IEEM-HsKA/AAD (retrieved 06/24/2021).

[29] The MITRE Corporation, ed. *CWE*. URL: https://cwe.mitre.org/ (retrieved 08/11/2021).

[30] The MITRE Corporation, ed. *CVE*. URL: https://cve.mitre.org/index.html (retrieved 08/11/2021).

[31] The OWASP Foundation, ed. *OWASP Top 10 - 2017. The ten most critical web application security risks*. Bel Air, MD 21014 US, 2017. URL: https://github.com/OWASP/Top10/raw/master/2017/OWASP%20Top%2010-2017%20(en).pdf (retrieved 07/18/2021).

[32] Object Management Group, ed. *Unified Modeling Language*. 2.5.1. Needham, MA 02494, USA, 2017. Chap. 15, 16. URL: https://www.omg.org/spec/UML/2.5.1/PDF (retrieved 07/22/2021).

WAF Signature Generation from Real-Time Information on the Web using Similarity to CVE

1st Masahito Kumazaki
Graduate School of Informatics
Nagoya University
Nagoya, Japan
Email: kumazaki@net.itc.nagoya-u.ac.jp

2nd Yukiko Yamaguchi
Information Technology Center
Nagoya University
Nagoya, Japan
Email: yamaguchi@itc.nagoya-u.ac.jp

3rd Hajime Shimada
Information Technology Center
Nagoya University
Nagoya, Japan
Email: shimada@itc.nagoya-u.ac.jp

4th Hirokazu Hasegawa
Information Security Office
Nagoya University
Nagoya, Japan
Email: hasegawa@icts.nagoya-u.ac.jp

***Abstract*—Zero-day attacks and attacks based on publicly disclosed vulnerability information are major threats to network security. To cope with such attacks, it is important to collect related information and deal with vulnerabilities as soon as possible. We have developed a system that collects vulnerability information related to web applications from real-time open information on the web, such as that found on Twitter and other discussion-style web sites, and generates web application firewall (WAF) signatures for them. In this study, first, we collected vulnerability information containing a specified keyword from the National Vulnerability Database (NVD) data feed and generated WAF signatures automatically. Then, we examined the suitability of the WAF signature generation from one tweet. Finally, we extracted tweets that might contain vulnerability information and labeled them using a filtering algorithm. We then further experimented on gathering and extracting vulnerability information for a target web application. First, we gathered past Common Vulnerabilities and Exposures (CVE) descriptions of the target web application and used them to generate a Doc2Vec model and word vectors. We also used the trained Doc2Vec model to generate word vectors gathered from open information sources such as Twitter, Stack Overflow, and Security StackExchange. After that, we extracted vulnerability information for the target web application by calculating the cosine similarity between the word vectors of the open information and the CVE descriptions. Experimental results demonstrated that our Doc2Vec-based extraction process can be easily adapted to individual web applications.**

Keywords–Web Application Firewall(WAF); Zero-day Attack; Vulnerability Information; Real-time Information.

I. INTRODUCTION

We first review the results of the vulnerability information filtering method based on pattern matching that we introduced in our previous work presented in SECURWARE 2020 [1].

Web applications are recognized as an important part of the social infrastructure, and the average person uses a variety of such applications every day. At the same time, cyberattacks are increasing year by year and are now widely recognized as a significant threat to the social infrastructure. There are many cases of serious damage caused by such attacks, such as classified information leakage by unauthorized access and attacks that exploit vulnerabilities [2] [3].

Among cyberattacks, attacks that exploit published vulnerabilities are particularly on the increase [4] [5]. In a recent case, the number of detected attacks on Apache Struts 2 increased immediately after the announcement of its vulnerability [6], some of which resulted in personal information leakage due to a lack of necessary countermeasures such as timely system updates [7] [8]. Another major information security issue is zero-day attacks, which occur before the release of vulnerability information or the provision of patches [20]. Google Chrome suffered such a zero-day attack in 2019 [10].

The generally recommended countermeasure against such cyberattacks is to apply fixed patches distributed by the vendor in a timely manner. However, in the case of zero-day attacks, there is an unprotected period before the patch is released.

We previously developed a web application firewall (WAF) signature generation system using real-time information on the web to mitigate zero-day attack problems [1]. Real-time information sources such as Twitter are used for a variety of purposes [11] [12], and they may contain the latest vulnerabilities and emergency plans, which are useful for mitigating the problem before any formal vulnerability information or patches are released. Our system automatically collects vulnerability information to construct WAF signatures to mitigate the problems until the release of formal countermeasures. Although there are previous studies on the automatic generation of signatures for intrusion detection systems (IDSs) [13] [14] [15], we focus on WAF in this study because our target is web applications. To acquire the latest vulnerability information from the web, our system collects vulnerability information from real-time data feeds on social networking services (SNSs), web forums for security technologies discussion, and similar sites. After cleansing the collected data, the system checks for associated vulnerable web applications and generates WAF signatures for them.

In this paper, we demonstrate the effectiveness of our proposal on a practical level as an extension of the basic proof of concept provided in our previous work [1]. We first discuss the results of the vulnerability information filtering method based on pattern matching for multiple information sources. Further, as a method for generating an extraction model for individual web applications, we introduce a method using the similarity of word vectors generated by a Doc2Vec model trained with past Common Vulnerabilities and Exposures (CVE) descriptions.

To evaluate the WAF signature generation from the vulnerability information part, we performed an experiment with vulnerability information provided by the National Vulnerability Database (NVD) [16] and a specific tweet. First, we extracted the vulnerability information including a specified keyword related to the target web application and performed automatic generation of the WAF signature, as a proof of

concept. Then, we collected vulnerability information from Twitter, a well-known SNS and real-time information source, and attempted to generate WAF signatures from the tweets. Finally, we examined the extraction of tweets that included vulnerability information of the target web application by means of a pattern matching approach. The results of these investigations demonstrated the efficiency of the proposed system.

We next performed a practical experiment in which vulnerability information of the target web application was extracted using a text processing framework. First, we gathered past vulnerability information of the target web application from the description section of a CVE as reference data. We used these reference data to train a Doc2Vec model to extract vulnerability information words of the target web application and then utilized the trained model to generate reference word vectors with text data from the CVE description. Then, we collected vulnerability information from open information such as SNS (e.g., Twitter) and web forums for discussion on security technologies (e.g., Stack Overflow). We generated word vectors from the text part of the open information gathered with the trained Doc2Vec model and treated them as open information vectors. Finally, we performed similarity calculations between the reference vectors and the open information vectors. If an open information vector had a large similarity to the reference vectors, we treated the text correlated to the open information vector as containing vulnerability-related information.

In section II of this paper, we describe the background of our study, namely, the existing countermeasures. In section III, we present our proposed system and the architectures with which it is implemented. In section IV, we describe the experiments we performed to evaluate the implemented system. In section V, we discuss additional real-time information sources. Section VI presents the pattern matching-based extraction for target web applications and section VII shows the word vector similarity-based extraction with the Doc2Vec model trained using past CVE descriptions. We conclude in section VIII with a brief summary.

II. BACKGROUND

Existing countermeasures against zero-day attacks include defense-in-depth solutions that combine multiple security appliances such as firewalls, IDS/intrusion prevention system (IPS)-based allow/deny lists, and so on. However, if they are based on static rules, these countermeasures may result in an unprotected period against attacks. To mitigate the damage caused by zero-day attacks, we previously proposed a WAF signature generation system that uses real-time information on the web. Specifically, the proposed system generates a WAF signature for blocking access to a vulnerable web application when it discovers from real-time information on the Internet that the application contains vulnerability information. As a result, the unprotected time against attacks is shortened and the damage will ideally be mitigated.

A. WAF (Web Application Firewall)

Web applications are becoming more complicated year by year, and as such it is getting harder to detect vulnerabilities in their implementations. Therefore, WAFs are often used as a security measure to protect web applications from ingress traffic and to mitigate attacks that exploit vulnerabilities [17]. A WAF can be installed in multiple locations, such as a host type installed on a web server, a network type installed on a communication path to the web server, or a cloud type using WAF services on a cloud provided by the cloud service provider. By setting rules to prevent attacks aimed at typical web application vulnerabilities, we can protect the web server from attacks that target a specific vulnerability. Some WAFs enable users to set their own rules for specific attacks, which makes it possible to prevent zero-day attacks by applying custom signatures. In general, a WAF use the following basic functions to prevent external attacks and notify the administrator.

Analyzing
: Analyze Hypertext Transfer Protocol (HTTP) communication based on a detection pattern defined in an allow/deny list.

Processing
: Perform pass-through processing, error processing, replacement processing, blocking processing, and so on. Judgement is based on the result of the analysis function.

Logging
: Record WAF activity. An audit log records any detected unauthorized HTTP communication and its processing method. An operation log records WAF operation information and error information.

B. ModSecurity

ModSecurity is an open-source host type WAF software provided by Trustwave. It has the following functions.

- Recording and auditing of whole HTTP traffic
- Real-time monitoring of HTTP traffic
- Flexible enough rule engine to act as an external patch for web applications
- Can be embedded as a module of web server software Apache, IIS, and NGINX

Also, the Open web application Security Project (OWASP) [18] provides a Core Rule Set (CRS) that includes signatures for typical cyberattacks for ModSecurity. In the experiments later in this study, we use ModSecurity as the WAF for our proposed system to take advantage of the flexibility of the rule engine and the versatility of the module.

C. Doc2Vec

Doc2Vec is an advanced implementation of the Paragraph Vector proposed by Mikolov et al. [19]. It can vectorize a variety of different-length texts without losing the word order or semantics. However, with some text it is not preferable to keep the word order, so in the current Doc2Vec framework the user can choose either a mode that does not keep the word order, called the distributed memory model of paragraph vector (PV-DM), or a mode that does keep the word order, called the distributed bag-of-words paragraph vector (PV-DBOW). Before vectorizing text to prepare it for processing, we can train the Doc2Vec model with training samples. This is one of the advantages of Doc2Vec compared to prior neural network approaches for word embedding.

One of the applications for Doc2Vec is a similarity calculation between documents. With this application, by applying

TABLE I. Real-time information sources

	API	Identification	Timestamp
Twitter [20]	✓	✓	✓
Stack Overflow [21]	✓	✓	✓
Reddit [22]	✓	✓	✓
teratail [23]	×	✓	✓
Security StackExchange [24]	✓	✓	✓

a vector similarity calculation method (e.g., cosine distance) to vectors obtained from documents, we can obtain a similarity value between documents.

III. Proposed system

The proposed system consists of the following four modules:

(1) Collection module
(2) Cleansing module
(3) Signature generation module
(4) Notification generation module.

Figure 1 shows the architecture of the proposed system and its data flow.

First, the collection module collects vulnerability information from real-time information sources such as SNS (Fig. 1 (1)). After that, the proposed system performs data cleansing on the collected data (Fig. 1 (2)). Finally, the proposed system generates WAF signatures and set them (Fig. 1 (3)). At the same time, the proposed system generates a notification file for the administrator (Fig. 1 (4)).

In this system, it is assumed that the administrator registers the names of the web applications and their version information into the system beforehand. It is on the basis of this registered information that the system extracts vulnerability information from the Internet.

A. Collection module

The proposed system generates WAF signatures from the real-time information sources shown in Table I. All these sources are equipped with IDs and timestamps. The collection module collects vulnerability information from the sources and saves their ID, timestamp, and body. Subsequent processes will require these IDs and timestamps to identify the articles and the date of publication.

B. Cleansing module

The proposed system performs data cleansing on collected data to remove duplicate information and get the necessary information. The cleansing module extracts the following attributes from text and web page. The system uses the following attributes for generating WAF signatures.

- Application name
- Vulnerability type
- Version information
- Vulnerability identification information such as a Common Vulnerabilities and Exposures (CVE)-ID

C. Signature generation module

If the web application name that is used for operating web application system is included among the attributes extracted by the cleansing module, the system generates a WAF signature to block HTTP requests to that application and applies it to the WAF. It also notifies the notification generation module that the signature has been generated.

D. Notification generation module

The proposed system generates a notification and sends it to the administrator. This notification includes information such as the name and version of the vulnerable web application. We expect that when the vulnerability is resolved (e.g., applying a patch), this notification will remind the administrator to remove the signature.

IV. Initial Experiments

We conducted the following initial evaluation experiments. For the target web application to collect vulnerability information, we used WordPress as a keyword for extracting vulnerability information due to its known history of many vulnerabilities.

A. Experiment 1: WAF signature generation using CVE information

First, as a proof of concept, we implemented the proposed system shown in Figure 1 with Python 3.6.8 and AWK scripts and examined the automatic generation of WAF signatures using NVD data feeds instead of real-time information.

1) Processing Method: The collection module (Fig. 1 (1)) obtained information from the NVD data feed, which contains CVEs, on a daily basis. The cleansing module (Fig .1 (2)) checked whether the keyword was included in the Common Platform Enumeration (CPE) name for each vulnerability information on the data feed. The following data were then extracted.

1) CVE-ID
2) CPE name
   The CPE name is a name that identifies the platform [25].

   ```
   cpe:2.3:[Part]:[Vendor]:[Product]:[Version]
   :[Update]:[Edition]:[SW_Edition]:[Target_SW]
   :[Target_HW]:[Language]:[Other]
   ```

   In this experiment, we used [Part] and [Product] from the CPE name. [Part] represents a product type with one character, where 'a' is an application, 'o' is an operating system, and 'h' is hardware. [Product] is the product name.
3) Version information

We extracted these data and stored them into JavaScript Object Notation (JSON) format.

Next, the signature generation module (Fig. 1 (3)), we generated a signature for ModSecurity from the extracted information to prevent access to the web application. With reference to `ModSecurity_41_xss_attacks.conf` and `ModSecurity_41_sqlinjection_attacks.conf` from ModSecurity's CRS, we added the following signatures.

- **VARIABLES:** REQUEST_COOKIES | !REQUEST_COOKIES:/__utm/ | REQUEST_COOKIES_NAMES | ARGS_NAMES | ARGS | XML:/

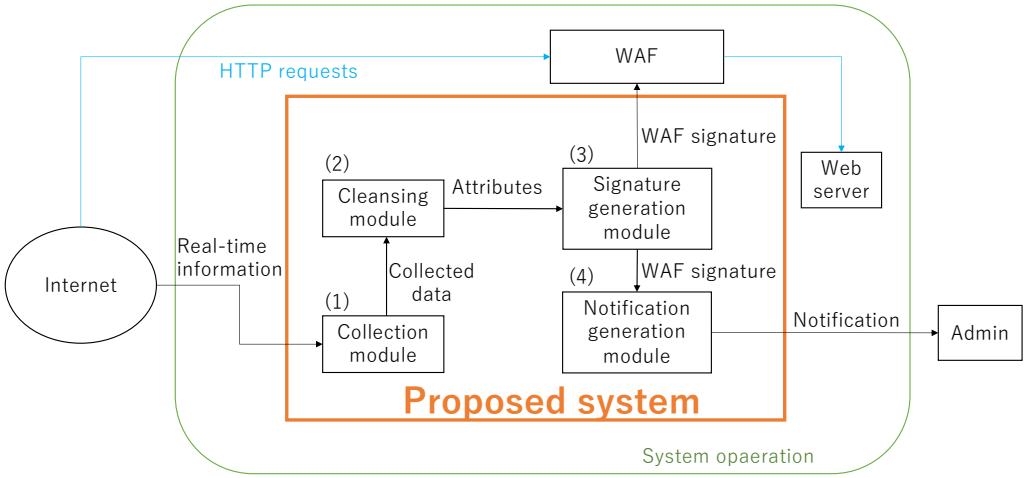


Figure 1. System architecture and data flow

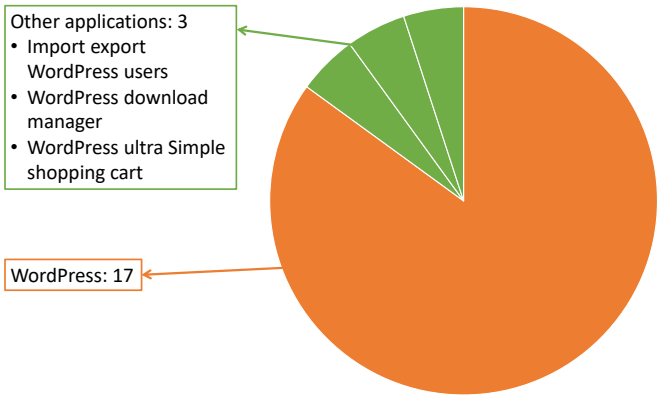


Figure 2. Extracted information in Experiment 1

```
1 SecRule REQUEST_COOKIES|!REQUEST:/__utm/|
    REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "
    import_export_wordpress_users" "phase:2,block,msg:'
    WordPress injection.'severity:'2',id:'15001'"
2 SecRule REQUEST_COOKIES|!REQUEST:/__utm/|
    REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "
    wordpress" "phase:2,block,msg:'WordPress injection.'
    severity:'2',id:'15002'"
3 SecRule REQUEST_COOKIES|!REQUEST:/__utm/|
    REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "
    wordpress_download_manager" "phase:2,block,msg:'
    WordPress injection.'severity:'2',id:'15003'"
4 SecRule REQUEST_COOKIES|!REQUEST:/__utm/|
    REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "
    wordpress_ultra_simple_paypal_shopping_cart" "phase:2,
    block,msg:'WordPress injection.'severity:'2',id
    :'15004'"
```

Figure 3. Generated WAF signature in Experiment 1

- **OPERATOR:** Regular expression using web application names
- **ACTIONS:** phase:2,block,msg:'*application name* injection.',severity:'2',id:'15000+line number'

In addition, we generated text files to provide notification about the signature generation in the notification generation module (Fig.1 (4)).

2) Results: We performed this process once daily for ten days from December 21, 2019 to December 30, 2019.

Since the results for all days during the collection period were the same, we present the result of a single day as an example in Figure 2.

The results of the automatically generated WAF signatures are shown in Figure 3. The signature for WordPress was generated from 17 cases of WordPress vulnerability information and other signatures were generated from one case corresponding to each application.

3) Consideration: The generated signatures show that, in addition to the signature for the actual WordPress application, other applications that include "wordpress" in their name have been blocked as well. These redundant rules place an additional burden on the WAF. To overcome this limitation, we need to improve the signature generation rule and fine-tune it.

TABLE II. Extracted attributes of the tweets

Key	Type	Description
id	Int64	tweet_id as an integer.
created_at	String	The time when the tweet was created.
username	String	User name who posted the tweet.
text	String	The tweet contents in UTF-8 format.
urls	List	Uniform Resource Locators (URLs) in the tweet[1].

B. Experiment 2: Generation of WAF signatures using twitter feed

Next, to investigate the possibility of WAF signature generation from real-time vulnerability information, we collected tweets from twitter feeds, chose one tweet, and generated a WAF signature from it. The tweets were collected using an Application Programming Interface (API) search by setting the query parameters to "wordpress". The chosen tweet is shown in Figure 4. The proposed system uses the information from tweets listed in Table II from tweets, so we extracted the following information.

- **id:** 1200259525707796482

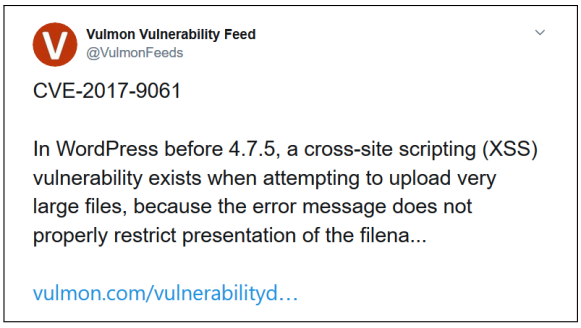


Figure 4. The tweet used in Experiment 2

```
SecRule REQUEST_COOKIES|!REQUEST:/__utm/|
    REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "
    wordpress" "phase:2,brock,msg:'WordPress XSS.'severity
    :'2',id:'15001'"
```

Figure 5. Generated WAF signature in Experiment 2

- **created_at:** 2019-11-29 03:45:45
- **username:** Vulmon Vulnerability Feed
- **text:** CVE-2017-9061\n\nIn WordPress before 4.7.5, a cross-site scripting (XSS) vulnerability exists when attempting to upload very large files, because the error message does not properly restrict presentation of the filena...\n\nhttps://t.co/anaw5-QSAoa"
- **urls:** [http://vulmon.com/vulnerabilitydetails?qid=CVE-2017-9061]

1) Method and results: A visual check of the web application and vulnerability information contained in the tweet resolved the following attributes from the text.

- **Application name:** WordPress
- **Vulnerability type:** XSS
- **Version information:** 4.7.5 and earlier
- **CVE-ID:** CVE-2017-9061

We also checked the web page indicated by the URL but could not obtain any additional attributes. As in experiment 1, we generated a ModSecurity signature from these attributes, which is shown in Figure 5.

2) Consideration: This experiment confirms that a WAF signature could be generated from collected tweets. However, we still need a method to select the relevant tweet. We expect to be able to extract information such as the web application name, its version, or the type of vulnerability through the pattern matching approach.

V. Consideration of Additional Sources

Currently, the real-time information source is only Twitter, and it could be prone to be disinformation. Therefore, we explored the possibility of using other information sources, which are listed in Table I.

To analyze whether these information sources are appropriate sources or not, we reviewed discussions about the following three WordPress vulnerabilities in the corresponding communities. These vulnerabilities are registered in the NVD and have a high Common Vulnerability Scoring System (CVSS) score.

TABLE III. Number of search hits

	2018-20148	2019-17669	2019-20041
Stack Overflow	10(-)	6(-)	10(-)
Reddit	2(-)	15(-)	11(-)
teratail	6(-)	6(-)	3(-)
Security StackExchange	2(-)	0(-)	2(-)

(): Number of search hits related CVEs

TABLE IV. Number of search hits in each knowledge community

Knowledge community\Year	2017	2018	2019	total
Stack Overflow	2,166	2,072	1,837	6,075
Reddit[cybersecurity]	31	172	266	469
Reddit[security]	16	60	151	227
teratail	241	196	172	609
Security StackExchange	1,166	1,072	768	4,006

- CVE-2018-20148 Published: December 14, 2018
- CVE-2019-17669 Published: October 17, 2019
- CVE-2019-20041 Published: December 27, 2019

Since we cannot collect information from teratail via API, we searched the above vulnerabilities by means of a Google search with the queries "WordPress" and "vulnerability". The duration of the search was set to one month before and after the vulnerability announcement. The results are shown in Table III, the numbers in parentheses refer to the number of search hits. These results show that we could not obtain any information about these vulnerabilities from these communities in a timely manner.

In addition, the current system has some problems. As the results shown in Table III are not suitable for a comparison of each knowledge community, we tried additional exploration. Specifically, we compared the number of search hits per site using the Custom Search API provided by Google to see the number of discussions about vulnerabilities in each knowledge community. We set the query "vulnerability" for all sites and set the period from January 1, 2017 to December 31, 2019. Since Reddit has a lot of topics that are not security-related, we only explored two subreddits ("security" and "cybersecurity").

The results are shown in Table IV. Among the knowledge communities examined in this study, Stack Overflow and Security StackExchange had the most active in discussions about the vulnerability, so we conclude they can be used as a good information source.

VI. Filtering and Extracting Tweets through Pattern Matching Approach

A. Description

In big data processing, we generally gather data from information sources through the API of the services or perform web scraping. However, data acquired in such a manner typically contain a lot of totally unrelated information, which means we have to perform a related information extraction or an unrelated information elimination. To circumvent this issue, we propose a filtering method based on regular expression. The method first confirms the existence of specific information such as application name and version number with regular expression and then decides whether the information is related information

or not. To determine the effectiveness of this method, we performed an experiment with gathered tweets.

B. Experiment 3-1: Regular expression-based extraction with tweet

First, on the basis of the results of Experiment 2, we extracted and filtered vulnerability information from the tweets. We then used the twitter API to collect tweets every day for the following period. After eliminating the duplicates, we further processed 1,116 tweets.

- **Collection period:** From December 4, 2019 to January 8, 2020
- **Search queries:** "WordPress AND Vulnerability", "WordPress AND XSS", and "WordPress AND injection"

To check the performance of the filter, we manually assigned the following labels to the tweets based on the relevance to the WordPress vulnerability.

0: WordPress vulnerability information
1: Other information

As a result of the manual labeling, 76 tweets regarding WordPress vulnerabilities were identified. The remaining 1,040 tweets were mistakenly extracted since there URLs referred to, e.g., a web page created using WordPress.

1) Method: We collected the attributes shown in Table II from the tweet and filtered them on the basis of its text context by pattern matching using regular expressions from the tweet's text and web page body indicated by the URL. The following attributes were extracted from the tweet and stored in JSON format.

- **ID:** tweet-id
- **App_name:** If the string includes "wordpress" this element is 1, otherwise 0.
- **CVE:** Extract the string "CVE-\d{4}-\d+" and store it as a list.
- **plugin / theme:** If the string includes "plugin" or "theme" this parameter is 1, otherwise 0.

From these attributes and a list of CVEs corresponding to the target application, we automatically assigned an estimated label for tweets in accordance with the flowchart shown in Figure 6. Each label corresponds to the following categorization result.

0: WordPress vulnerability information
1: Other information
1-a: Does not included the string "wordpress"
1-b: Expected to be a WordPress plugin or theme
1-c: No WordPress vulnerability in CVE list
2: Unfiltered

2) Results: We implemented the filter in Python as per the flowchart shown in Figure 6 and ran it on the 1,116 collected tweets. The results are shown in Table V. As we can see, our method filtered 597 tweets, 98.5% of which were filtered correctly. However, 519 were unfiltered. By using the pattern matching approach, the number of objects to analyze could be reduced by half.

Out of the seven tweets that were not correctly filtered, six were supposed to be labeled as 0 but were mistakenly labeled as 1-b. These were all weekly summaries of vulnerability information for WordPress and related modules and contained information about both WordPress and its plugins. This is why the filter misjudged them as information about WordPress plugins.

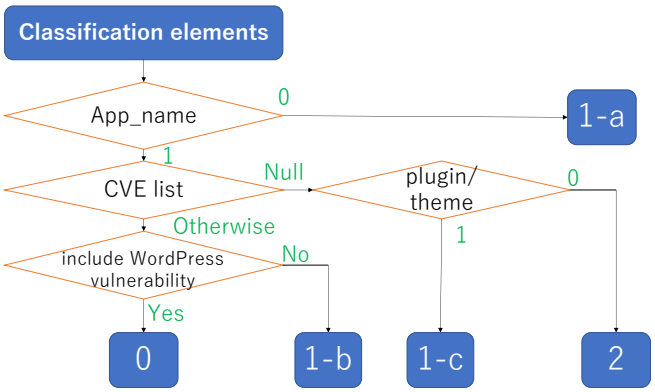


Figure 6. Flowchart of estimated label setting in Experiments 3-1 and 3-2

TABLE V. Filtering results of Experiment 3-1

		Estimated label					
		0	1-a	1-b	1-c	2	total
Correct label	0	13	0	6	0	57	76
	1	1	94	292	191	462	1,040
	total	14	94	298	191	519	1,116

3) Consideration: The information required for the proposed system is vulnerability information that is up-to-date or has not been officially announced, which is included in the tweets labeled as 2 in this experiment. Therefore, we need to provide a way to extract the required information from these tweets.

C. Experiment 3-2: Regular expression-based extraction with other data

To determine the difference among data collection periods and among data sources, we applied the regular expression-based filtering procedure discussed in Section VI-B to the other data. Table VI shows data sources and collection periods.

We performed regular expression-based filtering on the main text of tweet and web page content pointed to by the URL contained in the tweet. We also performed filtering on question texts for Stack Overflow and Security StackExchange data. We manually applied labels to Security StackExchange data and compared these estimated labels provided by the flow in Figure 6.

Table VII shows the filtering results with estimated labels. The label descriptions are the same as those in Section VI-B.

TABLE VI. Source and collection period of Experiment 3-2

Source site	Collection period	Number
Tweet-2021	from February 19, 2021 to June 30, 2021	5,843
Tweet-2020	from March 2, 2020 to June 2, 2020	4,702
Stack Overflow	from January 1, 2018 to December 31, 2020	48,377
Security StackExchange	from January 1, 2018 to December 31, 2020	151

TABLE VII. Filtering results of Experiment 3-2

	0	1-a	1-b	1-c	2	total
Tweet-2021	224	572	1,165	1,276	2,606	5,843
Tweet-2020	33	565	1,615	601	1,888	4,702
Stack OverFrow	1	17,328	3,843	0	27,165	48,337
Security StackExchange	1	8	12	1	129	151

When we compared Tweet-2020 and Tweet-2021, we found that we could filter and classify more than 60% of the data regardless of the data gathering period. Tweet-2021 contained seven times more WordPress vulnerability results (label 0) than Tweet-2020. This difference comes from the fact that a danger vulnerability CVE-2020-36326 (CVSS score: 9.8) was announced in the Tweet-2021 gathering period, which affected the number of WordPress vulnerability results.

We could filter and classify around 45% of the data from Stack Overflow, and there was only one WordPress vulnerability result. We speculate that community members of Stack Overflow who have an interest in vulnerability may also be members of Security StackExchange and asked about it on the Security StackExchange side.

We could only filter and classify around 15% of the data from Security StackExchange. This is presumably because discussions on Security StackExchange are complicated or higher level ones, such as "How can I operate WordPress to avoid security issues as much as possible?" and such discussions do not contain specific version numbers or specific plugins, which are required in the Figure 6 flow.

Table VII shows the relationship between manually applied correct labels and estimated labels. As mentioned, only 15% were classified, so we cannot include the remaining 85% in this discussion. Eight CVEs were issued in the data gathering period, but we only obtained one WordPress vulnerability result (label 0). However, the version numbers listed in the results were not a WordPress versions (i.e., they belonged to the other application), so the results lacked version number information. We clarify why these results were poor through a detailed analysis of the Security StackExchange text data. First, the texts here contained a lot more information than a tweet, which means that one text data frequently contained vulnerability discussions on multiple applications. Furthermore, we found that many text data did not contain key phrases or version numbers compared to tweets. On the other hand, we found that some text contained detailed information such as "problem of illegal communication occurrence" and appeal reminders about various problems. Such information might sometimes include the first report of a zero-day vulnerability, so it may be applicable to the WAF rule for zero-day vulnerability if we can treat such information properly.

To conclude of this subsection, our findings indicate that the filtering method we propose works well for tweets but less well for Security StackExchange, which features longer texts. We also found that the characteristics of the text data in Security StackExchange differ significantly from those of tweets, which may stem from the differing styles of the knowledge communities.

TABLE VIII. Correct/estimated labels of Security StackExchange in Experiment 3-2

		estimated_label					
		0	1-a	1-b	1-c	2	total
correct_label	0	1	0	2	0	5	8
	1	0	8	10	1	124	143
	total	1	8	12	1	129	151

TABLE IX. Learning sources of Experiments 4-1 and 4-2

Data form	Collection period	No.
CVE_WordPress	From August 17, 2005 to June 15, 2021	564
CVE_ALL1	From January 1, 2018 to December 31, 2018	16,511

VII. Narrowing down using Doc2Vec

A. Description

In this section, to automate the target web application vulnerability extraction, we propose a Doc2Vec-based method that trains a Doc2Vec model with past vulnerability descriptions and extracts vulnerability information from the latest open information (e.g., tweets). We use the description sections of CVEs as past vulnerability information sources. We first gathered past CVE descriptions of the target web application and used them to generate the Doc2Vec model and word vectors. We also generated word vectors from large amounts of open information gathered with the trained Doc2Vec model. We then calculated the cosine similarity between individual word vectors of open information and individual word vectors of CVE descriptions. If a word vector of an open information has a large similarity to word vectors of the past CVE vectors of a target web application, we consider the open information to contain vulnerability information of that application. To eliminate the vulnerability information of other web applications, we also created a Doc2Vec model from entire CVE descriptions and related word vectors. The final decision is made on the basis of both similarities to the CVE description of the target web application (high similarity) and similarities to the whole CVE descriptions (low similarity).

B. Experimental setup

We performed an experiment to investigate the relationship between the estimated label in Section VI and the similarity obtained by the Doc2Vec-based method. Table IX lists the data we used for training the Doc2Vec models. We gathered CVEs of the target application (WordPress) that were announced from August 17, 2005 to June 15, 2021 (CVE WordPress). We also gathered all of the CVEs that were announced in 2018 (CVE ALL). We collected the above CVEs using NVD's REST API [25].

We trained and created two Doc2Vec models: The one is WP_R_model, which uses the description section of individual CVE_WordPress data, and ALL_R_model, which uses the description section of individual CVE_ALL data. The Doc2Vec parameters used for training are itemized below. A parameter

TABLE X. Verification data for Experiments 4-1 and 4-2

Data form	Correction period	No.
Tweet	From December 4, 2019 to January 8, 2020	1,116
Security StackExchange	From January 1, 2018 to December 31, 2020	151

value existing at the top is a default value and values existing in the cases are varied during evaluations.

- DM: 1 (learn with PV-DM)
- Learning Rate: 0.025 (0.0012, 0.0025)
- Vector Size: 300 (30, 100, 450, 500, 600, 800, 1000)
- Min_count: 10 (1, 2, 3, 5, 8, 15)

We performed word vector generation for the description section of individual CVE_ALL data with the ALL_R_model and treat it as a summary of the description of individual CVE_ALL data. This word vector set is named V_ALL. Similarly, We performed word vector generation for the description section of individual CVE_WordPress data with the WP_R_model and treat it as a summary of the description of individual CVE_WordPress data. This word vector set is named V_WP.

As test data, we decided to use manually labeled tweet data and Security StackExchange data. We did not used Stack Overflow data because it contains quite a few vulnerability discussion entries (as described in Section VI-C). Table X lists the test data. We performed word vector generation on individual tweet data for both Doc2Vec models and treat them as a summaries of the individual tweet data. These word vector sets are named V_TW_WP (generated with WP_R_model) and V_TW_ALL (generated with ALL_R_model). Similarly, we performed word vector generation for individual Security StackExchange data for both Doc2Vec models and treat them as a summaries of the individual Security StackExchange data. These word vector sets are named as V_SSE_WP (generated with WP_R_model) and V_SEE_ALL (generated with ALL_R_model).

Then, we calculated Similarity_App and Similarity_All values for individual tweet data with the following two steps. First, we chose a tweet data and calculated the cosine distance between the V_TW_WP of the tweet data and an individual V_WP. Second, we calculated the average of the top-5 cosine distance values and used it as the Similarity_App value for that tweet data. This procedure was performed for all tweet data to obtain their Similarity_App values. The same procedure was also performed with V_TW_ALL and V_ALL to obtain Similarity_All values for all tweet data. Similarity_App and Similarity_All for Security Stack Exchange data were generated using the same procedure with V_WP, V_ALL, V_SSE_WP, and V_SSE_ALL.

Finally, we set the threshold value for both Similarity_App and Similarity_All with the heuristic method and calculated the below evaluation index, which is widely used in classification experiment.

- True Positive (TP)
- False Positive (FP)
- True Negative (TN)
- False Negative (FN)
- Accuracy (Acc)
- Precision (Pre)
- True Positive Rate (TPR)
- False Positive Rate (FPR)
- True Negative Rate (TNR)
- False Negative Rate (FNR)

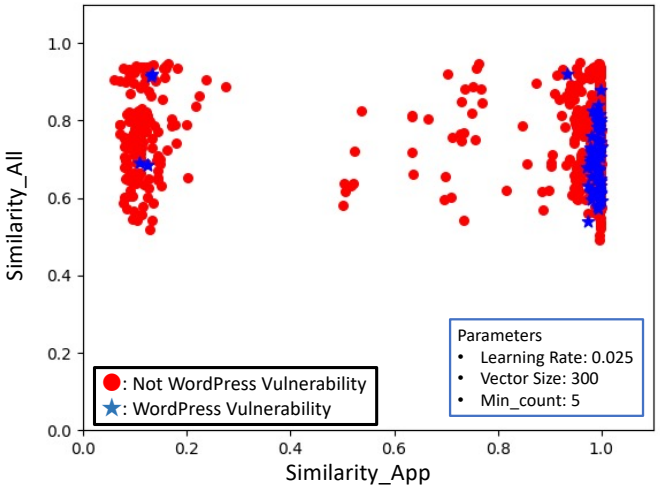


Figure 7. Experriment 4-1: Scatter plot of similarity between tweets and CVEs

C. Experiment 4-1: Results with tweets

Table XI shows the evaluation indices of tweet data with the default Doc2Vec parameters and the parameters with top 2 accuracy. Tweet data contains a comparatively small number words, so it adopts a smaller Vector Size value and smaller Min_count value. The best parameter in this experiment was Vector Size = 300 and Min_count = 5 with Similarity_App $\geq$ 0.95 and Similarity_All $<$ 0.8 threshold values. The result did not change even in the short Vector Size value, which means we can reduce the calculation cost by urilizing the shorter Vector Size value (e.g., Vector Size = 30). When we reduced the Min_count to less than 5, the result became dramatically worse. This result suggests that too small Min_count will include words that are unnecessary for the Doc2Vec model and thereby lead to bad results. The learning rate parameters affected quite a few of the results in this manner, so we had to omit them.

Figure 7 shows the distribution of Similarity_App and Similarity_All with the WordPress Vulnerability or Not WordPress Vulnerability label in the best parameter. The horizontal axis denotes Similarity_App and the vertical axis denotes Similarity_All. The blue star and red circle indicate that a tweet data does or does not contain a WordPress vulnerability, respectively. We can see that a large number of WordPress Vulnerability data are gathered near the Similarity_App = 1.0 area. However, a significant amount of Not WordPress Vulnerability data is also gathered there, which indicates a high number of false positive results. Similarity_All can eliminate some of the Not WordPress Vulnerability data. However, since V_ALL is generated from all of the CVEs, and as such contains many security and vulnerability related words, WordPress data has a comparatively large similarity to V_ALL values, which reduces the separation ability with Similarity_All.

D. Experiment 4-2: Results with Security StackExchange

Table XII shows the evaluation indices of Security Stack Exchange data with the default Doc2Vec parameters and parameters with top-2 accuracy. Security StackExchange data contains a comparatively large number words, so it adopts

TABLE XI. Metrics of Experiment 4-1: Difference by Vector_Size and Min_count.

	TP	FP	TN	FN		Acc	Pre	TPR	TNR	FPR	FNR
Vector: 300, Count: 10	66	722	317	11		34.3%	85.7%	8.4%	96.6%	3.4%	91.6%
Vector: 300, Count: 5	68	614	425	9		44.2%	88.3%	10.0%	97.9%	2.1%	90.0%
Vector: 30, Count: 5	70	637	402	7		42.3%	90.9%	9.9%	98.3%	1.7%	90.1%

TABLE XII. Metrics of Experiment 4-2: Difference by Vector_Size.

	TP	FP	TN	FN		Acc	Pre	TPR	TNR	FPR	FNR
Vector: 300, Count: 10	5	44	99	3		68.9%	62.4%	10.2%	97.1%	2.9%	89.8%
Vector: 450, Count: 10	5	32	111	3		76.8%	62.5%	13.5%	97.3%	2.6%	86.5%
Vector: 600, Count: 10	6	57	86	2		60.9%	75.0%	9.5%	97.7%	2.3%	90.5%

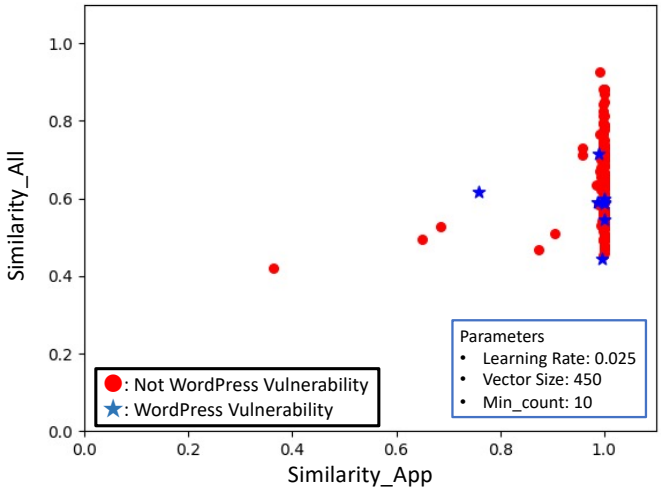


Figure 8. Experriment 4-2: Scatter plot of similarity between SSE and CVEs

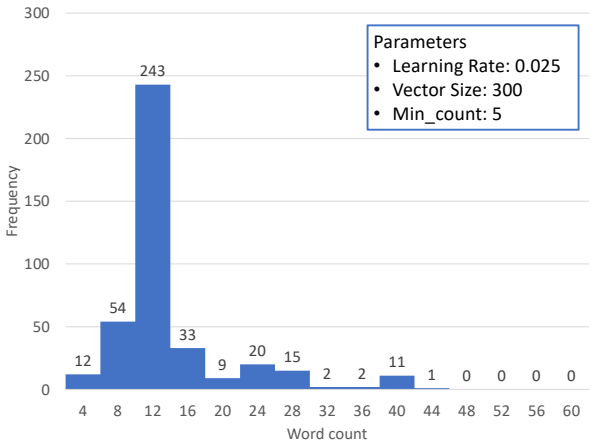


Figure 9. Experriment 4-1: Histgram of true negatives

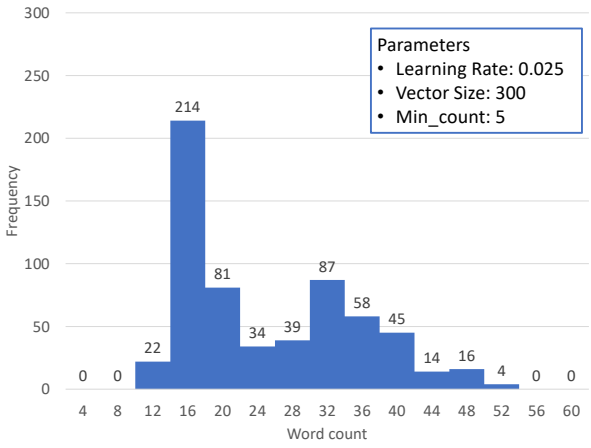


Figure 10. Experriment 4-1: Histgram of false positives

a larger Vector Size value. The best parameter in this experiment was Vector Size = 450 and Min_count = 10 with Similarity_App $\geq$ 0.98 and 0.54 $<$ Similarity_All $<$ 0.64 threshold values. We conclude that the default Vector Size is not sufficient to represent a large number words. When we reduced Min_count to less than 10, the result became worse.

Figure 8 shows the distribution of Similarity_App and Similarity_All with WordPress Vulnerability or Not WordPress Vulnerability labels in the best parameter. The organization of the figure is identical to Figure 7. Similar to the results for the tweet data, a lot of the Not WordPress Vulnerability data here is gathered near the Similarity_App = 1.0 area, which indicates a high number of false positive results. This trend was stronger than it was in the tweet data, as indicated by almost all of the Not WordPress Vulnerability data in Figure 8 being distributed close to Similarity_App = 0.1. However, the Similarity_All values of WordPress Vulnerability data were comparatively small, so we can omit the Not WordPress Vulnerability Data by reducing the Similarity_All threshold value compared to the tweet data result. This characteristics is connected to the higher accuracy result than that of the tweet data.

E. Relationship between negatives and word count in individual tweet data

As discussed in Section VII-B, since Doc2Vec-based classification could not classify negatives, it had a higher number of false positive results, especially for tweet data. To analyze why this occurred, we examined the relationship between negatives and word count in individual tweet data. Our hunch was that, since tweet data cobtain fewer words, they might have been comparatively harder to separate.

Figures 9 and 10 show the relationship between word count and true negatives and false positives, respectively. The horizontal axis denotes word count in bin size 4 and the vertical axis denotes the frequency of an individual bin.

As we can see in Figure 9, true negatives appeared more frequently in comparatively lower word count areas (less than 16). In contrast, in Figure 10, we can see that false positives were concentrated more on big word count areas (more than 16). These findings demonstrate that,contrary to our expection, the proposed method is quite capable of separating small word count data. If we can improve the classification result of the higher word count area, the current false positive rate will be mitigated. There are many approaches we could take for making such an improvement, such as using a different model with higher word count areas (e.g. utilizing a more complicated text classification model).

F. Summary of Doc2Vec results

The regular expression-based classification discussed in Section VI cannot classify large amounts of data, especially in Security StackExchange data. In contrast, we found here that Doc2Vec-based classification can perform classification on whole data with moderate performance. Furthermore, it can omit negatives with high accuracy in both datasets. Thus, we recommend combining both regular expression-based and Doc2Vec-based classifications with the following rules.

- Apply the Doc2Vec-based classification in the first stage and omit data classified as negative.
- Apply the regular expression-based classification to data classified as positive in the first stage.

VIII. CONCLUSION

In this paper, we proposed a WAF signature generation system using real-time open information (e.g., SNS) to provide early protection for web applications. To extract vulnerability information from real-time open information, we developed a regular expression-based filtering method and a Doc2Vec-based similarity calculation method evaluated them with Twitter and Security Stack Exchange data.

With the regular expression-based filtering method, we could classify 60% of the tweet data as to whether it contained information about vulnerability of the target web application. However, we could only classify 10% of the Security StackExchange data, as these data were comparatively long and sometimes contained multiple vulnerabilities of several different applications.

With the Doc2Vec-based method, we could classify both tweet data and Security StackExchange data with moderate accuracy, especially when it came to extracting true positive data (i.e., data that actually discussed the vulnerability of the target web application). However, there were still a number of false positive results, and it requireed further classification of data rated as positive.

On the basis of these findings, we proposed a sequential use of both methods. In the first stage, we perform the Doc2Vec-based filtering is applied to omit data rated as negative, as those data contain almost no actually positive data. In second stage, regular expression-based filtering is applied to perform detailed classification for extracting actually positive data.

We also examined the relationship between word count and classification accuracy in data rated as negative in the Doc2Vec-based method. We found that the proposed method could classify small word count areas well enough but struggled with larger word count areas. It should be possible to significantly reduce the number of false positives if we use a more complicated text classification algorithm or advancement of future text classification algorithm.

With the advancement of text classification algorithms expected in the future, our proposed method shows good potential for generating WAF signatures and will be beneficial for protecting web applications from attacks following official vulnerability notifications, especially in case of zero-day vulnerability.

REFERENCES

[1] M. Kumazaki, Y. Yamaguchi, H. Shimada, H. Hasegawa, "WAF Signature Generation with Real-Time Information on the Web," In Proceedings of the 14th International Conference on Emerging Security Information, Systems and Technologies (SECURWARE 2020), ISBN: 978-1-61208-821-1, pp. 40-45, November 2020.

[2] Significant Cyber Incidents — Center for Strategic and International Studies https://www.csis.org/programs/technology-policy-program/significant-cyber-incidents [retrieved: July, 2020]

[3] The 15 biggest data breaches of the 21st century | CSO Online https://www.csoonline.com/article/2130877/the-biggest-data-breaches-of-the-21st-century.html [retrieved: August, 2020]

[4] 10 Major Security Threats 2019 — Information-Technology Promotion Agency (IPA), Japan https://www.ipa.go.jp/files/000076989.pdf [retrieved: July, 2020]

[5] Think Fast: Time Between Disclosure, Patch Release and Vulnerability Exploitation — Intelligence for Vulnerability Management, Part Two | FireEye Inc https://www.fireeye.com/blog/threat-research/2020/04/time-between-disclosure-patch-release-and-vulnerability-exploitation.html [retrieved: August, 2020]

[6] Attacks Heating Up Against Apache Struts 2 Vulnerability | Threatpost https://threatpost.com/attacks-heating-up-against-apache-struts-2-vulnerability/124183/ [retrieved: July, 2020]

[7] US House of Representatives Committee on Oversight and Government Reform, "The Equifax Data Breach", Majority Staff Report 115th Congress, 2018. https://republicans-oversight.house.gov/wp-content/uploads/2018/12/Equifax-Report.pdf [retrieved: July, 2020]

[8] GMO Payment Gateway, "Apology and Report for Leak of Personal InformationDueto Unauthorized Access", 2017. https://www.gmo-pg.com/en/corp/newsroom/pdf/170310_gmo_pg_en.pdf [retrieved: July, 2020]

[9] Committee on National Security Systems, "Committee on National Security Systems(CNSS) Glossary", CNSSI No. 4009, 2015

[10] Chrome Releases: Stable Channel Update for Desktop https://chromereleases.googleblog.com/2019/03/stable-channel-update-for-desktop.html [retrieved: July, 2020]

[11] T. Sakaki, O. Makoto, and M. Yutaka, "Earthquake shakes Twitter users: real-time event detection by social sensors.", In Proceedings of the 19th international conference on World wide web, pp. 851-860, 2010.

[12] O. Oh, M. Agrawal, and H. R. Rao, "Information control and terrorism:Tracking the mumbai terrorist attack through twitter.", Information-Systems Frontiers, vol. 13, no. 1, pp. 33 – 43, 2011.

[13] S. More, M. Matthews, A. Joshi, and T. Finin, "A Knowledge-Based Approach To Intrusion Detection Modeling.", In Proceedings of 2012 IEEE Symposium on Security and Privacy Workshops, pp. 75-81, May 2012.

[14] C. Kreibich and J. Crowcroft, "Honeycomb: Creating Intrusion Detection Signatures Using Honeypots.", SIGCOMM Computer Communication Review., Vol. 34, No. 1, pp. 51-56, January 2004.

[15] S. Singh, C. Estan, G. Varghese, and S. Savage, "Automated Worm Fingerprinting.", In Proceedings of the 6th Symposium on Operating Systems Design and Implementation, pp.4-4, December 2004.

[16] NVD - Home https://nvd.nist.gov/ [retrieved: October, 2020]

[17] K. Lawrence and L. Luo, “Interactive management of web application firewall rules.”, U.S.Patent, No. 9,473,457, 2016

[18] OWASP Foundation | Open Source Foundation for Application Security https://owasp.org/ [retrieved: October, 2020]

[19] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space,” arXiv:1301.3781, January 2013.

[20] Twitter https://twitter.com [retrieved: October, 2020]

[21] Stack Overflow - Where Developers Learn, Share, & Build Careers https://stackoverflow.com [retrieved: October, 2020]

[22] reddit: the front page of the internet https://www.teratail.com [retrieved: October, 2020]

[23] teratail teratail.com [retrieved: October, 2020]

[24] Information Security Stack Exchange https://security.stackexchange.com/ [retrieved: October, 2020]

[25] B. Cheikes, D. Waltermire, and K. Scarfone, “Common platform enumeration: Naming specification version 2.3”, NIST Interagency Report 7695, pp.11-13, 2011.

[26] NVD - New NVD CVE CPE API and SOAP Retirement https://nvd.nist.gov/General/News/New-NVD-CVE-CPE-API-and-SOAP-Retirement

Automatic Mapping of Threat Information to Adversary Techniques Using Different Datasets

Otgonpurev Mendsaikhan*
ogo@net.itc.nagoya-u.ac.jp

Hirokazu Hasegawa†
hasegawa@icts.nagoya-u.ac.jp

Yukiko Yamaguchi*
yamaguchi@itc.nagoya-u.ac.jp

Hajime Shimada*
shimada@itc.nagoya-u.ac.jp

*Information Technology Center, Nagoya University, Furocho, Chikusa Ward, Nagoya, Aichi 464-8601, Japan
†Information Security Office, Nagoya University, Furocho, Chikusa Ward, Nagoya, Aichi 464-8601, Japan

***Abstract*—Along with the growth in the usage of software in almost every aspect of human life, the risks associated with software security vulnerabilities also increase. The number of average daily published software vulnerabilities exceeds the human ability to cope with it; hence, various threat models to generalize the threat landscape have been developed. The most prevalent threat model MITRE ATT&CK proved to be a valuable tool for the security analyst to perform cyber threat intelligence, red and blue teaming, and so on. However, the security analyst must prioritize his/her defense by manually mapping the daily published threat information to the adversarial techniques listed in MITRE ATT&CK for his/her day-to-day operation. This paper proposes a method to automatically map the cyber threat information using a multi-label classification approach. We conducted four experiments using three publicly available datasets to train and test seven multi-label classification methods and one pre-trained language model in six evaluation measures. According to our estimate, the LabelPowerset method with Multilayer Perceptron as the base classifier performs best in our experiment.**

Keywords–Multi-label classification; MITRE ATT&CK; Cyber Threat.

I. INTRODUCTION

In our previous work [1] we experimented with various multi-label classification methods to evaluate the performance to automatically map the vector representations of vulnerability description to adversary techniques and tactics. In this work, we extend the scope to map cyber threat information, including the vulnerability description to the same adversarial techniques.

The digital age has presented various opportunities to society along with different challenges. One of the biggest challenges comes with the risk of cyber-attack, data breaches and loss of intellectual property, etc. Software security vulnerability is one of the most significant factors behind these challenges. According to the US National Vulnerability Database (NVD), the total number of reported vulnerabilities as of June 2020 is 146,000 [2], and this number is increasing year by year. In 2019 alone, 20,362 vulnerabilities are reported on NVD, which is a 17.6% increase from 2018 (17,308) and 44.5% increase from 2017 (14,086), and the trend is likely to be upwards [3].

Given this large number of reported vulnerabilities, tracking individual vulnerabilities is nearly impossible. Hence, various approaches and threat models are developed to generalize the threat landscape and ease the burden of a security analyst. One of the most commonly used approaches is a curated knowledge base called MITRE ATT&CK® that enlists adversary behaviors, including their tactics and techniques based on real-world observations. It is a robust framework commonly used as a threat model in adversary emulation, red and blue teaming, and cyber threat intelligence practices [4]. MITRE ATT&CK generalizes the adversary attack techniques and tactics based on the common weaknesses of the systems without mentioning specific products or vulnerabilities.

Even though the MITRE ATT&CK proved to be a helpful framework, the need to identify the specific threat that individual vulnerability poses in the adversarial landscape still exists. In layperson's terms, MITRE ATT&CK is the playbook of steps that house robbers would take to rob a house (e.g., find open access), and software security vulnerability is the weaknesses of the house security (e.g., an unlocked door or broken window). For effective defense, the house owner needs to combine this information, the most common approaches that house robbers use, and the weaknesses of his/her house to better understand the situation and prioritize his/her defenses.

This paper proposes a method to automatically map the cyber threat information, including vulnerability description to adversary techniques and tactics. Since a specific threat or vulnerability can be associated with more than one adversarial technique, we believe developing a multi-label classification model that can infer the adversarial techniques to a given threat would be suitable. Since every vulnerability has associated textual description, we believe using the features of this text; a classic multi-label classification algorithm could produce a result that could be useful for a practical purpose.

Mapping threat information to adversarial tactics and techniques requires a certain level of expertise and domain knowledge. Thus, it may consume a considerable amount of time for the security analyst. In our previous work [1] we experimented

with a limited dataset to demonstrate the method, and the LabelPowerset method with Multilayer Perceptron as base classifier performed best. In this paper, we extend the work by including an additional and more comprehensive dataset and state-of-the-art language model Bidirectional Encoder Representations from Transformers (BERT) and compared the results. By utilizing such existing tools and data, we believe the task of mapping threat information could be automated to spare the human analyst from manual labor. Hence, the paper aims to seek the possibilities to automate the mapping of threat information to adversarial techniques by exploring the existing tools and evaluating them with different datasets.

This work aims to extend the scope of the [1] with additional datasets and methods and explore the possibilities to gain further insight into the current knowledge in the security domain.

The specific contributions of the paper are as follows:

1) To propose an approach to automate the mapping of threat information to adversarial technique.
2) Explore and experiment with various multi-label classification methods and language models to compare the performance.

The remainder of this paper is organized as follows. Section II will review the related research and how this paper differs in its approach. In Section III, we will briefly discuss the background information to be used for this research. In Section IV, the experimental dataset used in this study will be described, and in Section V, the experimental setup of the proposed multi-label classification approach will be discussed. In Section VI, we will show the experimental result and corresponding analysis, and finally, we will conclude by discussing the implications of this result in Section VII.

II. Related work

There have been various studies related to the content of this work. But to the best of our knowledge, using a language model on multi-label classification task of text-based threat information to map to adversarial technique has not been published yet.

A. Threat intelligence and MITRE ATT&CK framework

There has been an attempt to use a multi-label classification approach to map cyber threat intelligence reports to adversarial techniques and tactics. Legoy et al. implemented a tool called rcATT. This system predicts tactics and techniques related to given cyber threat reports and outputs the results using Structured Threat Information eXpression (STIX) format [5]. They focused on extracting MITRE ATT&CK techniques and tactics from cyber threat reports. They used more straightforward approaches for text representation and classification algorithms. In contrast, we focused on mapping the threat information to the same framework, though using more neural and deep learning approaches.

Also, extracting general Tactics, Techniques, and Procedures (TTP) from cyber threat information is gaining some attention. Husari et al. developed a system to automate Cyber Threat Intelligence (CTI) analytics that learns attack patterns [6]. They combined Natural Language Processing (NLP) and Information Retrieval (IR) techniques to extract threat actions from threat reports based on their semantic relationships. Their focus was to extract actionable TTP from threat reports, whereas our focus is to identify the adversarial techniques that can exploit the specific threat.

Apart from extracting an adversarial technique from textual documents, some studies have directly mapped the malware behavior to the MITRE ATT&CK framework. Oosthoek et al. did the automated analysis of 951 unique families of Windows malware and mapped them onto the MITRE ATT&CK framework [7]. They generated a behavior signature of the malware in the sandbox and mapped the signature to the corresponding MITRE ATT&CK technique. Their work focused on mapping the malware based on its behavior to the adversarial techniques defined in the MITRE ATT&CK framework. In contrast, our focus is to map the threat information that the adversary could exploit to the same techniques through its textual representation.

Some researchers have been working on the information provided by the MITRE ATT&CK framework to improve the adversarial predictions. Al-Shaer et al. presented their statistical machine learning analysis on Advanced Persistent Threat (APT) and software attack data reported by MITRE ATT&CK to infer and predict the techniques the adversary might use [8]. They associated adversarial techniques using hierarchical clustering with 95% confidence, providing statistically significant and explainable technique correlations. Our focus is to correlate individual threat information to the adversarial techniques and create a model that can be used to map new threats to the MITRE ATT&CK framework automatically.

There have also been works on classifying the vulnerability information based on its textual description. Huang et al. proposed an automatic vulnerability classification model built on Term Frequency-Inverse Document Frequency (TF-IDF), Information Gain (IG), and deep neural network [9]. They validated their model with CVE descriptions of the National Vulnerability Database and compared them to the performances of SVM, Naive Bayes, and kNN algorithms. We are also attempting to classify the vulnerability information based on its textual description. Still, Huang et al. focused on a multi-class classification that each vulnerability belongs to a specific category. In contrast, we attempt to classify a vulnerability or threat information into multiple adversarial techniques simultaneously.

Hemberg et al. proposed an open-source, relational graphing tool BRON which links MITRE ATT&CK, CWE, CAPEC, and CVE information to gain further insight from available threat intelligence in [10]. Their proposed method correlated those publicly accessible information sources to make it more usable for the analysts and systematically hunt the threat. Our approach is similar by associating the available threat information with a publicly accessible knowledge base of MITRE ATT&CK to assist the human analysts.

Zhou et al. proposed a method to automatically identify Indicators of Compromise (IOC), an essential artifact of cyber threat intelligence using a neural-based sequence labeling model to identify IOCs from cybersecurity reports [11]. The model used attention mechanism and token spelling features to identify low-frequency IOCs from long sentences of the cybersecurity reports. We are also using an attention-based pre-trained model to determine the associated technique in the threat information.

B. Multi-label classification using language models

Medina et al. created and analyzed a text classification dataset from United Nation's Sustainable Development Goals [12]. They experimented with various multi-label classification methods and pre-trained language models, and according to their estimation, the pre-trained language model BERT showed the best performance. Similarly, Sovrano et al. proposed to utilize the Text Similarity Approach using Term Frequency-Inverse Document Frequency (TF-IDF) and pre-trained sentence embedding framework Universal Sentence Encoder (USE) [13]. In this work, we use a similar approach of labeling threat information in text format by embedding it through USE and applying multi-label classification models on top of it. Also, we compared those results with the performance of the BERT language model, which used its own embedding layers instead of USE.

Liu et al. proposed to utilize deep learning to Extreme Multi-label Text Classification (XMTC) task with a family of new Convolutional Neural Network (CNN) models tailored for multi-label classification task [14]. Their approach successfully scaled to the most extensive datasets of 6 benchmark datasets and consistently produced the best or the second-best results on all the datasets. Their focus was the multi-label classification performance on the extreme setting, i.e., too many labels to predict. Pal et al. proposed to correlate the labels in multi-label classification task through attention-based graph neural network [15]. The attention in their model allowed the system to assign different weights to neighbor nodes per label, thus, allowing it to learn the dependencies among labels.

Since the inception of the BERT language model, there have been various attempts to train and utilize it for specific domains, especially in a multi-label classification task. Lee et al. described a patent classification system created by fine-tuning the BERT language model [16]. Their model outperformed the state-of-the-art results in classifying the patent claims in multi-label settings. Similarly, Adhikari et al. presented DocBERT for document classification task [17]. By minimizing cross-entropy and binary cross-entropy loss, DocBERT achieved state-of-the-art results across four popular datasets in single-label and multi-label tasks.

Although there have been various attempts to utilize different language models in a multi-label classification setting, its potential has yet to be fully explored in the cybersecurity domain.

III. Background

Since this study is at the intersection of different fields, the theoretical background knowledge is briefly explained in this section.

A. Vulnerability Modeling

There have been several attempts to standardize the reporting and modeling of software security vulnerabilities or weaknesses and threat landscape in general. In this section, we will discuss a few relevant schemes for this study.

1) Common Vulnerabilities and Exposures: Common Vulnerabilities and Exposures (CVE) is a list of entries, each containing an identification number, description, and at least one public reference for publicly known cybersecurity vulnerabilities [18]. CVE was launched in 1999 and now became the standard naming convention to address interoperability and disparate databases and tools. CVE entries, also called CVEs, CVE IDs, and CVE numbers by the community, provide common reference points so that cybersecurity products and services can speak the same language. CVE is an international cybersecurity community effort, and each new CVE entry is assigned by CVE Numbering Authorities (CNAs).

The majority of the disclosed vulnerabilities are stored at the NVD for centralized vulnerability management purposes. The NVD is the U.S. government repository of standards-based vulnerability management data and is known as the de facto central database of software security vulnerabilities [19]. CVEs stored at NVD proved to be a valuable resource for vulnerability management and overall cybersecurity-related research.

2) Common Attack Pattern Enumeration and Classification: Common Attack Pattern Enumeration and Classification (CAPEC) efforts provide a publicly available catalog of common attack patterns that helps users understand how adversaries exploit weaknesses in applications, and other cyber-enabled capabilities [20]. CAPEC was established by the U.S Department of Homeland Security in 2007 and continuously evolved to include public participation and contributions. CAPEC defines "Attack Patterns" as descriptions of adversaries' common attributes and approaches to exploit known weaknesses in cyber-enabled capabilities. Each attack pattern captures knowledge about how specific parts of an attack are designed and executed and provides guidance on mitigating the attack's effectiveness.

CAPEC differs from the MITRE ATT&CK framework to focus on application security and enumerates exploits against vulnerable systems. In contrast, the MITRE ATT&CK framework focuses on network defense and provides a contextual understanding of malicious behavior. CAPEC is mainly used for application threat modeling and developer training and education, whereas ATT&CK is used to compare network defense capabilities and hunt new threats.

3) MITRE ATT&CK framework: Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) was created at MITRE corporation to systemically categorize adversary behavior in September 2013 [4]. It was initially designed to document and categorize post-compromise adversary TTPs against Microsoft Windows systems and later added other platforms called ATT&CK for Enterprise and publicly released in May 2015. Subsequently, complementary models such as PRE-ATT&CK, ATT&CK for Mobile, and ATT&CK for ICS have been published in 2017 and 2020. The ATT&CK framework consists of the following components:

- **Adversary group**: Known adversaries that are tracked and reported in threat intelligence reports.
- **Tactics**: Tactics represent the adversary's tactical objective: the reason for performing an action.
- **Technique/Sub-Technique**: Techniques represent "how" an adversary achieves its tactic, whereas Sub-technique further breaks down techniques into more specific descriptions of actions to reach the goal.
- **Software**: Software represents an instantiation of a technique or sub-technique at the software level.

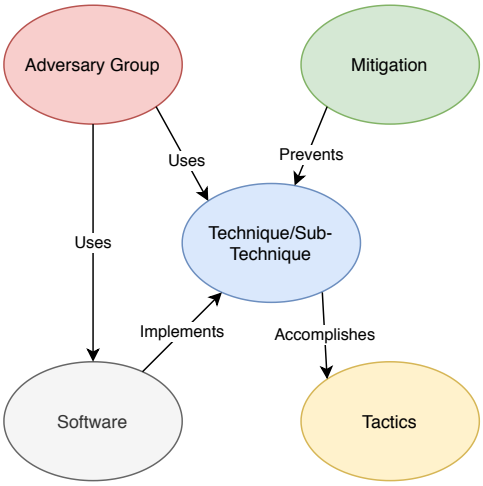


Figure 1. MITRE ATT&CK components and their relationship.

- **Mitigation**: Mitigation represents security concepts and technologies to prevent a technique or sub-technique from being successfully executed.

The relationship between the components is visualized in Figure 1.

The MITRE ATT&CK framework is constantly enriched with techniques and sub-techniques. At the time of writing, there are 266 techniques/sub-techniques of 12 tactics in the MITRE ATT&CK Enterprise model, 174 techniques of 15 tactics in the PRE-ATT&CK and 79 techniques of 13 tactics in ATT&CK for Mobile model.

B. Multi-label classification

Classification is the task of learning to classify the set of examples that are from a set of disjoint labels L, $|L|> 1$. If $|L|= 2$, then the learning problem is called a binary or single-label classification and if $|L|> 2$, it is a multi-class classification. In the case of multi-class classification, the example should correspond to a single class or label. In contrast, in multi-label classification, the examples are associated with a set of labels $Y \subseteq L$ [21]. According to Madjarov et al., the multi-label classification methods could be of the following categories [22].

1) **Algorithm adaptation methods:** The existing machine learning algorithms that are adapted, extended, and customized for multi-label classification problems. The examples include: boosting, k-nearest neighbors, decision trees, and neural networks.
2) **Problem transformation methods:** This method transforms the multi-label classification into one or more single-label classification or regression problems. It is further divided into categories as binary relevance, label power-set, and pair-wise methods.
3) **Ensemble classification:** The ensemble methods are developed on top of existing problem transformation or algorithm adaptation methods. The examples include Random k-label sets (RAkEL) and ensembles of pruned sets (EPS) etc.

C. Evaluation measures of multi-label classification

Since the multi-label classification task is different from the traditional binary classification, the evaluation metrics to measure the method's performance also differ. The multi-label classification measures generally fall into the following categories according to [22].

1) Example based measures
2) Label based measures
3) Ranking based measures

The evaluation measures used in this study are briefly discussed below. In the definitions, y_i denotes the set of true labels for example x_i and $h(x_i)$ denotes the set of predicted labels for the same examples. N is the number of examples, and Q denotes the total number of possible class labels.

1) Subset Accuracy: Subset Accuracy, also called as Exact Match Ratio is the most strict metric, indicating the percentage of samples that have all their labels classified correctly. It can be calculated as shown in (1):

$$Accuracy(h) = \frac{1}{N}\sum_{i=1}^{N} I(h(x_i) = y_i) \tag{1}$$

where $I(true) = 1$ and $I(false) = 0$.

2) Micro averaged F1 score: Since the classification is on multiple labels, the results have to be averaged out. Micro-precision and micro-recall are the measures averaged over all the example/label pairs. In the definitions below TP_j, TN_j denote the number of True Positive and True Negative, FP_j, FN_j denote the number of False Positive and False Negative examples per label λ_j when considered as binary classification.

$$Precision = \frac{\sum_{j=1}^{Q} TP_j}{\sum_{j=1}^{Q} TP_j + \sum_{j=1}^{Q} FP_j} \tag{2}$$

$$Recall = \frac{\sum_{j=1}^{Q} TP_j}{\sum_{j=1}^{Q} TP_j + \sum_{j=1}^{Q} FN_j} \tag{3}$$

The Micro averaged F1 Score is the harmonic mean between micro-precision and micro-recall.

$$F1 = \frac{2 \times microPrecision \times microRecall}{microPrecision + microRecall} \tag{4}$$

3) Macro averaged F1 score: Macro-precision and macro-recall are the measures averaged across all labels and defined as shown in (5) and (6).

$$Precision = \frac{1}{Q}\sum_{j=1}^{Q} \frac{TP_j}{TP_j + FP_j} \tag{5}$$

$$Recall = \frac{1}{Q}\sum_{j=1}^{Q} \frac{TP_j}{TP_j + FN_j} \tag{6}$$

Macro-F1 is the harmonic mean between precision and recall, where the average is calculated per label and then averaged across all labels. If P_j and R_j are the precision and recall for all $\lambda_j \in h(x_i)$ from $\lambda_j \in y_i$ then Macro F1 is defined as in (7):

$$F1 = \frac{1}{Q}\sum_{j=1}^{Q} \frac{2 \times P_j \times R_j}{P_j + R_j} \tag{7}$$

4) Hamming loss: Hamming loss evaluates how many times an example-label pair is misclassified, i.e., a fraction of incorrectly predicted labels.

$$HammingLoss(h) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{Q} \mid h(x_i)\Delta y_i \mid \quad (8)$$

where Δ stands for the symmetric difference between two sets. The smaller the Hamming loss better the model performance.

5) Ranking loss: Ranking loss evaluates the average fraction of label pairs that are reversely ordered for the particular example.

$$RankingLoss(h) = \frac{1}{N}\sum_{i=1}^{N}\frac{\mid D_i \mid}{\mid y_i \mid\mid \bar{y_i} \mid} \quad (9)$$

where

$D_i = \{(\lambda_m, \lambda_n) \mid f(x_i, \lambda_m) \leq f(x_i, \lambda_n), (\lambda_m, \lambda_n) \in y_i \times \bar{y_i})\}$, while $\bar{y_i})\}$ denotes the complementary set of y in L. The smaller the Ranking loss better the model performance.

6) Micro averaged Receiver Operating Characteristic: Receiver Operating Characteristic (ROC) is a measure used mainly in binary classification to study the output of a classifier. It is a probability curve that plots the True Positive Rate (TPR) against False Positive Rate (FPR) at the various threshold and essentially separates the signal from noise. ROC is usually represented as the Area Under Curve (AUC) graph. In order to use ROC in a multi-label setting, examples are binarized per each label and averaged for aggregated contributions of all classes to compute the Micro averaged metric. Biggest data science online community Kaggle [23] uses Micro averaged ROC score to evaluate the competitions in multi-label problems. Hence, we decided to adopt this measure to evaluate the performances of different models.

D. Language models in Natural Language Processing

Since natural language can not be absolutely formalized in specific rules in a similar way as programming languages, computational linguists approached it by specifying the model of the language from the example texts. Thus, language modeling became a crucial component of Natural Language Processing (NLP) and used various statistical and probabilistic techniques to determine the probability of a given sequence of text to appear in a particular position.

The latest trend in NLP includes the utilization of a pre-trained language model in a transfer learning setting. A pre-trained language model is a language model which has been fed a large amount of unannotated data. As a result, the model learns the usage of various words and general rules of the language. Then the model is transferred to a downstream task where it is fed a smaller, task-specific dataset, through which the model is fine-tuned to perform well on the task. It is basically creating a machine equivalent of a "well-read" human being.

Another significant advancement is transformer-based language models. The attention mechanism is used to selectively attend to some part of the input example, similar to humans focusing on the object's detail. The transformer is a novel architecture introduced in [24] that was considered a breakthrough in NLP. It relies on the self-attention mechanism to compute the input and output representations by handling the dependencies between them.

Current state-of-the-art results in NLP belong to the models such as Generative Pre-trained Transformer-3 (GPT-3) [25], Bidirectional Encoder Representations from Transformers (BERT) [26] and XLNet [27] that were based on the transfer learning and transformer architecture.

This study used two pre-trained and one transformer-based language model, namely Universal Sentence Encoder (USE) and Bidirectional Encoder Representations from Transformers (BERT). The pre-trained USE model is used to embed or convert the text document into vector representations that could be used to apply traditional multi-label classification methods. In contrast, the pre-trained, transformer-based BERT model is used as a standalone model that performs embedding and classification in itself.

IV. Experimental Dataset

The amount and quality of the dataset used are essential to the performance of the trained model. Hence, data has been collected from different sources to represent the diverse nature of the threat information.

A. Data sources

We collected data from three different repositories that are publicly available. The data sources are as follows:

1) ENISA: The European Union Agency for Cybersecurity (ENISA) published a report in December 2019 titled State of Vulnerabilities 2018/2019 [28]. The report aimed to provide an insight into both the opportunities and limitations of the vulnerability ecosystem. They collected in total 27,471 vulnerability information published during 1st January 2018 to 30th September 2019 from various data sources. As part of the analysis of the collected data, the authors mapped the CVEs to the MITRE ATT&CK technique using the common CAPEC information found in both NVD and ATT&CK. The authors generously made available the dataset they have analyzed [29] and we utilized the CVE information mapped to MITRE ATT&CK tactics and techniques for training and testing the multi-label classification model.

The ENISA report dataset represents the vulnerability information in the form of CVE descriptions. It has 8,077 CVEs that are mapped to 52 unique MITRE ATT&CK techniques or, in this instance, labels. The mean length of the example CVE description is 368 characters, and minimum/maximum lengths are 40 and 3,655 characters long. For the purpose of this experiment, this dataset will be denoted as ENISA in short.

2) TRAM: Threat Report ATT&CK Mapping (TRAM) is a tool developed by MITRE to aid the analyst in mapping finished reports to ATT&CK. TRAM uses a Logistic Regression model to predict the mapping of the ATT&CK technique for a given report. MITRE generously released the source code and the corresponding dataset used to train the model [30]. The dataset contains example sentences or phrases representing specific techniques and maps them to one or more techniques.

The TRAM dataset represents the short threat information in the form of sentences or phrases. It has 3,005 example sentences mapped to 188 unique MITRE ATT&CK techniques. The mean length of the example sentence is 84 characters, and minimum/maximum lengths are 15 and 465 characters long.

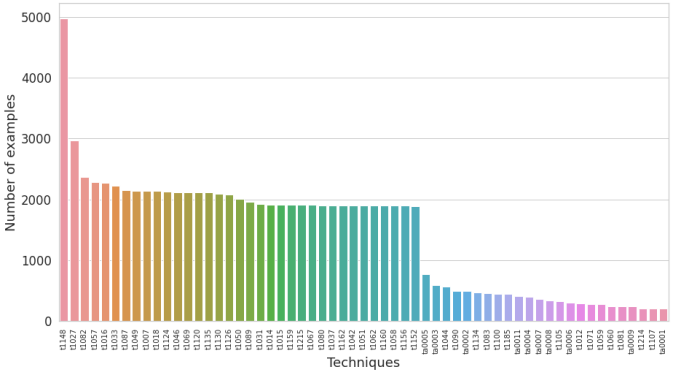


Figure 2. Most common techniques that have at least more than 200 examples associated with it.

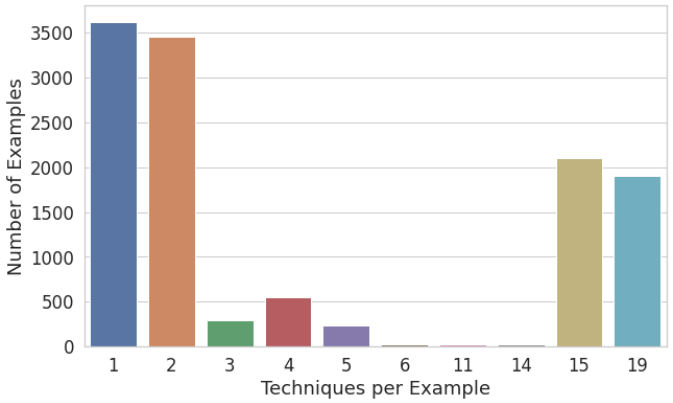


Figure 3. Number of techniques associated per example (top 10).

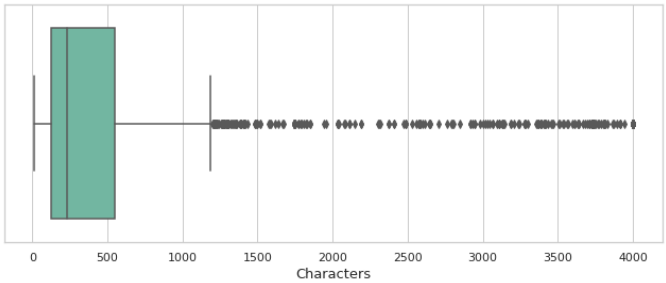


Figure 4. Text length distribution of the dataset.

For the purpose of this experiment, this dataset will be denoted as TRAM in short.

3) rcATT: Legoy et al. implemented a tool called rcATT, a system that predicts tactics and techniques related to given cyber threat reports and described it in [5]. They collected the threat reports referenced in the original MITRE ATT&CK framework per each individual technique to train the tool. They generously made their source code and the parsed threat reports publicly available, and we believe it is the most accurate data we could utilize in this experiment.

The rcATT represents the long descriptive information in the form of threat reports. It has 1,490 example reports mapped to 227 unique MITRE ATT&CK techniques. The mean length of the example report is 19,270 characters, and minimum/maximum lengths are 625 and 457,759 characters long. For the purpose of this experiment, this dataset will be denoted as rcATT in short.

Since the examples of the rcATT dataset are too long for embedding framework, we trimmed the examples to a maximum of 4,000 characters long to ease the computational burden.

B. Dataset analysis

From 3 repositories, we collected a total of 12,572 examples mapped to 239 unique techniques. The characteristics of each data repository and the combined dataset are shown in Table I.

TABLE I. DATASET PROPERTIES.

Dataset	*Examples*	*Techniques*	*Length*	*Type of example*
ENISA	8,077	52	Medium	CVE descriptions
TRAM	3,005	188	Short	Sentences
rcATT	1,490	227	Long	Threat reports
Combined	12,572	239	Mixed	Mixed

The combined dataset is unevenly distributed in terms of the technique association, and there are only 58 techniques with more than 200 examples associated. In Figure 2, the techniques with more than 200 examples associated are shown, and we can see that technique T1148 - HISTCONTROL has 4,978 examples associated with it.

The examples of the combined dataset also have varying characteristics. Figure 3 shows the top 10 instances of label associations. From Figure 3, we can see that majority of the examples, i.e., 7,072 examples out of 12,572, has either 1 or 2 labels associated with them.

The length of the text in the combined dataset also greatly varies. Figure 4 shows this variation in terms of the boxplot graph. Note that the rcATT examples have been trimmed to a maximum of 4,000 characters to ease the computational burden. From Figure 4, we can see that at least 75% (third quartile) of the examples are a less than 550 characters long, and the estimated maximum value would be around 1,200 characters long.

In terms of the content of the examples, a quick analysis using the wordcloud reveals that common words used in the vulnerability descriptions or threat reports are the words that are most repeated in the dataset. Figure 5 shows the wordcloud analysis.

V. Experimental Setup

Using the background information of Section III and the experimental dataset discussed in Section IV, we conducted the experiment to map the threat information to adversarial techniques. Depending upon the experimental setup, we designed the following two separate experiments.

- Converting the text information into its vector representation and apply traditional multi-label classification methods.
- Feed the BERT model with raw text to convert it into its internal representation and performs multi-label classification.

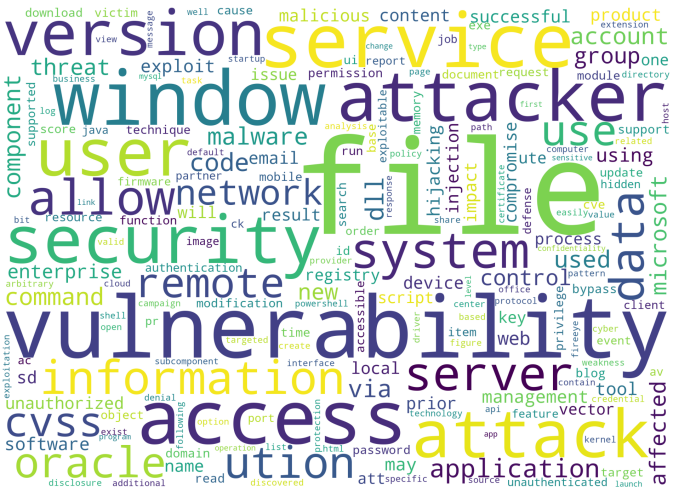


Figure 5. Most common words in the dataset.

In this section, we will discuss the details of these environmental differences.

A. Text representation in multi-label classification

To conduct the multi-label classification, we need to convert the given text into numerical vectors, also known as embeddings. Conventionally, vector embeddings were achieved through shallow algorithms such as Bag of Words (BoW) or Term Frequency-Inverse Document Frequency (TF-IDF). These approaches have been superseded by predictive representation models such as Word2Vec [31], GloVe [32], and so on. The utilization of deep neural networks has been proven to be superior in different fields. Thus, various studies have adopted deep neural models to embed the text into vector space, such as Facebook's InferSent [33] and Universal Sentence Encoder (USE) from Google Research. Perone et al. evaluated different sentence embeddings, and Universal Sentence Encoder outperformed InferSent in terms of semantic relatedness and textual similarity tasks [34]. Therefore, for this research, Universal Sentence Encoder has been utilized to generate the vector embeddings of the text.

The sentence embeddings from USE produce good task performance with little task-specific training data. Thus, we decided to utilize a Deep Averaging Network (DAN)-based USE model introduced in [35] to represent the threat information in numerical vectors so that traditional multi-label classification methods could be applied. The model takes English sentences of variable lengths as input and produces 512 fixed-dimensional vector representations of the sentences as output [36].

B. Multi-label classification model selection

The 512 fixed-dimensional vectors generated by USE are treated as features for the classifier. To determine the suitable model to map the threat information to MITRE ATT&CK techniques, we experimented with 1 Algorithm Adaptation, 3 Problem Transformation, and 1 Ensemble multi-label classification methods using the open-source library scikit-multilearn [37]. The experimented methods are listed below.

- **Multi-label k-Nearest Neighbors (MlkNN)** is the adaptation of the popular k-nearest neighbors (kNN) algorithm to the multi-label classification task and an example of the Algorithm adaptation method. We estimated the number of neighbors k to be most optimal when $k = 3$ where $1 \leq k \leq 30$ when optimized for macro-average F1 measure.
- **LabelPowerset** is a Problem Transformation method that transforms a multi-label problem into a multi-class problem with one multi-class classifier trained on all unique label combinations. It maps each combination to a unique id number and performs multi-class classification using the classifier as a multi-class classifier and combination ids as classes.
- **ClassifierChain** is also a Problem Transformation method. The classifiers are linked along a chain where the i-th classifier deals with the binary relevance problem associated with its label. The feature space of each link in the chain is extended with the 0/1 label associations of all previous links.
- **BinaryRelevance** is the well known one-against-all method. It learns one classifier for each label using all the examples labeled with that label as positive and remaining as negative. And while making a prediction, each binary classifier predicts whether its label is relevant for the given example or not. It is an example of the Problem Transformation method.
- **RAndom k-labELsets multi-label classifier (RAkELd)** is an Ensemble method that divides the label space into equal partitions of size k, trains a LabelPowerset classifier per partition, and predicts by summing the result of all trained classifiers.

The methods mentioned above use traditional classification algorithms for the multi-label classification task. Since neural networks have been proven to be superior in almost every task, we also experimented with more neural approaches as multi-label classification algorithms. Since the multi-label classification task of our experiment does not require the sequential input or memory state of the input, we experimented with a simple Multilayer Perceptron (MLP) neural model to conduct the classification. Szymański et al. included a wrapper in a scikit-multilearn library that allows any Keras or PyTorch compatible backend to be used to solve multi-label problems through problem-transformation methods [38]. We utilized it to conduct the same experiment with neural methods. The following lists the neural methods we used along with their basic parameters.

- **LabelPowerset (neural)** LabelPowerset method with the Multilayer Perceptron as the base classifier. It has two hidden layers, and the softmax function is used for activation.
- **BinaryRelevance (neural)** BinaryRelevance method with the Multilayer Perceptron as base classifier. It has two hidden layers, and the sigmoid function is used for activation.

In Section VI, we will discuss the evaluation results of these different methods.

C. BERT

BERT is designed to learn deep bidirectional representations from an unlabeled text by jointly conditioning both the

left and right contexts in contrast to the previous attempts of predicting a token in a unidirectional (left-to-right, right-to-left) way [26]. BERT achieves bidirectionality by using a pre-training objective called a Masked Language Model (MLM). Before feeding the text sequence to a model, BERT replaces 15% of the words in each training example with a [MASK] token. Then, the task of the model is to predict the original token based on the non-masked tokens. In addition to the MLM task, BERT also employs the Next Sentence Prediction (NSP) task, where the model takes a pair of sentences and then tries to predict whether the second sentence is subsequent to the first. The model is fed 50% of the subsequent sentences during the training while the other 50% of sentences are ordered randomly.

To successfully train with MLM and NSP tasks, BERT preprocesses the input text according to the following steps.

1) A special [CLS] token is placed at the beginning of the first sentence, and a [SEP] token is placed right before the second.
2) Token embeddings, where dense embeddings for each token, including [CLS] and [SEP], will be learned.
3) Sentence embeddings indicate which tokens belong to which sentence. This process is similar to token embeddings; however, vocabulary size is limited to only two words.
4) Positional embeddings are borrowed from the original transformer paper [24]. Since the transformer is not recurrent, it needs to learn the sequential information with the help of positional embeddings.

BERT has been pre-trained on BookCorpus (800M words) and English Wikipedia (2,500M words) with the goal of minimizing the combined loss of MLM and NSP tasks. Fine-tuning BERT on a classification task is relatively straightforward by simply adding a linear layer on top of the transformer output for the first [CLS] token. Applying BERT to downstream tasks involves fine-tuning for the task-specific data.

For the purpose of this experiment, we used BERT-Base uncased model, which contains 30,522 words. Since BERT uses the WordPiece tokenization method, out of vocabulary words are split into subwords, and a group of subwords represents the word. We adapted BERT's sequence classification class for the multi-label classification task using binary cross-entropy with a logits loss function. The final model consisted of the Input Embedding layer, 12 BERT attention layer, and output layer as per the number of labels to be predicted. We trained the model for ten epochs with a batch size of 8 and sequence length of 512. The learning rate was kept to 3e-5, as recommended in the original paper.

VI. Experimental result

Using the experimental dataset discussed in Section IV, we conducted four separate experiments, with each dataset and the combined dataset of 12,572 examples. All the models have been trained with randomly selected 66.6% examples of the dataset and tested on the remaining 33.3% examples. Since the dataset is limited in size, we evaluated the models with a 10-fold cross-validation method. The only exception is in the case of the BERT model due to its computationally costly nature. In this section, we will review the results of each experiment and analyze them.

A. ENISA dataset

The evaluation results for the ENISA dataset is listed in Table II.

From the results listed in Table II, we could see that LabelPowerset using the neural model as a base classifier, has the best scores in 3 of the 6 measures, and BERT has the best results in the remaining 3 measures. The nature of the ENISA dataset is that it consists of quite uniform medium-length text (CVE descriptions) and makes up more than 64% of the total dataset. Hence, the results of this experiment may have the most influence over the final outcome of the Combined dataset.

B. TRAM dataset

The evaluation results for the TRAM dataset is listed in Table III.

From the results listed in Table III, we could see that LabelPowerset using the neural model as a base classifier, has the best scores in every measure except Ranking loss, in which the neural BinaryRelevance model has a lead. Unfortunately, we were not able to produce any meaningful result with the BERT model using this dataset. Since the nature of the TRAM dataset is short sentences and phrases, we assume that BERT may not work very well with the short text. We reduced the sequence length of the BERT model from 512 characters to 64 characters, but it did not yield any improvements. Since the dataset has fewer examples associated with more labels than the ENISA dataset, the resulting performances also seem to have deteriorated.

C. rcATT dataset

The evaluation results for the rcATT dataset is listed in Table IV.

From the results listed in Table IV, we could see that no single model has superior performance in all the measures. Since the nature of the rcATT dataset is long threat reports, it could potentially improve the model performance. However, it is clear that all the multi-label classification models perform poorly compared to the previous two datasets in every except one measure. This poor performance could be because the size of the training and test data is small compared to the other datasets (999 and 491 examples, respectively), and fewer examples are associated with more techniques (1,490 examples with 227 techniques).

D. Combined dataset

The evaluation results for the Combined dataset is listed in Table V.

From the results listed in Table V, we could see that LabelPowerset using the neural model as a base classifier, has 3 out of 6 best results and BERT has 2, and neural BinaryRelevance has the best score in Hamming loss only. Since the combined dataset is the combination of 3 datasets, the performances of the models fall within a range of the best and worst results. We believe the results of the combined dataset are reasonable performance and considered it as the final experimental result.

TABLE II. ENISA DATASET EXPERIMENTAL RESULT.

		Micro Average			Macro Average					
Algorithm	*Accuracy score*	*Prec.*	*Rec.*	*F1 Score*	*Prec.*	*Rec.*	*F1 Score*	*Hamm. loss*	*Rank. loss*	Micro ROC
MlkNN	0.6079	0.7310	0.6193	0.6700	0.6225	0.4960	0.5403	0.1089	0.3617	0.7848
LabelPowerset	0.6028	0.7233	0.5640	0.6320	0.5756	0.5283	0.5014	0.1159	0.3754	0.7588
ClassifierChain	0.4993	0.5997	0.6201	0.6094	0.4219	0.4678	0.4224	0.1423	0.4330	0.7647
BinaryRelevance	0.3757	0.5937	0.6508	0.6205	0.4727	0.6096	0.4875	0.1424	0.3283	0.7765
RakelD	0.4235	0.6374	0.6133	0.6243	0.5070	0.5801	0.4937	0.1310	0.3446	0.7688
LabelPowerset (neural)	**0.7363**	0.7491	0.7464	0.7470	0.6681	0.6213	**0.6316**	0.0907	**0.2503**	0.8455
BinaryRelevance (neural)	0.5526	0.7821	0.7062	0.7415	0.6921	0.5873	0.6236	0.0882	0.2910	0.8313
BERT	0.7201	0.8269	0.7384	**0.7801**	0.6646	0.5930	0.6247	**0.0742**	0.2643	**0.8524**

TABLE III. TRAM DATASET EXPERIMENTAL RESULT.

		Micro Average			Macro Average					
Algorithm	*Accuracy score*	*Prec.*	*Rec.*	*F1 Score*	*Prec.*	*Rec.*	*F1 Score*	*Hamm. loss*	*Rank. loss*	Micro ROC
MlkNN	0.5095	0.7194	0.5196	0.6031	0.2437	0.2045	0.2117	0.0037	0.4753	0.7593
LabelPowerset	0.6370	0.6470	0.6370	0.6420	0.2439	0.2544	0.2371	0.0039	0.3579	0.8175
ClassifierChain	0.0519	0.0746	0.1054	0.0873	0.0149	0.0486	0.0185	0.0120	0.8975	0.5491
BinaryRelevance	0.2612	0.3481	0.6983	0.4642	0.1787	0.2840	0.1974	0.0088	0.3048	0.8456
RakelD	0.2639	0.3533	0.6976	0.4686	0.1804	0.2840	0.1989	0.0086	0.3053	0.8453
LabelPowerset (neural)	**0.6902**	0.7019	0.6938	**0.6978**	0.2856	0.2897	**0.2763**	0.0033	**0.3013**	**0.8461**
BinaryRelevance (neural)	0.5271	0.7853	0.5803	0.6671	0.2764	0.2311	0.2420	**0.0031**	0.4156	0.7897
BERT	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0054	1.0000	0.5000

TABLE IV. RCATT DATASET EXPERIMENTAL RESULT.

		Micro Average			Macro Average					
Algorithm	*Accuracy score*	*Prec.*	*Rec.*	*F1 Score*	*Prec.*	*Rec.*	*F1 Score*	*Hamm. loss*	*Rank. loss*	Micro ROC
MlkNN	0.0174	0.3920	0.2730	**0.3153**	0.0724	0.0572	0.0559	0.0349	0.7509	0.6303
LabelPowerset	0.0121	0.2394	0.0822	0.1170	0.0302	0.0147	0.0147	0.0364	0.8898	0.5371
ClassifierChain	0.0007	0.1381	0.5541	0.2192	0.0432	0.2077	0.0604	0.1173	0.6798	**0.7224**
BinaryRelevance	0.0000	0.0988	0.5894	0.1678	0.0574	0.2796	**0.0826**	0.1724	0.6946	0.7108
RakelD	0.0000	0.1232	0.4974	0.1947	0.0598	0.1926	0.0780	0.1197	**0.6777**	0.6940
LabelPowerset (neural)	**0.0302**	0.2824	0.2195	0.2434	0.0918	0.0680	0.0687	0.0402	0.7848	0.6015
BinaryRelevance (neural)	0.0054	0.4737	0.2296	0.3058	0.0717	0.0431	0.0484	0.0316	0.7896	0.6109
BERT	0.0000	0.6042	0.0679	0.1222	0.0163	0.0063	0.0086	**0.0187**	0.9757	0.5335

TABLE V. COMBINED DATASET EXPERIMENTAL RESULT.

		Micro Average			Macro Average					
Algorithm	*Accuracy score*	*Prec.*	*Rec.*	*F1 Score*	*Prec.*	*Rec.*	*F1 Score*	*Hamm. loss*	*Rank. loss*	Micro ROC
MlkNN	0.5460	0.7367	0.6104	0.6675	0.3183	0.2235	0.2501	0.0182	0.4044	0.8018
LabelPowerset	0.5468	0.6884	0.5162	0.5898	0.2975	0.1926	0.2124	0.0214	0.4255	0.7545
ClassifierChain	0.0230	0.0598	0.3319	0.1013	0.0439	0.2396	0.0444	0.1763	0.7215	0.5854
BinaryRelevance	0.1293	0.2448	0.7424	0.3681	0.1099	0.5133	0.1537	0.0762	0.3464	0.8359
RakelD	0.1258	0.3067	0.7139	0.4289	0.1313	0.4172	0.1738	0.0568	0.3356	0.8321
LabelPowerset (neural)	**0.6631**	0.7093	0.6948	0.7018	0.3353	0.2524	0.2701	0.0177	**0.3123**	**0.8430**
BinaryRelevance (neural)	0.4850	0.7776	0.6754	0.7229	0.3424	0.2480	**0.2735**	0.0155	0.3548	0.8347
BERT	0.4464	0.8420	0.6817	**0.7534**	0.1676	0.1363	0.1453	**0.0130**	0.5274	0.8389

E. Result analysis

The best results of the experiments measured by Micro ROC are listed in Table VI along with the models which have shown the best performance in at least two measures. The results of Table VI and the dataset properties listed in Table I show that the performances of the multi-label classification deteriorate when the number of labels, in this instance, techniques to be predicted, increases and could depend highly on the number of examples used. Also, the length of the text matters in which shorter text has a higher probability of correct technique to be predicted.

We could conclude from the listed results that the LabelPowerset method with Multilayer Perceptron as base classifier performs best in most cases, and BERT comes in second. Since running a BERT model requires a computationally expensive environment, the neural LabelPowerset model would be an ideal choice for predicting the adversarial techniques in cyber threat information.

TABLE VI. COMBINED DATASET EXPERIMENTAL RESULT.

Dataset	Best Micro ROC	Best model
ENISA	0.8524	LabelPowerset (neural), BERT
TRAM	0.8461	LabelPowerset (neural)
rcATT	0.7224	None
Combined	0.8430	LabelPowerset (neural), BERT

VII. CONCLUSION

This paper proposed an approach to automatically map the cyber threat information to adversary techniques in the cybersecurity context. We converted various threat information into vector space and experimented with different multi-label classification methods, as well as a state-of-the-art language model to identify the most suitable method to map the threat information into MITRE ATT&CK adversarial techniques. We used 12,572 examples from 3 open datasets to conduct 4 independent experiments to train and test 7 multi-label classification methods and 1 pre-trained language model in 6

evaluation measures.

According to the results of the experiments converting threat information into vector space using a pre-trained USE model and applying the neural LabelPowerset method to conduct multi-label classification to predict adversarial techniques yielded the best results. State-of-the-art pre-trained language model BERT performed second in which BERT's internal embedding represents the threat information, and the final layer is fine-tuned for multi-label purposes.

Based on the experiment results, we believe that by embedding the given threat information using Universal Sentence Encoder and applying the LabelPowerset method with Multilayer Perceptron as a base classifier, we could effectively predict the adversarial techniques.

However, we acknowledge the limitations of the paper, including the lack of explainability in the models so that the labels it predicted could not be entirely analyzed with current settings. Also, there has been some progress with the NLP and language models since the experiments were first designed, thus not reflected in this work. Those limitations are to be addressed in future work.

REFERENCES

[1] O. Mendsaikhan, H. Hasegawa, Y. Yamaguchi, and H. Shimada, "Automatic mapping of vulnerability information to adversary techniques," in The Fourteenth International Conference on Emerging Security Information, Systems and Technologies (SECURWARE). IARIA, 2020, pp. 53–59.

[2] National Vulnerability Database, 2020 (accessed June 1, 2020). [Online]. Available: https://nvd.nist.gov/general/nvd-dashboard

[3] D. Bekerman and S. Yerushalmi, The State of Vulnerabilities in 2019, 2020 (accessed June 10, 2020). [Online]. Available: https://www.imperva.com/blog/the-state-of-vulnerabilities-in-2019/

[4] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, "MITRE ATT&CK®: Design and philosophy," The MITRE Corporation, Tech Report, 2020.

[5] V. Legoy, M. Caselli, C. Seifert, and A. Peter, Automated Retrieval of ATT&CK Tactics and Techniques for Cyber Threat Reports, 2020, vol. abs/2004.14322.

[6] G. Husari, E. Al-Shaer, M. Ahmed, B. Chu, and X. Niu, "Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources," in Proceedings of the 33rd Annual Computer Security Applications Conference, ser. ACSAC 2017. New York, NY, USA: Association for Computing Machinery, 2017, p. 103–115. [Online]. Available: https://doi.org/10.1145/3134600.3134646

[7] K. Oosthoek and C. Doerr, SoK: ATT&CK Techniques and Trends in Windows Malware, 2019, pp. 406–425.

[8] R. Al-Shaer, J. M. Spring, and E. Christou, Learning the Associations of MITRE ATT&CK Adversarial Techniques, 2020.

[9] G. Huang, Y. Li, Q. Wang, J. Ren, Y. Cheng, and X. Zhao, "Automatic classification method for software vulnerability based on deep neural network," IEEE Access, vol. 7, 2019, pp. 28 291–28 298.

[10] E. Hemberg, J. Kelly, M. Shlapentokh-Rothman, B. Reinstadler, K. Xu, N. Rutar, and U.-M. O'Reilly, "Linking threat tactics, techniques, and patterns with defensive weaknesses, vulnerabilities and affected platform configurations for cyber hunting," 2021.

[11] S. Zhou, Z. Long, L. Tan, and H. Guo, "Automatic identification of indicators of compromise using neural-based sequence labelling," 2018.

[12] S. R. Medina, "Multi-label text classification with transfer learning for policy documents," 2019.

[13] F. Sovrano, M. Palmirani, and F. Vitali, "Deep learning based multi-label text classification of unga resolutions," in Proceedings of the 13th International Conference on Theory and Practice of Electronic Governance, ser. ICEGOV 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 686–695. [Online]. Available: https://doi.org/10.1145/3428502.3428604

[14] J. Liu, W.-C. Chang, Y. Wu, and Y. Yang, "Deep learning for extreme multi-label text classification," in Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, ser. SIGIR '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 115–124. [Online]. Available: https://doi.org/10.1145/3077136.3080834

[15] A. Pal, M. Selvakumar, and M. Sankarasubbu, "Multi-label text classification using attention-based graph neural network," in ICAART, 2020.

[16] J. Lee and J. Hsiang, "Patentbert: Patent classification with fine-tuning a pre-trained BERT model," CoRR, vol. abs/1906.02124, 2019. [Online]. Available: http://arxiv.org/abs/1906.02124

[17] A. Adhikari, A. Ram, R. Tang, and J. Lin, "Docbert: BERT for document classification," CoRR, vol. abs/1904.08398, 2019. [Online]. Available: http://arxiv.org/abs/1904.08398

[18] Common Vulnerabilities and Exposures, 2020 (accessed June 25, 2020). [Online]. Available: https://cve.mitre.org/

[19] National Vulnerability Database, 2020 (accessed June 22, 2020). [Online]. Available: http://nvd.nist.org

[20] About CAPEC, 2020 (accessed June 25, 2020). [Online]. Available: https://capec.mitre.org/about/index.html

[21] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," IJDWM, vol. 3, 2007, pp. 1–13.

[22] G. Madjarov, D. Kocev, D. Gjorgjevikj, and S. Deroski, "An extensive experimental comparison of methods for multi-label learning," Pattern Recogn., vol. 45, no. 9, 2012, p. 3084–3104. [Online]. Available: https://doi.org/10.1016/j.patcog.2012.03.004

[23] Kaggle: Your Machine Learning and Data Science Community, 2021 (accessed June 20, 2021). [Online]. Available: https://www.kaggle.com/

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," CoRR, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[25] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[26] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," CoRR, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[27] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," 2020.

[28] V. Katos, S. Rostami, P. Bellonias, N. Davies, A. Kleszcz, S. Faily, A. Spyros, A. Papanikolaou, C. Ilioudis, and K. Rantos, "State of vulnerabilities 2018/2019," European Union Agency for Cybersecurity (ENISA), Tech Report, 2019.

[29] Threat Report ATT&CK Mapping (TRAM) is a tool to aid analyst in mapping finished reports to ATT&CK, 2020 (accessed May 1, 2021). [Online]. Available: https://github.com/mitre-attack/tram

[30] TRAM, 2019 (accessed May 10, 2020). [Online]. Available: https://github.com/enisaeu/vuln-report/

[31] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in 1st International Conference on Learning Representations, 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[32] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[33] InferSent, 2018 (accessed January 10, 2020). [Online]. Available: https://github.com/facebookresearch/InferSent

[34] C. S. Perone, R. Silveira, and T. S. Paula, "Evaluation of sentence embeddings in downstream and linguistic probing tasks," CoRR, vol. abs/1806.06259, 2018. [Online]. Available: http://arxiv.org/abs/1806.06259

[35] D. Cer, Y. Yang, S. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y. Sung, B. Strope, and R. Kurzweil, "Universal sentence encoder," CoRR, vol. abs/1803.11175, 2018. [Online]. Available: http://arxiv.org/abs/1803.11175

[36] universal-sentence-encoder, 2018 (accessed January 11, 2020). [Online]. Available: https://tfhub.dev/google/universal-sentence-encoder/2

[37] Multi-Label Classification in Python, 2018 (accessed May 15, 2020). [Online]. Available: http://scikit.ml/

[38] P. Szymański and T. Kajdanowicz, "A scikit-based Python environment for performing multi-label classification," ArXiv e-prints, 2017.

Empirical Analysis of Trustworthiness Attributes in the Context of Digitization

Sandro Hartenstein
OvGU Magdeburg / HWR-Berlin
Berlin, Germany
e-mail: sandro.hartenstein@hwr-berlin.de

Steven Schmidt
DB Station&Service AG / OvGU Magdeburg / HWR-Berlin
Berlin, Germany
e-mail: s_schmidts19@stud.hwr-berlin.de

Andreas Schmietendorf
HWR-Berlin
Berlin, Germany
e-mail: andreas.schmieten-dorf@hwr-berlin.de

***Abstract*— This paper describes the concept, implementation and first results of a multidimensional research approach to improve the trustworthiness of digital services. It presents the current perception of the concepts of trust and trustworthiness in technical and sociological systems, and their connection as an identified gap. Well-known environmental analysis is used to define the dimensions. The empirical investigations are designed separately for each dimension, or domains of study. The goal is to subsequently create a holistic and robust concept for trustworthy socio-technical systems.**

Keywords - Trustworthiness; Digitization; Information Systems; Society.

I. INTRODUCTION

This paper expands the view on the concept, realization and first results of the idea, which was originally and briefly presented at the *The Fourteenth International Conference on Digital Society* 2020, for the empirical analysis of digital services in the context of digitization [1]. The variety of services offered in the digital world is constantly evolving and rapidly increasing due to the establishment of digital aspects in everyday private and professional life. The use of digital services is highly dependent on trustworthiness [2] [3] [4].

However, the concepts of trust and trustworthiness are understood differently in different academic and industrial disciplines, as are the attributes associated with them.

This idea paper aims to present a possible approach to analyze significant factors of trustworthiness through various empirical studies from different fields. The trustworthiness attributes can vary widely from discipline to discipline. In general, it is assumed that these attributes differ mainly only in their weighting, related to the observed discipline for that they are relevant.

This document is divided into six sections. Section I contains the brief introduction. In Section II, different terms and viewpoints on the topic of trust and trustworthiness are described to explain the motivation for this approach. In Section III, past and current related work is then examined, to demonstrate the variations of the current understanding and related contexts that have been evaluated. Section IV describes what could be done to achieve a generic and general model of trustworthiness attributes and associated weights according to the area under study. The conceptional procedure to accomplish this idea is described in detail, as well as what fields are going to be involved as part of the planned project to enable this work. The fields and their individual empirical approach, thus, are presented briefly to demonstrate the general idea of the approach. Section V presents the early results. The outcomes of the analysis of WebAPIs and the survey for public wifis are visualized and explained in the context of trustworthiness. The Section VI contains a summary of the current status of the project and briefly lists the open work packages.

II. TERMS & VIEWPOINTS

An agreed-upon definition of trust in the context of digital services is:
"Trust by definition entails a willingness by the [trustor] to make herself vulnerable to the possibility that another will act to her detriment" [5, p. 28], which is also based on the sociological perspective on trust as an fulfilment of expectations towards a person or a system by taking risks [6] [7] [8] [9].

An acceptable definition of trustworthiness in the context of digital services therefore is formed over time and relative to the perception of certain, system specific attributes. Generally, trustworthiness of systems can be defined as being based on ability, benevolence and integrity of the system [10] which correlates with the development of *predictability* over *dependability* to *faith* over time, regarding expectations towards the *persistence*, *technical competence* and *fiduciary responsibility* of a system as shown in Lee and Moray [7], p. 1245. Related approaches tend to a similar characteristic [11] [12] [13].

Digitization depends on the well-being of users. Entrusting data and work steps to a computer system will be criticized by users. In addition to advantages, there are also disadvantages. Trust is the key to accepting digital services, and therefore the key to increasing productivity through digitalization. This creative paper shows the dimensions of trust. These needs are resolved by the supplier. There are several participants with different interests and understandings of trust and trustworthiness.

In the context of the credibility of digital services, the needs of stakeholders are consumers, providers and third-party trustees. Consumers are trying to use services that are as trustworthy as possible, because the impact of data abuse is becoming more and more obvious. Digital service providers need consumer confidence in their products. They also need reliable supply services. The third independent authority can confirm the credibility of the digital service to the user, as long as it has the confidence of the user and can verify the service. From a service point of view, there are two main factors that play a decisive role in its reputation among consumers. User trust and service credibility are these two factors.

In a research project called OPerational Trustworthiness Enabling Technologies, in short OPTET, the prerequisites for

trust in the context of Web-based services were determined. The result is that trust can be personal, transferable, and based on core trust, such as in an organization. The credibility of the service is based on its attributes and the attributes confirmed by third parties. Figure 1 summarizes these correlations and their impact [14] [15] [16].

This research focuses on the social, economic, and technological factors that influence trust in digital services, as an implementation of the proposal we presented at the ICDS conference [1]. Based on the analysis of trust and trustworthiness, the following influencing factors can be determined. Social factors can be distinguished by personal, recommendation, and derived trust. Personal trust is characterized by emotions, such as browser authentication status color (red-dangerous, green-good) or knowledge, such as knowledge about two-factor authentication procedures. Recommended trust is based on trusted third parties. Derivative trust is usually formed by the experience of the organization and its status.

The technical factor is the credibility attribute of the service. These should be objectively measured or confirmed by a third party during the development and operation process. Economic factors are characterized by profit expectations.

The provider aims to provide reliable digital services. He can achieve this by optimizing all factors, but each factor must have a minimum level. For example, a certain service may be technically perfect, that is, completely credible, but the provider's reputation is poor, so the derived trust is low, and the service is not entirely credible. One factor that affects consumers is no risk or low risk. If the risk is lower, the service will be more trusted because the potential loss is controllable. However, many users do not realize the value of user data. Therefore, risk assessment is useful for all stakeholders.

Trust in digital services has been shaped by different impacts. The identified influences are personal trust, referral trust and trust in the institution. These findings are based on McKnight's model of trust [17] and Robbins' trust-risk-act model [18] and is visualized in Figure 1. The trust models are briefly explained in the Section III.

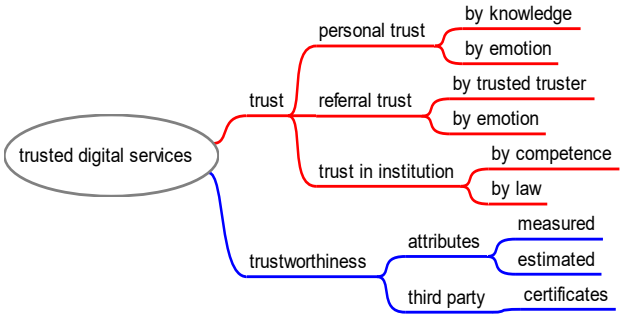


Figure 1. Trust and trustworthiness for digital services [own representation based on [15] [16] [14] [19].

In our view, a holistic view of trustworthiness in the context of digital services that addresses more than quality and security is missing. The possibilities for the technical consideration of services are limited, since only the interfaces are known, not their execution code.

The structure of the well-known Social, Technological, Economic, Environmental, Political, Legal, and Ethical Environment analyses, or STEEPLE, was used to classify the trust-building measures as a view from the outside [20, pp. 80-84]. From the authors' perspective, the environment analysis for a digital service is essential for its trustworthiness and consumer confidence.

In our discussion of trust patterns, we presented the conceptual considerations that trust in WebAPI-based architectures consists of more than just security aspects. We showed that trust develops from personal trust, e.g., in the provider or the technology and trustworthiness of the system. We have classified the trustworthiness attributes ac-cording to product, process and resource in order to determine the appropriate indicators for trust patterns. The categorization is based on the influence on the trust and trustworthiness, which is based on trust aspects [21]. For the OpenAPIs Trustability Parser we also use this classification. The challenge in terms of determining individual values of the attributes, since many indicators are transparent.

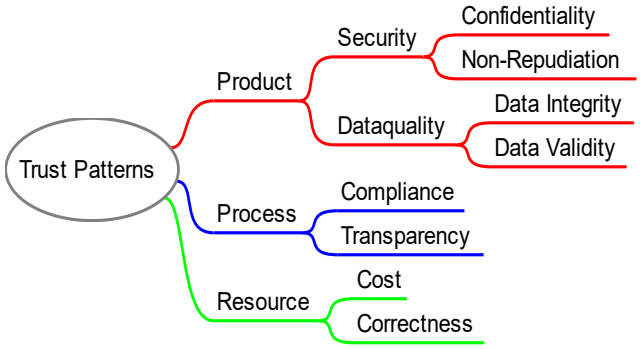


Figure 2. Classification of trustworthiness attributes for trust pattern [21].

III. RELATED WORK

Research on trust has been around for a long time; the basics have been interesting since the 1950s. Recent research has commercial reasons [22]. For trust in software and its use, this section briefly introduces the most important concepts.

The social driver of trust is the honesty, integrity and reliability of the interactive partner. Solving these relationships is the essence of trust. This is also the foundation of social system and market stability. There is no doubt that trust is the basis of everyday interaction.

In early considerations, trust was measured against expected results [23]. If it is good, trust will be established. If the situation is not good, trust will be destroyed. Later, the emotional aspects and the behavior of the participants were identified as important influencing factors [24]. The change in perceptual ability seems to occur mainly in citizens with high trust and little knowledge, and the change in perceived benevolence mainly occurs in citizens with low knowledge and low trust [25].

In a business setting, the cognitive and emotional dimensions of trust were found to be powerful, independent, and interrelated when it comes to building trust relationships with businesses [22]. In a study of the relationship with public institutions, it was found that they are considered more trustworthy than private companies [26]. In a 2001 consideration, all aspects and the implications for trust are integrated into a single design. This is illustrated in Figure 2. Basically, he distinguishes trust in institutions through psychology and sociology that affect personal trust.

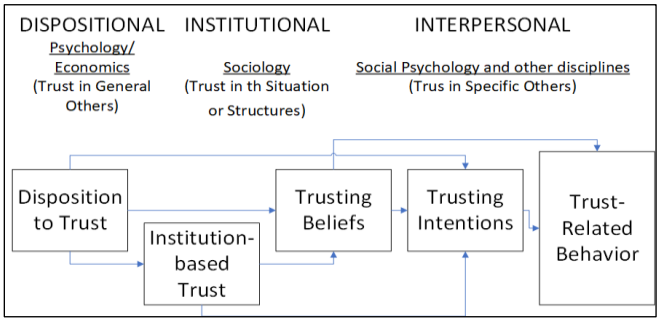


Figure 3. Interdisciplinary model of trust constructs [recreated from 17, p. 33].

An interdisciplinary model of trust was proposed as a modern trust-risk-behavior model, called relational trust [18, p. 985]. It is illustrated in Figure 3 and visualizes the links between trust, risk assessment, and relationships with activities. The factors that affect trust are the characteristics of the actors and the relationship between the actors and external parties.

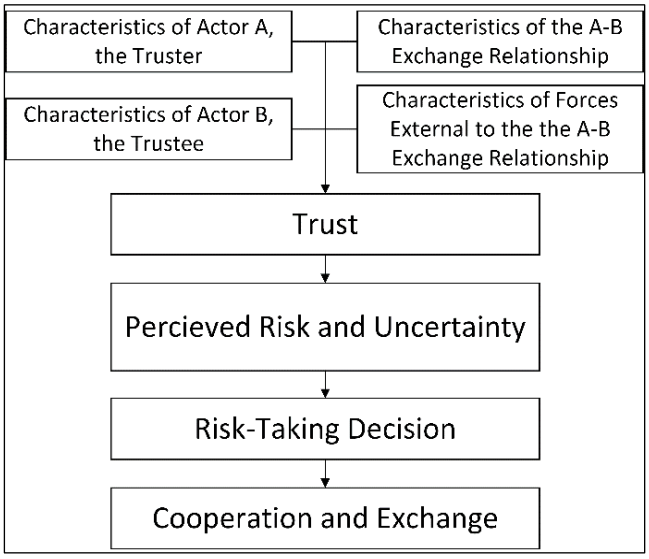


Figure 4. Structural-cognitive model of trust. [recreated from 18, p. 982].

In the OPTET research project, attributes for trustworthy software were compiled from literature on existing frameworks and surveys in software companies [19, pp. 546-547]. Many attributes were found that represent the nature of a web-based application in terms of its trustworthiness. Since these proper-ties must be evaluated at each point in the life cycle of the application, the main phases with the respective artefacts. A distinction is made between development, marketplace and runtime. In the development phase, the source code is available for analysis. In marketplace phase the software is compiled but not in use. The runtime phase means that the software is operational. Our analysis refers to available WebAPIs, therefore in the runtime context.

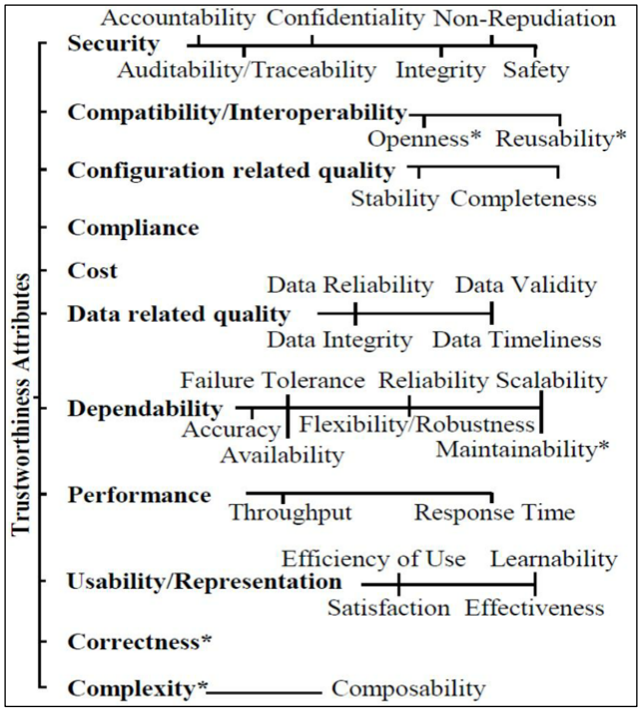


Figure 5. Trustworthiness attributes [27, p. 236].

These attributes have a context-specific impact on credibility. The fields and types of socio-technical systems, referred to as STS, are related. These attributes are measurable and can be influenced by weight mapping.

The top three attributes identified in a study of 72 relevant papers are security, dependability and usability. In almost 2/3 of the literature, security is mentioned as the most important attribute. Almost half of them mentioned reliability. One-quarter of the paper mentions usability as an important attribute of credibility. Figure 5 shows all the attributes and their dependencies [21, p. 25].

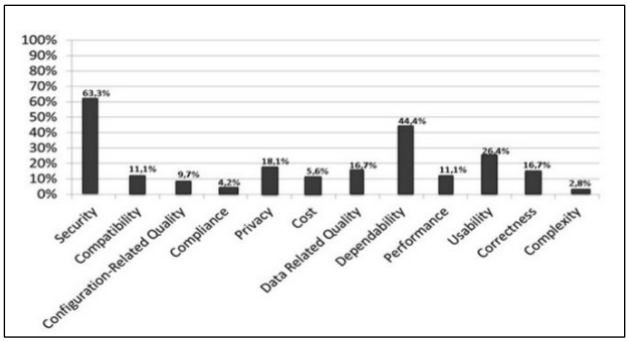


Figure 6. Classified trustworthiness attributes [28, p. 24].

Trustworthiness is an objective property of the WebAPI based primarily on security and quality attributes, but also includes specific attributes such as Complexity, Cost, Privacy, and Compliance [16, p. 14].

Trust is a subjective assessment and differs from the perspective of the respective stakeholder. This assessment is based on knowledge, emotions and expectations of the observer [29, p. 4]. Knowledge in relation to trust states that one understands the rights and duties of a provider and can also understand the measures taken to secure the requirements. Emotion in relation to trust states that one trusts the provider from an inner feeling without knowing the exact measures. The personal risk assessment of a possible misconduct is subjective. Most often, guarantees of the provider are included. The expectations in relation to trust are characterized by the hope that the system will perform the task with trust, because, for example, the function execution is important for the user and he has effort when changing providers [16, p. 11].

APIs, application programming interfaces, are important elements of software development for transparently coupling functionalities, resources and components. Hereby many requirements are fulfilled to the software development. On the one hand can be ensured opposite monolithic systems, an exchangeability and the re-use of particular parts. On the other hand, functions can be outsourced. If this takes place over the public Internet and the functionality as services are offered, we mean WebAPIs. The WebAPIs Economy has several actors. There is the provider, which offers at least one service through a public interface. There is a consumer that integrates these services into its software. This is not limited to one service or one provider. The added value is generated just by combining different services. In between, there can be API brokers that consolidate providers and consumers so that one of the two needs only one point of contact. Furthermore, there is the end user, who uses the consumer's finished application or service [30, p. 19].

The OpenAPI Trustability Parser can support all stakeholders in terms of evaluating the trustworthiness of WebAPIs. For this purpose, it takes the view of the consumer. WebAPIs are implemented for and by software developers and can be used as glue to hold together an increasingly digital world. They shall be specified in a suitable manner so that both the tasks of development-side composition and operationally used communication are supported [30, p. 12].

In our view, a good specification can also provide many indications and statements about individual aspects of trustworthiness. Therefore, the specifications are examined according to the OpenAPI specification [31]. The OpenAPI specification defines a standard for describing Restful APIs. It is promoted by the OpenAPI Initiative, which is supported by the Linux Foundation. Members include Google, IBM, Microsoft and SAP. Operationally, it is implemented by swagger.io, for example [32]. A public directory of WebAPIs according to OAS is provided by APITree.com, for instance [33].

Despite user rates for Wi-Fi access outside people's homes increasing across a range of countries including Germany [34] [35], the limited data published since 2015 indicates user rates only just exceeding 50%, with levels in 2015 at 39% [36] and 55% in 2018 [37] or lower [38]. Usage varies according to provider, with cafes and restaurants (77%) and hotels (88%) at the vanguard [36]. In 2018, user rates shifted slightly in favor of transport infrastructure, with public transport at 60% and mainline railways at 59% [37].

To actually make use of the proposed approach derived from the results of this work for the conception of digital services, the field of Requirements Engineering becomes important. As shown in [39], the field of Service Engineering fits as the wider context, whereas aspects such as Software Engineering or even product Engineering are not to be excluded [40, p. 102]. The general aim is to develop services, that deliver a value to a asking unit as a result or product, by generating a use of potentials and processes of a offering unit whilst market factors are respected [40, p. 58].

The evaluation of the current common systemic view on the matter of the conception of digital services in [39] shows a conceptual lack of possible elicitation, evaluation or management mechanics in current Requirements Engineering approaches within Service Engineering regarding trust building requirements. The underlying definitions of Requirements Engineering for this matter is the knowledge, documentation, specification and management of relevant requirements through process orientation, stakeholder focus and the evaluation of risk and value considerations [41]. This is something that has not yet been applied to the fundamental problem described earlier in [1].

IV. CONCEPT

An environmental perspective becomes important, as different context fields with different environmental factors and thus attuites and requirements a present in this generic approach. The PEST model by [42] originally took four environmental perspectives into account for the analysis: Political, economic, social and technological influences. Younger perspectives extended this approach by ecological, legal and ethical dimensions [20].

In an economical sense this analysis enables market insights as foundation for strategic decisions regarding marketing aspects [43, p. 238]. Regarding the aim of the work described in this paper, it shall serve as an orientation and foundation to cluster requirements coming from or being related to the dimensions.

The concept provides for a multidimensional model of trustworthiness based on the shown STEEPLE environment analysis. The architecture is intended to be designed so that it can be extended by further research in various fields across the sociotechnical spectrum by looking at the STEEPLE dimensions. The initial fields and respective systems considered are as follows:

- **S_1 - Trustworthy WebAPIs.**
  The consideration focuses on the collection and analysis of various trustworthiness-enhancing attributes of WebAPIs. The goal is to investigate the weighting of the different attributes in order to be able to define baseline requirements for digital services.
- **S_2 - Trustworthy public WiFi.**
  The empirical investigation in this area focuses on trustworthiness attributes in public WiFis through questionnaires and comparative interviews between user groups of different services with different trustworthiness levels.
- **S_3 - Trusted AI Web Services.**
  This research area is concerned with identifying and evaluating the trustworthiness of web services that use artificial intelligence in addition to S1, due to more complicated aspects of trustworthiness when it comes to AI approaches.
- **S_4 - Trusted web services of intermediaries.**
  Similar to the previous area, a variety of web presences of self-established mediators will be studied to gain a collection of empirically validated trustworthiness attributes in this area to enrich the proposed overall model with weights unique to this area.

Subarea S_1 deviates from the original idea of simulating the development process [1], since a simulation was assessed as unsuitable upon closer examination. On the one hand, the empery is very limited and on the other hand a theoretical model is already necessary for a simulation. However, this is the goal of the study. Simulation can be used later to evaluate the findings.

Figure 7 shows a schematic representation of the proposed research objective. Each empirically determined trustworthiness attribute (Ai) is to be weighted per system under study (Sj). In addition, these attributes will be categorized according to the STEEPLE dimensions, allowing the formation of clusters. This is helpful to understand system specifications and build a general model. Any further investigation of similar or other systems will add to the overall set, but will also add information about different weights that are unique per area studied. This enables a generic overview over relevant attributes but also a specific derivation for similar systems under consideration in for example Requirements Engineering Frameworks, as this poses a fundamental view of relevant trustworthiness requirements in this field.
In the following two subsections, we present the sub concepts for S_1 and S_2. The S_3 and S_4 sub concepts are currently in the design phase.

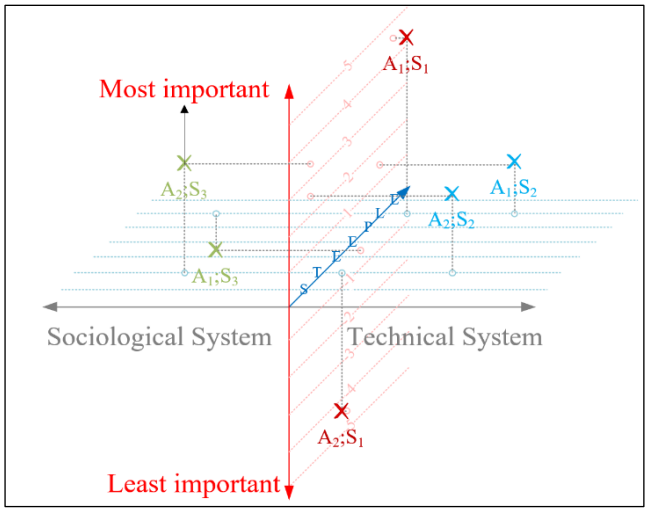


Figure 7. Visualization of the proposed approach.

A. *Concept S_1*

For the first subarea S1 of the project, the empirical investigation of WebAPIs in the context of trustworthiness, the following was designed: The idea is to be able to evaluate the WebAPIs in a structured way according to trust and trustworthiness by parsing OpenAPI specifications. For this purpose, it is necessary to identify indicators for determining the individual attributes and to query them during parsing. Basically, there are two types of indicators that are used for evaluation in relation to trustworthiness attributes. On the one hand, there is the quantitative and on the other hand the qualitative.

The quantitative indicators are countable methods, parameters and data types of the specification. For example, it is relevant how many primary data types and un-structured data types are used. From this, for example, a statement can be made about Data Integrity and its Data Validity. Primary data types are easier to check for integrity in contrast to complex data structures. The attack surface can be deter-mined from the number and type of methods provided. For example, read operations via GET method have much less malicious potential than POST methods. The indicators are used for calculation with the help of metrics in the analysis step. An evaluation can then be made from their results in comparison with defined reference values.

The evaluation of qualitative indicators is significantly more complex, since on the one hand requirements are checked against current requirements and on the other hand several indicators have to be combined. For example, the requirements for authentication and authorization are a good indicator for security, e.g., Confidentiality and Non-Repudiation. The evaluation of the indicator consists of several parts, like the technology, key length and cipher modes. Also, evaluating attributes from the categories of performance, usability and complexity is only possible with qualitative evaluation of the requirements to parameters by the specification. Knowledge of the individual data type is helpful, but not sufficient. Best practices and standards are to serve as reference values for this purpose. Simple metrics are not sufficient at

this point, so that further procedures must be used, such as Cosmic function points, as described by us at the IWSM conference [44], or AI analyses.

For these objectives, we have created the following work plan. First of all, it is necessary to find a suitable parser that can capture all indicators and is also integrable. It should also be open and independent of the analysis and evaluation module. In the second step, the parser and a metric analysis should be created with the help of a proto-type. In the third step, the analysis capabilities of the support tool will be exam-ined in a proof of concept with the help of a concrete scenario. In the fourth step, the evaluation of the trustworthiness in the analysis part will be extended.

In the first step, the optimal parser framework was determined. For this purpose, the criteria were defined and evaluated in a decision matrix for each candidate. This is presented in Table 1. The candidates are three Java libraries and two JavaScript modules, where one is deployed as a command line application. The criteria are the supported OpenAPI Standard versions, technical requirements, quantitative and qualitative analysis capabilities. By technical requirements we mean the possibilities to integrate the framework into our tool-chain. Quantitative analysis involves the evaluation of amounts of data types and methods in the specification un-der study. Qualitative analysis includes the capabilities to detect specific methods, such as authentication and authorization, and evaluate them. It also includes the detection of redundant and unnecessary methods and data types. With the help of the respective documentation and test implementations, we have determined that all java candidates fulfill the functional requirements. There are differences in handling and documentation. For this reason, the Swagger parser was selected for the proof of concept. An own developed parser was also considered, but due to the non-specific requirements for trustworthiness in the parsing activity, this was discarded.

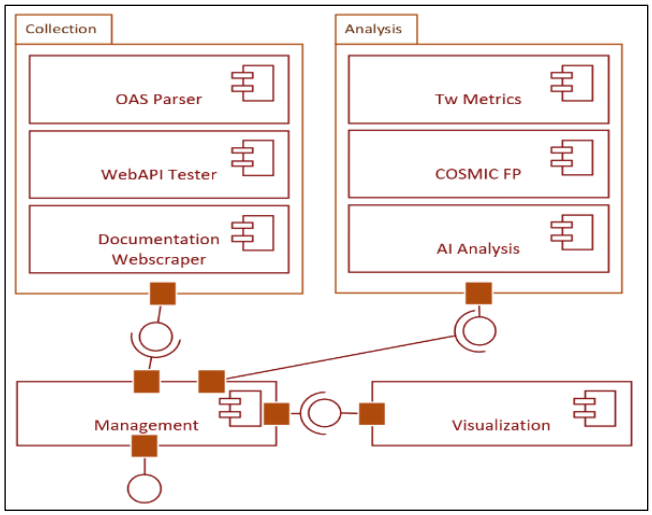


Figure 8. Scheme of software architecture for empirical data collection and processing.

The parser is an important part in the full toolchain for the evaluation of WebAPIs in context of trustworthiness. Additional components are also required for analysis, visualization and management. We chose microservices as our architecture model because it allows us to flexibly integrate different collection and analysis methods. The WebAPIs to be analyzed are captured and tracked via a management component. The architecture concept is shown in Figure 8. The analysis components can also be extended later in this way, so that cosmic function points analysis (COSMIC FP) and machine learning (AI Analysis) can be involved in addition to trustworthiness metrics (Tw Metrics). This architecture also makes it possible to integrate and combine other data collection methods, web scraping and testing.

TABLE 1 DECISION MATRIX FOR PARSING FRAMEWORK

Candidate	Type	OAS	technical requirements	quantitative analysis	qualitative analysis
Swagger-parser [45]	Java library	2 3	can be integrated into Java app; Restful support thereby possible	Simple, all documented methods	Security requirements can be checked if they are present in the OpenApi definition.
Openapi4j [46]	Java library	3	can be integrated into Java app; Restful support thereby possible	Medium, change the specification and subsequent serialization	Security requirements can be checked
KaiZen OpenApi Parser [47]	Java library	3	can be integrated into Java app; Restful support thereby possible	Complex, creates an object, which can be queried. Queries must be created.	Security requirements can be checked if they exist in the object.
Openapi-format [48]	Javascript CLI	3	can be used as a module, thus can be integrated into a NodeJS framework	Complex, prints all methods on the console	Security requirements cannot be checked.
openapi-snippet [49]	Javascript Container	2 3	can be integrated into a JavaScript web framework	Simple, returns an array of the methods	Security requirements cannot be checked.

B. Concept S_2

The following was designed for the S2 strand of the project, the empirical investigation of the relevance and perception of trustworthiness in the context of public WiFi as an example of an exposed digital public service.

The idea of the study was to investigate beneficial and restricting factors of public WiFi usage. Existing studies showed a low average usage rate of public WiFis of around 50% of potential users [36] [50] [37]. The evaluation then was supposed to concentrate on the reasons for and against the usage by forming an online questionnaire, which was conducted in Germany. A key element was the representativeness in a demographical way, so results would be accountable for the whole sociological picture. Content wise the survey was divided into different divisions, each focusing on different aspects of public WiFis. After general and statistically relevant questions like age, gender, education etc. data regarding the primarily used mobile device and tendency towards mobile network or public WiFi usage was collected. Following questions concentrated on the aspect of usage factors and personal perception of the relevance and perception of these factors. Part of the questions were explicit, closed answer multiple choice types. Others, to verify or falsify closed questions, were top of mind questions with a free text response option. Collectively they form a representative image on this matter through validation. The last part concentrated on risks and trust concerning public WiFis and the perception of same. Regarding the trustworthiness of such systems, it was also examined, which factors benefit trustworthiness and how they are perceived regarding their importance.

V. RESULTS

In this section, we present the preliminary results of the project areas S_1 and S_2, whose concepts are explained in Section IV.

A. Results S_1

In 2018, an investigation of WebAPI specifications was conducted. In this study, the specifications regarding the security requirements were examined. Over 900 WebAPI specifications were examined for security requirements. Only 601 could be automatically parsed and evaluated. The selection of specifications is based on market share in order to be representative. So about 50% are from *Microsoft Azure*, *Amazon AWS* and *Google Cloud*. The survey provides insight into the status of concrete indicators for confidentiality: transport encryption, authentication and authorization. At this point, a repetition of the survey is useful to survey the current state.

As shown in Figure 9, transport encryption is defined at over 90% and over 50% of the specification requires authentication. Over 64% of these expect an OAuth 2.0 token for authentication and authorization [51]. In 2021, the WebAPIs specifications were checked for these same criteria using the new analysis system. Updated specifications were used if they were available. There were 671 specifications valid and could be parsed. The goal is to determine the changes in terms of security and to evaluate the functionality of the prototype. The charts in Figure 9 show the results of the 2018 survey compared to those from 2021. Transport encryption will be supported by almost all WebAPIs in 2021. In the current analysis, 95% of all examined specify HTTPS and only 7% specify the unencrypted HTTP protocol. While in 2018, 17% still specified HTTP. It also requires little effort due to the wide availability of free certificates from *LetsEncrypt*. The benefit in terms of confidentiality outweighs this.

Compared to 2018, the share of WebAPIs that require authentication through their specification has grown from 75% to 82%. WebAPIs without authentication are therefore becoming increasingly rare. In terms of misuse and stability of the APIs, this development is certainly to be welcomed. From a data privacy point of view, this can be assessed differently OAuth 2 is the most frequently offered method for authentication and authorization. Compared to 2018, the share decreased slightly from 64% to 61%. Basic Auth is now only specified for 4% of WebAPIs. Compared with 2018, however, the proportion has fallen slightly from 5% to 4%. The basic authentication of http was also less specified. The percentage of APIs using an API Key has increased from 32% to 39%. The API key is a secret string and often serves as both a unique identifier and a secret token for authentication and authorization.

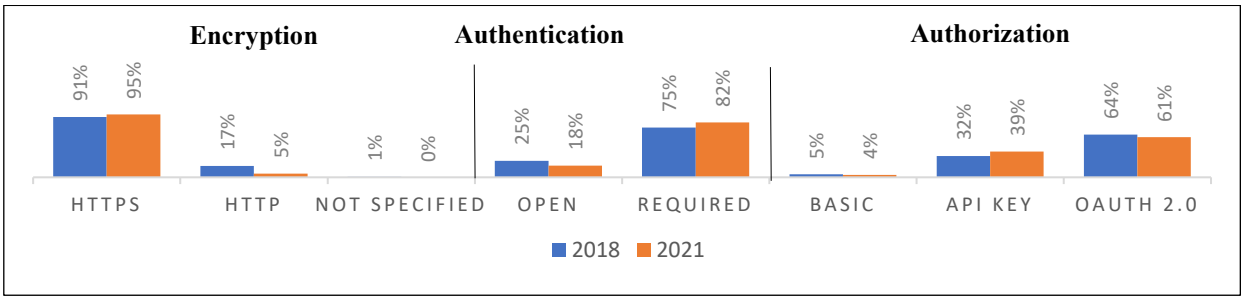


Figure 9. Comparison of the security requirement for WebAPIs from 2018 and 2021 [51].

B. *Results S_2*

The results have previously been published in [52] and [53] with a deeper statistical analysis of the results. Regarding the specific aim of this paper, the relevant results concern the relevance and perception of trustworthiness attributes and therefore form system requirements for this area. First, it is shown, that the trustworthiness of those systems – like estimated – is rather low at 52.10%. With the security of those systems subjectively evaluated at 40.30% a significant correlation of .75 shows the strong interconnection of security and trustworthiness attributes of a digital service as such, as shown in Figure 10.

		Public WiFi Trustworthiness
Public WiFi Security	Pearson Correlation	.75
	Sig. (1-tailed)	.000
	N	1000

Usage

Trustworthiness

		Deutsche Bahn	Deutsche Telekom	McDonald's
Deutsche Bahn	Pearson Correlation	.40	.22	.31
	Sig. (1-tailed)	.000	.000	.000
	N	1000	1000	1000
Deutsche Telekom	Pearson Correlation	.20	.38	.22
	Sig. (1-tailed)	.000	.000	.000
	N	1000	1000	1000
McDonald's	Pearson Correlation	.23	.23	.56
	Sig. (1-tailed)	.000	.000	.000
	N	1000	1000	1000

Figure 10. Evaluations of correlations betweens usage, perceived trustworthiness and security of a system from [53].

This implied the question about the correlation of trustworthiness and usage, whereas the mean of selected service providers formed a correlation of .44 and therefore can be considered as relevant. The underlying implication, that the service provider is a significant factor in this system can be confirmed by the results regarding trust building attributes of a public WiFi, where the service provider was named as the most relevant attribute with 24.35% of all answers, followed by the previously discussed aspects of security at 20.87%, as seen in Figure 11.

Regarding the personal preference of relevant trustworthiness attributes, encryption aspects (66.20%) and a renowned service provider (51.50%) confirm this image as they mark the top two aspects.

As the latter suggests, not only functional requirements were found applicable, as a third-party certification as well as communicative aspects such as the detailed clarification of data usage by the provider where in the lead compared to classical conceptions among public WiFi services such as the acceptance of terms of use etc. Therefore, the presentation of these explicit but nonfunctional requirements has to be taken into consideration, which poses another motivation for the general aim of the presented work.

Figure 11. Most relevant trustworthiness attributes of S2 from [53].

VI. Conclusion and Future Work

The initial results of subareas S1 and S2 clearly show that security and quality are important characteristics for trustworthy services. The comprehensible, transparent communication of measures contributes significantly to the acceptance of the services due to a higher trustworthiness. This correlation could also be shown statistically.

In the further progression of the project, we would like to deepen the sub-areas S1 and S2, as well as work on S3 and S4. The S3 subdomain addresses the trustworthiness of AI web services. For this purpose, the trustworthy properties are to be determined with the help of prototypical tests. Part S4 examines the trustworthiness of the mediator profession. With the help of automated web scrapers, findings on this are to be found and linked. A preliminary study conducted in 2019 serves as the starting point for the investigation [54]. The goal is to determine the sociological role in the context of trustworthy web services.

This will allow us to combine the results from the subareas and obtain a multidimensional picture of the trustworthiness of digital services, as described in section IV. The goal is to support the viewpoints with empirical data in order to be able to set up the requirements for trustworthy services in concrete measures.

Continuing this thought process, a framework benefiting from using such a model could be helpful. An assessment on how necessary a generic requirements engineering framework for trustworthy digital services would be can be found in [39], as well as an proposed approach. Basically, a related framework would provide processes, methods and tools as well as provided forms of documentations for requirements engineering. The generic aspect towards including trustworthiness aspects includes a holistic variety of requirements sources for the requirements elicitation, as well as a model to map trustworthiness requirements across functional and non-functional groups, resources, processes and the product in form of a result at the customers end of a digital service. As part of the EUMovE Project and upcoming activities this approach will be further developed and discussed in the future.

References

[1] S. Hartenstein, S. Schmidt, and A. Schmietendorf, "Towards an Empirical Analysis of Trustworthiness Attributes in the Context of Digitalization," in *The Fourteenth International Conference on Digital Society*, Valencia, Spain, 2020, pp. 112–116. Accessed: Nov. 25 2021. [Online]. Available: https://www.thinkmind.org/articles/icds_2020_3_130_10047.pdf

[2] S. Utz, P. Kerkhof, and J. van den Bos, "Consumers rule: How consumer reviews influence perceived trustworthiness of online stores," *Electronic Commerce Research and Applications*, vol. 11, no. 1, pp. 49–58, 2012, doi: 10.1016/j.elerap.2011.07.010.

[3] European Commission, *Trustworthy AI.* [Online]. Available: https://ec.europa.eu/digital-single-market/en/news/trustworthy-ai-brochure (accessed: Oct. 16 2020).

[4] World Economic Forum, *Our Shared Digital Future Building an Inclusive, Trustworthy and Sustainable Digital Society: Insight Report.* [Online]. Available: http://www3.weforum.org/docs/WEF_Our_Shared_Digital_Future_Report_2018.pdf (accessed: Oct. 16 2020).

[5] C. A. Hill and E. A. O'Hara O'Connor, "A Cognitive Theory of Trust," *SSRN Journal*, 2005, doi: 10.2139/ssrn.869423.

[6] B. M. Muir, *Operators trust in and percentage of time spent using the automatic controllers in a supervisory process control task*. Toronto: University of Toronto, 1989.

[7] J. Lee and N. Moray, "Trust, control strategies and allocation of function in human-machine systems," *Ergonomics*, vol. 35, no. 10, pp. 1243–1270, 1992, doi: 10.1080/00140139208967392.

[8] M. Söllner and J. M. Leimeister, "What We Really Know About Antecedents of Trust: A Critical Review of the Empirical Information Systems Literature on Trust," in *Psychology of Emotions, Motivations and Actions, Psychology of trust: New research*, D. Gefen, Ed., Hauppauge, New York: Nova Science Publishers, 2013, pp. 127–155. [Online]. Available: https://www.researchgate.net/publication/262534043_What_We_Really_Know_About_Antecedents_of_Trust_A_Critical_Review_of_the_Empirical_Information_Systems_Literature_on_Trust

[9] C. L. Corritore, B. Kracher, and S. Wiedenbeck, "On-line trust: concepts, evolving themes, a model," *International Journal of Human-Computer Studies*, vol. 58, no. 6, pp. 737–758, 2003, doi: 10.1016/S1071-5819(03)00041-7.

[10] R. C. Mayer, J. H. Davis, and F. D. Schoorman, "An Integrative Model of Organizational Trust," *The Academy of Management Review*, vol. 20, no. 3, p. 709, 1995, doi: 10.2307/258792.

[11] Gefen, Karahanna, and Straub, "Trust and TAM in Online Shopping: An Integrated Model," *MIS Quarterly*, vol. 27, no. 1, p. 51, 2003, doi: 10.2307/30036519.

[12] M. Kohring, *Vertrauen in Medien - Vertrauen in Technologie*. Stuttgart: Akademie für Technikfolgenabschätzung in Baden-Württemberg, 2001. [Online]. Available: http://elib.uni-stuttgart.de/handle/11682/8694

[13] R. Kuhlen, "Vertrauen in elektronischen Räumen," in *Informationelles Vertrauen für die Informationsgesellschaft*: Springer, Berlin, Heidelberg, 2008, pp. 37–51. Accessed: Nov. 11 2021. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-540-77670-3_3

[14] S. van der Graaf, W. Vanobberghen, M. Kanakakis, and C. Kalogiros, "Usable Trust: Grasping Trust Dynamics for Online Security as a Service," vol. 9190, pp. 271–283, 2015, doi: 10.1007/978-3-319-20376-8_25.

[15] A. Chakravarthy *et al.,* "OPTET D2.4 Socio-economic evaluation of trust and trustworthiness," OPTET. Accessed: Oct. 16 2020. [Online]. Available: https://www.researchgate.net/publication/317488309_OPTET_D24_Socio-economic_evaluation_of_trust_and_trustworthiness

[16] S. Wiegand *et al.,* "D2.5 – Consolidated report on the socio-economic basis for trust and trustworthiness," OPTET, 2015. Accessed: Oct. 16 2020. [Online]. Available: https://www.researchgate.net/publication/317488377_OPTET_D25_-_Consolidated_report_on_the_socio-economic_basis_for_trust_and_trustworthiness

[17] D. H. McKnight and N. L. Chervany, "Trust and Distrust Definitions: One Bite at a Time," in *Lecture Notes in Computer Science*, vol. 2246, *Trust in Cyber-societies: Integrating the*

Human and Artificial Perspectives, R. Falcone, M. Singh, and Y.-H. Tan, Eds., Berlin, Heidelberg: Springer, 2001, pp. 27–54.

[18] B. G. Robbins, "What is Trust? A Multidisciplinary Review, Critique, and Synthesis," *Sociology Compass*, vol. 10, no. 10, pp. 972–986, 2016, doi: 10.1111/soc4.12391.

[19] N. Gol Mohammadi *et al.,* "An Analysis of Software Quality Attributes and Their Contribution to Trustworthiness," *Proceedings of the 3rd International Conference on Cloud Computing and Services, Science*, pp. 542–552, 2013.

[20] G. Johnson, K. Scholes, and R. Whittington, *Strategic Management [orig.: Strategisches Management]: An Introduction; Analysis, Decision and Implementation [orig.: Eine Einführung; Analyse, Entscheidung und Umsetzung],* 9th ed. München: Pearson Studium, 2011.

[21] S. Hartenstein, S. Schmidt, and A. Schmietendorf, "Trust Patterns in Modern Web-API Based Service Architectures - More than Technical Security Aspects," in *Patterns 2021*: IARIA, 2021, pp. 23–25. Accessed: May 5 2021. [Online]. Available: http://thinkmind.org/articles/patterns_2021_2_10_70007.pdf

[22] A. Patrick, S. Marsh, and P. Briggs, "Designing Systems That People Will Trust," *Security and Usability*, NRC 47438, pp. 75–99, 2005. [Online]. Available: https://www.researchgate.net/profile/Pamela_Briggs/publication/44081283_Designing_Systems_That_People_Will_Trust/links/00b7d5344d8b27f675000000.pdf

[23] G. Simmel, *The sociology of Georg Simmel: Selected writings*. New York: Free Pr, 1964.

[24] D. Trček, "A Brief Overview of Trust and Reputation over Various Domains," in *SpringerBriefs in Information Systems, Trust and Reputation Management Systems: An e-Business Perspective*, D. Trček, Ed., Cham: Springer International Publishing, 2018, pp. 5–19.

[25] T. Nguyen, "Trust and Sincerity in Art," *Ergo*, 2020. [Online]. Available: https://www.researchgate.net/publication/343239976_Trust_and_Sincerity_in_Art

[26] S. G. Grimmelikhuijsen and A. J. Meijer, "Effects of Transparency on the Perceived Trustworthiness of a Government Organization: Evidence from an Online Experiment," *JOPART*, vol. 24, no. 1, pp. 137–157, 2014, doi: 10.1093/jopart/mus048.

[27] S. Paulus, N. G. Mohammadi, and T. Weyer, "Trustworthy Software Development," in *Lecture Notes in Computer Science, Communications and Multimedia Security*, D. Hutchison et al., Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 233–247.

[28] N. G. Mohammadi *et al.,* "Trustworthiness Attributes and Metrics for Engineering Trusted Internet-Based Software Systems," in *Communications in Computer and Information Science, Cloud Computing and Services Science*, M. Helfert, F. Desprez, D. Ferguson, and F. Leymann, Eds., Cham: Springer International Publishing, 2014, pp. 19–35.

[29] A. Hoffmann, H. Hoffmann, and M. Söllner, "Twenty Software Requirement Patterns to Specify Recommender Systems that Users Will Trust," *ECIS 2012 Proceedings.Paper 1*, 2012. [Online]. Available: https://www.alexandria.unisg.ch/228939/1/Hoffmann%20et%20al.%202012.pdf

[30] S. Hartenstein, K. Nadobny, S. Schmidt, and A. Schmietendorf, *Sicherheits- und Compliance-Management im Lebenszyklus von Web APIs: Ergebnisse eines Forschungsprojektes an der HWR Berlin/Uni Magdeburg*. Berlin: Logos-Verlag, 2020.

[31] OpenAPI Initiative, *OpenAPI Specification.* [Online]. Available: https://spec.openapis.org/oas/v3.1.0 (accessed: May 3 2021).

[32] SmartBear Software, *Swagger.* [Online]. Available: https://swagger.io/ (accessed: May 21 2021).

[33] ApiTree, *APITree Hub.* [Online]. Available: https://www.apitree.com/ (accessed: May 21 2021).

[34] iab Austria, *ondevice research.* [Online]. Available: http://www.iab-austria.at/wp-content/uploads/2015/07/IAB-Mobile-Video-Usage-FINAL.pdf

[35] Eurostat, *Eurostat - Data Explorer,.* [Online]. Available: http://appsso.eurostat.ec.europa.eu/nui/submitViewTableAction.do

[36] A. Grieß, *Nur Minderheit nutzt WLAN außerhalb der eigenen vier Wände.* [Online]. Available: https://de.statista.com/infografik/3575/nutzung-von-fremden-wlan-netzen/ (accessed: Mar. 6 2021).

[37] EarsandEyes GmbH, *Öffentliches WLAN in Deutschland.* [Online]. Available: https://www.earsandeyes.com/download/wlan-report

[38] Statista, *Beliebteste Nutzungsorte von WLAN 2018 | Statista.* [Online]. Available: https://de.statista.com/prognosen/953712/umfrage-in-deutschland-zu-den-beliebtesten-nutzungsorten-von-wlan (accessed: Feb. 20 2021).

[39] S. Schmidt, "Zur Notwendigkeit eines generischen Requirements Engineering Frameworks für vertrauenswürdige IT-Services," in *Berliner Schriften zu modernen Integrationsarchitekturen*, vol. 25, *Online-Workshop (e) trust – Vertrauen in Digitale Dienste (Werte – Risiken – Prinzipien – Methoden – Techniken)*, A. Schmietendorf, Ed., 1st ed., Düren: Shaker Verlag, 2021.

[40] H.-J. Bullinger and A.-W. Scheer, Eds., *Service Engineering: Entwicklung und Gestaltung innovativer Dienstleistungen ; mit 24 Tabellen,* 2nd ed. Berlin: Springer, 2006.

[41] M. Glinz, *A Glossary of Requirements Engineering Terminology.* Accessed: Jul. 21 2021. [Online]. Available: https://www.merlin.uzh.ch/contributionDocument/download/9869

[42] C. Bowman, *Strategy in practice*. Harlow: Prentice Hall Financial Times, 1998.

[43] H. Meffert, C. Burmann, and M. Kirchgeorg, *Marketing: Grundlagen marktorientierter Unternehmensführung ; Konzepte, Instrumente, Praxisbeispiele,* 10th ed. Wiesbaden: Gabler, 2008.

[44] S. Hartenstein, K. Nadobny, S. Schmidt, and A. Schmietendorf, "An Approach for a Fast Cost Validation of Web-Based APIs supported by Functional Size Measurement with COSMIC," in vol. 2476, *IWSM-Mensura 2019: International Workshop on Software Measurement and International Conference on Software Process and Product Measurement 2019*, Ayca Kolukisa Tarhan, Ahmet Coskuncay, Ed., Haarlem, The Netherlands: CEUR Workshop Proceedings, 2019, pp. 103–111. Accessed: Oct. 21 2019. [Online]. Available: http://ceur-ws.org/Vol-2476/short2.pdf

[45] *swagger-parser.* [Online]. Available: https://github.com/swagger-api/swagger-parser (accessed: May 12 2021).

[46] *openapi4j.* [Online]. Available: https://github.com/openapi4j/openapi4j (accessed: May 12 2021).

[47] *KaiZen-OpenApi-Parser.* [Online]. Available: https://github.com/RepreZen/KaiZen-OpenApi-Parser (accessed: May 12 2021).

[48] *openapi-format.* [Online]. Available: https://github.com/thim81/openapi-format (accessed: May 12 2021).
[49] *openapi-snippet.* [Online]. Available: https://github.com/ErikWittern/openapi-snippet (accessed: May 12 2021).
[50] EarsandEyes GmbH, *Report Öffentliches WLAN in Deutschland (Public WLAN in Germany Report).* [Online]. Available: https://www.earsandeyes.com/wp-content/uploads/2019/05/EARSandEYES_Report_%C3%96ffentliches_WLAN.pdf (accessed: Feb. 2 2020).
[51] A. Reichenbach and A. Schmietendorf, "Empirische Untersuchung zur Open API Spezifikationen," in *Berliner Schriften zu modernen Integrationsarchitekturen*, Band 18, *API-First/API-Management - Open APIs als Treiber der Digitalisierung: Workshop im Rahmen der Enterprise Computing Conference, 19. April 2018, Hamburg*, A. Schmietendorf and A. Nitze, Eds., 1st ed., Aachen: Shaker, 2018, pp. 1–28.
[52] S. Schmidt, "Schaffung eines vertrauenswürdigen, öffentlichen WLANs - Herangehensweise und Teilergebnisse," in *Berliner Schriften zu modernen Integrationsarchitekturen*, vol. 24, *ESAPI 2020: 4. Workshop Evaluation of Service-APIs*, A. Schmietendorf and K. Nadobny, Eds., 1st ed., Düren: Shaker, 2020, pp. 35–48. Accessed: Jul. 17 2021. [Online]. Available: https://www.researchgate.net/publication/345081598_Schaffung_eines_vertrauenswurdigen_offentlichen_WLANs_-_Herangehensweise_und_Teilergebnisse
[53] S. Schmidt, "On the perception and relevance of trustworthiness in public wireless networks," in *Advances in Security, Networks, and Internet of Things: Proceedings from SAM'21, ICWN'21, ICOMP'21, and ESCS'21*, 2021.
[54] W. H. Letzel and A. Schmietendorf, "Digitalisierung und Mediation aus der Anwenderperspektive," no. 1, 4-10, 2019.

Analyzing the Attack Surface and Threats of Industrial Internet of Things Devices

Simon Liebl*†, Leah Lathrop*, Ulrich Raithel‡, Andreas Aßmuth*, Ian Ferguson†, and Matthias Söllner*

*Technical University of Applied Sciences OTH Amberg-Weiden, Amberg, Germany
E-mail: {s.liebl | l.lathrop | a.assmuth | m.soellner}@oth-aw.de
†Abertay University, Dundee, UK
E-mail: i.ferguson@abertay.ac.uk
‡SIPOS Aktorik GmbH, Altdorf, Germany
E-mail: ulrich.raithel@sipos.de

***Abstract*—The growing connectivity of industrial devices as a result of the Internet of Things is increasing the risks to Industrial Control Systems. Since attacks on such devices can also cause damage to people and machines, they must be properly secured. Therefore, a threat analysis is required in order to identify weaknesses and thus mitigate the risk. In this paper, we present a systematic and holistic procedure for analyzing the attack surface and threats of Industrial Internet of Things devices. Our approach is to consider all components including hardware, software and data, assets, threats and attacks throughout the entire product life cycle.**

***Keywords*—Threat analysis; attack surface; Industrial Internet of Things; cyber-physical systems; cloud.**

I. Introduction

The Internet of Things (IoT) is increasingly making its way into our daily lives. Smart devices are omnipresent, in smart homes, medical and infrastructure applications, and building automation. Industrial applications, for example, in manufacturing, the automotive and oil and gas industry, are other major fields of application, summarized as the Industrial Internet of Things (IIoT). The hype around the IoT and IIoT led, for example, to dozens of different platforms and, consequently, to compatibility problems. The race for the shortest time to market also led to security and privacy issues, as these topics have been neglected or even omitted entirely so far. To address the latter issues, our work in [1] and this extension aim to support IIoT device manufacturers and operators in identifying threats against their devices.

According to [2], about every third Industrial Control System (ICS) computer was attacked within the second half of 2020. The situation was also exacerbated by the COVID-19 pandemic, as increasing Remote Desktop Protocol (RDP) connections also led to a rise in brute force attacks on them. A rise in network-capable Operational Technology (OT) components has been observable for years anyway [3]. The main threat arises from ransomware, i.e., malware that encrypts files and demands ransom, and coinminers, i.e., malware used to mine cryptocurrencies [4]. The attack on Colonial Pipeline [5] showed once again that larger parts of the population can also be affected by such attacks. In addition, the attack on a U.S. water treatment facility in early 2021 [6] highlighted that ICSs in critical infrastructures are particularly at risk of targeted attacks.

As a consequence of the increasing threats, IIoT manufacturers must secure their devices to prevent such incidents. However, implementing best practices around default passwords is not enough for properly secured devices. Manufacturers, and also operators, must therefore be fully aware of threats and the resulting risks. The purpose of this paper is to support them in their threat analysis. Our goal is a holistic view of the attack surface of IIoT devices. To achieve this, all components including hardware, software, and data, assets, as well as threats and attacks should be considered throughout the entire life cycle of the device. Our research methodology can be described as follows: the first two steps are analyses of the components and assets of an IIoT device, followed by a threat and an attack categorization. By considering all assets in the threat categorization, a complete list should be enabled. Similarly, the component analysis should provide a complete attack and weakness categorization.

The remainder of this paper is structured as follows: in Section II, the different terms around the IIoT are clarified. Section III presents related work from other researchers as well as organizations. In Section IV, the components of an IIoT device are decomposed into hardware, software, and data, followed by an asset analysis in Section V. In Section VI, a threat categorization is presented that consists of nearly 50 categories organized into 10 groups. A similar categorization of attacks and weaknesses throughout the life cycle of an IIoT device is introduced in Section VII. Section VIII recommends six steps for a systematic threat analysis of IIoT devices. The paper ends with conclusions in Section IX.

II. The Industrial Internet of Things

This section briefly explains the terms IoT, IIoT, Cyber-Physical System (CPS), ICS, Information Technology (IT), and OT and how they are related (see Figure 1).

The IoT is a network of connected devices, which are sensors and/or actuators fulfilling a specific function. The infrastructure enables communication with other equipment along with the storage, processing, and distribution of data to

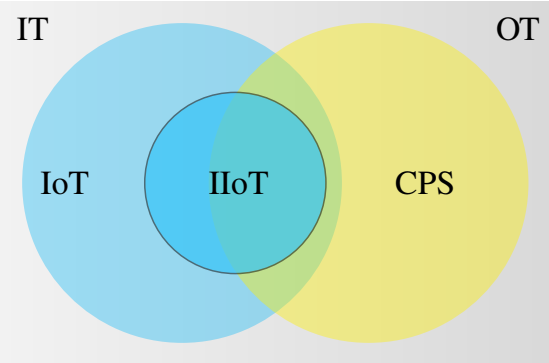


Figure 1. Relationship of the various terms, adopted from [8].

other systems and users [7]. Cloud services realize universal accessible utility services, e.g., for device management, and centralized data processing for analytics, which may be supported by Artificial Intelligence (AI). The gained knowledge can be made available to users, other systems and the devices themselves. Use cases span many domains, such as consumer applications (e.g., smart home), commercial (e.g., medical and healthcare, transportation), and infrastructure applications (e.g., smart grid).

The IIoT is a part of the IoT, but differentiates in some aspects. The IIoT integrates the previously separated areas of IT and OT by connecting OT components, such as machines and control systems, with IT systems and business processes [8]. The integration is accomplished, for instance, through edge devices and gateways that enable processing in the cloud. The leading use cases of the IIoT are operational intelligence, asset monitoring, and predictive maintenance [9]. The goals are, among others, to increase productivity, improve safety, gain flexibility and agility, and reduce energy consumption. It should be noted that use cases, services, and communication in the IIoT are machine-oriented, while these are human-centered in many IoT applications, such as in consumer IoT [8].

The OT components listed above are usually installed within an ICS, which is structured into several layers. In the Purdue reference model, level 0 describes the physical process, which is sensed and manipulated by sensors and actuators and controlled by, for example, Programmable Logic Controllers (PLCs) in level 1. Level 2 includes Human-Machine Interfaces (HMIs) and the Supervisory Control And Data Acquisition (SCADA) system to provide operators with aggregated information. Layers above contain backup servers and Enterprise Resource Planning (ERP) systems, among others.

The last term that needs to be clarified is CPS. These can be found in the IoT/IIoT and in an ICS. Their primary task is the control of a physical process in the real world using sensors and actuators. Furthermore, they are equipped with network capability. Another characteristic is that they require real-time interaction with the physical world [10].

Different objectives and requirements emerge from the characteristics of CPSs and ICSs. Besides integrity and availability, the security goals for IoT devices are centered on confidentiality and privacy, as personal data, such as health data, is processed. IIoT devices, in contrast, focus additionally on safety and the impact on the environment and society [11]. In industrial plants, humans work with heavy machinery in a confined workspace. An accident can potentially cause injury, death, damaged production equipment or environmental disasters. The impact of an IIoT failure may be worse in critical infrastructures, such as energy and water supply, food, and health, as large parts of the population may be affected.

To sum up, the IIoT allows the processing of data, e.g., produced by CPSs in an ICS, in the cloud. This is realized by connected edge devices, gateways, and OT components. The control of physical processes leads to further requirements such as safety. The increased connectivity and the high requirements result in additional threats and increased risk to IIoT devices, which is further discussed in Section VII.

III. Related Work

The need for action in the area of IoT security was recognized by experts a long time ago [12]. In recent years, this has also been identified by government institutions and industry. Nevertheless, manufacturers of embedded systems still struggle to integrate security features or even to work according to the principle of security by design.

Varga et al. [13] discuss IoT threats in the automation domain. Based on an IoT architecture consisting of the four layers sensors and actuators, networking, data processing, and application, the authors present different threats and the required countermeasures. In [14], Atamli et al. describe a threat-based security analysis that focuses on the three use cases power management, smart car, and smart healthcare system. Initially, they discuss sources of threats and classify attack vectors. Afterwards, the security and privacy impact of attacks in the area of the listed use cases is described. Last, desirable security and privacy properties are defined. Abomhara et al. [15] provide background information on IoT devices and services, threats, attacks, and security and privacy goals. Subsequently, the motivation of attacks and a classification of possible intruders are presented. In [16], Wurm et al. conducted security analyses on a consumer IoT and an IIoT device and demonstrated how these devices could be exploited.

Dozens of organizations and government institutions, such as the Industrial Internet Consortium (IIC), the Cloud Security Alliance (CSA), and the US National Institute of Standards and Technology (NIST), have published guidelines and best practices for IoT security. Particularly noteworthy is [17], which brought together about 100 documents from 50 different organizations, resulting in 13 points for recommended action. Furthermore, the contributions of the European Union Agency for Cybersecurity (ENISA) and the German Federal Office for Information Security (BSI) are recommended. The former have published several analyses and recommendations for the various areas of the IoT [18]. For example, the baseline security recommendations for IoT [19] provides a threat taxonomy, attack scenarios and a list of security measures. The

BSI annually publishes an Information Security Management System (ISMS), the so-called IT-Grundschutz Compendium, which also considers ICS components, embedded systems, and IoT devices, among others. In addition to organizational aspects, technical issues are also addressed, including a list of threats and necessary security requirements.

There are numerous papers and guidelines describing the threats to IoT devices. However, the threats are often described only in general terms and in a jumbled manner. For example, the aforementioned threat taxonomy by ENISA lists 25 threats, but attack techniques (e.g., replay of messages), weaknesses (e.g., software vulnerabilities), and threat consequences (e.g., sensitive information leaking) are considered without further distinction. Systematic and holistic approaches are needed to enable the identification of all attack vectors of IIoT devices by their manufacturers. It is this shortcoming that we wish to address in this work.

IV. Components of an IIoT Device

To enable the full analysis of attack surfaces, a breakdown of the components of a typical device is presented in the following section. The components can be grouped into the three categories hardware, software, and data, which are described in the following subsections in detail. Figure 2 presents an overview of the components an IIoT device may contain.

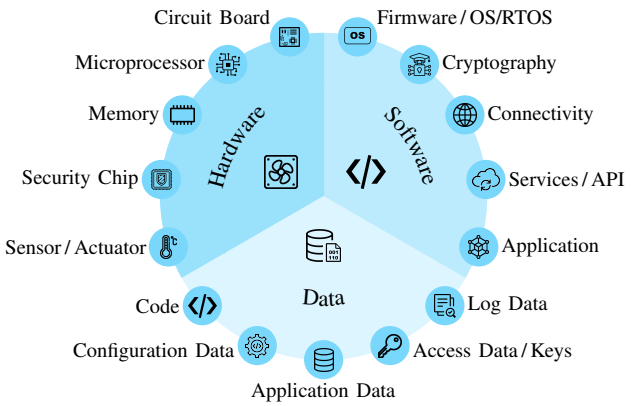


Figure 2. Hardware and software components and types of data that IIoT devices may contain.

A. Hardware

The first component that comes to mind is the enclosure. It must be suitable for the environment and may have to be explosion-, water-, and dust-proof. Appearance, size, and usability are particularly important in the consumer sector, but the industrial field also appreciates these properties. Plant operators, for example, prefer simple and space-saving installation in the switching cabinet and quick familiarization with handling by employees. In critical applications, tamper protection is used to detect modifications to the device.

A central component of the interior are Printed Circuit Boards (PCBs). They are often the bridge between the various mechanical and electrical parts of a device and also connect the countless electrical components on a PCB such as controllers, Integrated Circuits (ICs), oscillators, fuses, and basic electrical elements.

The core components on a PCB are processors. Several types are available, each with its own benefits and drawbacks. Among others, there are microprocessors, microcontrollers, Application-Specific Integrated Circuits (ASICs), and Field-Programmable Gate Arrays (FPGAs). Microprocessors ship in a single IC, which vice versa may contain multiple microprocessors in case of a multi-core design. They can be made for general-purpose or specialized on a specific task, e.g., signal, graphics, and physics processing. In addition to the microprocessor, dozens of peripherals are integrated into microcontroller ICs, such as memory, analog and digital inputs and outputs, serial communication interfaces, a Real-Time Clock (RTC) and in-circuit debug support. Increasingly, security features such as a Trusted Execution Environment (TEE), a True Random Number Generator (TRNG), a cryptography accelerator, and a Physical Unclonable Function (PUF) are also being embedded. ASICs are customized for a certain task and their advantages include, for instance, greater performance and optimized size. Unlike ASICs, FPGAs can be updated after production and are more cost-effective, especially for smaller quantities. IoT devices usually employ microcontrollers as their main Central Processing Unit (CPU) because they are feature-rich and are still low-priced and compact. The average CPU has a single core and the clock speed is in the double-digit MHz range, which drastically limits the performance compared to desktop CPUs in IT systems.

Memory can be integrated into the microcontroller, placed as separate IC on the PCB or connected by slots in the enclosure. The main memory is typically Static Random Access Memory (SRAM) or Dynamic Random Access Memory (DRAM). There are different technologies for data storage, for example, FLASH, Electrically Erasable Programmable Read-Only Memory (EEPROM), and One-Time Programmable (OTP) memory. Some security-focused microcontrollers include a on-the-fly encryption engine that enables secure storage on external ICs. Removable storage technologies, such as SD cards or USB sticks, are used, for example, to export user data or import user applications. The total main memory is usually in the kilobyte range, the data storage reaches the megabyte range.

Like memory, a security chip can also be integrated into a microcontroller or placed on the PCB. Security chip is used as umbrella term for secure cryptoprocessors, which can be a Trusted Platform Module (TPM) or a Secure Element (SE). An exception are Hardware Security Modules (HSMs) that are integrated via an external module. Their capabilities differ in certain functions, although the common basic idea is to outsource all cryptographic operations to a tamper-proof co-processor.

Another substantial part of IIoT devices are input and output components including sensors and actuators. The power supply is also a type of input component, as it supplies the device with power via a cable connection, battery, or solar

panel, among others. User input interfaces may include simple switches or utilize more advanced human input devices such as keyboards or touchpads. According to Sikder et al. [20], the various sensor can be categorized into environmental sensors (e.g., audio, image, temperature and humidity sensor), position sensors (e.g., inductive, ultrasonic, proximity, and magnetic sensor) and motion sensors (e.g., flow sensor, gyroscope, accelerometer); we extended this list with industrial sensors, summarized as process sensors (e.g., current, pressure, and chemical sensor). The output components can be grouped in visual outputs (e.g., LEDs, display), audio outputs (e.g., loudspeaker), power outputs (e.g., relay, power electronics), and actuators (e.g., electric and pneumatic actuator).

The last group of components in this list are connectivity elements. This includes conductive paths on the PCB and wires within the device. The numerous communication interfaces of a system may require controllers, connectors, or antennas.

B. Software

The firmware is the linking component between hardware and software of an embedded system, as it provides software with low-level access to the hardware. Simple embedded devices often have no underlying Operating System (OS); therefore, they run only the firmware, which is known as bare metal program. The most utilized OS in IoT devices is Linux [21], followed by FreeRTOS, an open source Real-Time OS (RTOS). In addition to real-time capability, some CPSs in safety-critical applications may require the fulfillment of further standards such as IEC 61508. Specialized OSs have been developed to comply with these requirements, for example, SAFERTOS achieved the highest Safety Integrity Level (SIL) of IEC 61508 for a software-only component, i.e., SIL 3.

Although hardware-supported cryptography in the form of cryptographic accelerators or security chips is more efficient than software libraries, most IoT devices use mainly software-based cryptography. The main reasons for this are that crypto-hardware is hardly available and software libraries often do not exploit available hardware for portability reasons [22]. There is a wide range of fee-based as well as cost-free crypto libraries available. However, there are some challenges that complicate their use. First, only a few of them can easily be adapted to systems without an OS, which is about every fourth IoT device [21]. Second, the required storage space exceeds the frequently limited memory. Finally, the execution of cryptographic algorithms on low-power devices is highly time-consuming and, therefore, compromises other requirements such as real-time capability and usability. These challenges are addressed in the research field lightweight cryptography in order to provide efficient and storage-saving cryptographic software libraries on all devices.

Connectivity is one of the major topics addressed during the design of an IoT device as it defines how a device interacts with other systems as well as users. A wide range of communication protocol stacks are used for the various applications. There are stacks for local communication (e.g., USB), Internet communication (e.g., TCP/IP), and automation processes (e.g., Modbus). Currently, there is a trend towards Ethernet-based automation protocols to take advantage of synergy. IIoT devices usually implement several protocol stacks for compatibility reasons. This results in devices supporting legacy protocols such as PROFIBUS and HART as well as more recent ones such as PROFINET, OPC UA, MQTT, and LoRa.

Smart services unlock the value of the IoT. Countless devices can thus connect with each other and create an elaborated decision. At the same time, they can be centrally monitored and controlled. One central service is device management, which includes device registration, organization, inspection, and software and firmware updates. This may be accompanied by several other services, for instance, monitoring, logging, and attestation services. Frequently, a local service is also required for configuration and control that is usually only available in the local network. Therefore, many devices implement an embedded web server or a mobile app Application Programming Interface (API) for this purpose.

Last, every device runs its own specific application. Depending on the use case, this might be providing sensor values, controlling an actuator, or bundling messages from multiple smart sensors. Edge devices may also perform data pre-processing or minor analytics. Additionally, some devices allow users to execute their own programs and code.

C. Data

IIoT devices hold various types of data. First of all, any code including the firmware and application can be considered as data. Code is preferably stored in a FLASH memory or, if it is not available, in an EEPROM.

During device setup, many options need to be configured. Configuration data can include network settings, environment and calibration parameter, sensor and actuator settings, and machine learning parameters. It is stored in an EEPROM and optionally also stored in the cloud as backup.

Application data is specific to the device. This can be collected input data, such as sensor values, and produced output data, such as analysis results.

Devices that implement basic security controls require authentication data and cryptographic keys. Local access to the device via display, browser, or mobile app should only be allowed after entering valid credentials. The user database is often stored locally and contains the names of the respective users with passwords or the derived hash values. Cryptographic keys are, for example, required to securely communicate with any other entity such as a cloud service. Not only secret keys are stored, but also public keys, for instance the manufacturer's public key, to be able to verify firmware signatures. User databases need to be updatable and are therefore stored in an EEPROM, whereas public keys may be required to be tamper-proof and consequently stored, for example, in an OTP memory.

IoT devices log relevant events for system diagnosis. These include application-specific events as well as security-relevant

Figure 3. Assets from the perspective of manufacturers, owners, and users as well as their common objectives.

events such as login attempts, for example. Unfortunately, in practice, some devices also log highly sensitive data such as passwords.

V. Assets of an IIoT Device

This section emphasizes the assets of an IIoT device based on the previously gathered components. We identified three major stakeholders in a device: manufacturers, owners, and users. By putting oneself in the shoes of the particular group, the interests and assets worth protecting can be identified. This, in turn, contributes to the understanding of what attackers might be targeting. Figure 3 summarizes the assets for each stakeholder as well as common ones highlighted in the arrow.

First, the assets of each group starting with manufacturers are considered and then the common ones. A primary asset of manufacturers is their Intellectual Property (IP), which can be further distinguished into the domains hardware, software, and process. Hardware IP includes, for instance, the design of an ASIC or PCB and mechanical components. Manufacturers put a lot of effort in optimizing the hardware design for energy usage, heat flow, and size. Consequently, hardware is one of the assets worth protecting in order to have an advantage over competitors. The same applies to software IP. Developing the firmware, inventing application-specific algorithms, or designing machine learning models is time-consuming. Therefore, manufacturers want to protect their software against copying, theft, or publication. Process IP means the overall concept or a unique solution for a specific problem that might be worth protecting. Two further assets are liability and reputation. Manufacturers have a certain responsibility to ensure that no unexpected incidents, such as physical injury or property damage, occur. Any incidents could also have an impact on their company's image.

Next, the assets are considered from the owner's perspective, the person or organization that purchased the device. First of all, owners want to protect their physical property, which is the newly acquired device and also all other belongings. Second, the owner has virtual property. This includes, for instance, code for custom applications and (configuration) data stored on the device. Additionally, devices produce data during operation. This includes collected sensor values and also meta information such as the availability of a device. Furthermore, owners want to prevent or detect tampering with the device by users.

Users foremost interest when using the device is their health. It must neither be endangered by one-time events nor by long-term use of the device. Users also want to protect their data on the device, which they actively stored on it or which was generated during the use of it. Especially IoT devices with dozens of sensors might collect audio and video, health, and behavioral data that could compromise the privacy of users.

Last, the assets and objectives that all stakeholders have in common. Functionality and safety are important attributes of an IoT device. This includes logical operations, such as successful firmware updates or changing of settings, and physical operations, such as moving an actuator or cutting off electricity if a human is present. If these requirements are not achieved, this can have consequences for the manufacturer's reputation, destroy production facilities and cause downtime, or injure users.

VI. Threat Categorization

In this section, the threats to IIoT devices are analyzed. The ENISA defines a threat as "any circumstance or event with the potential to adversely impact an asset through unauthorized access, destruction, disclosure, modification of data, and/or denial of service" [23]. In order to better understand and assess threats, we categorized and grouped them according to their impact. The collected threat categories were identified based on the preceding asset analysis and are presented in Figure 4. The various threat groups are briefly outlined below.

A. Nefarious Activity / Abuse / Misuse

The first group summarizes rather generic threats from nefarious activity, abuse, and misuse. Abuse of personal data is the use of personal information in a manner for which it was not intended, for example, a company selling ones email address to advertisers. Another threat is tampering with data or also poisoning of data, which can occur in the context of machine learning. Any threat that compromises availability is summarized as denial of service. The impact of these threats is significant in ICSs because they might lead to a production stop and, thus, cause financial loss or even worse in critical infrastructures. Information disclosure and leakage is sharing of data that is intended to be confidential. Misuse of computing or electrical power can be caused by malware or the inappropriate use of the device by employees. Computing power can be misused by botnets that launch Distributed Denial of Service (DDoS) attacks, mine cryptocurrencies, or spread spam. Employees might use the device to charge their phone, affecting the functionality of the device. Privilege abuse is a threat that arises from employees who maliciously use their privileges. About two out of three incident are financially motivated; further reasons are fun, grudge and espionage [24].

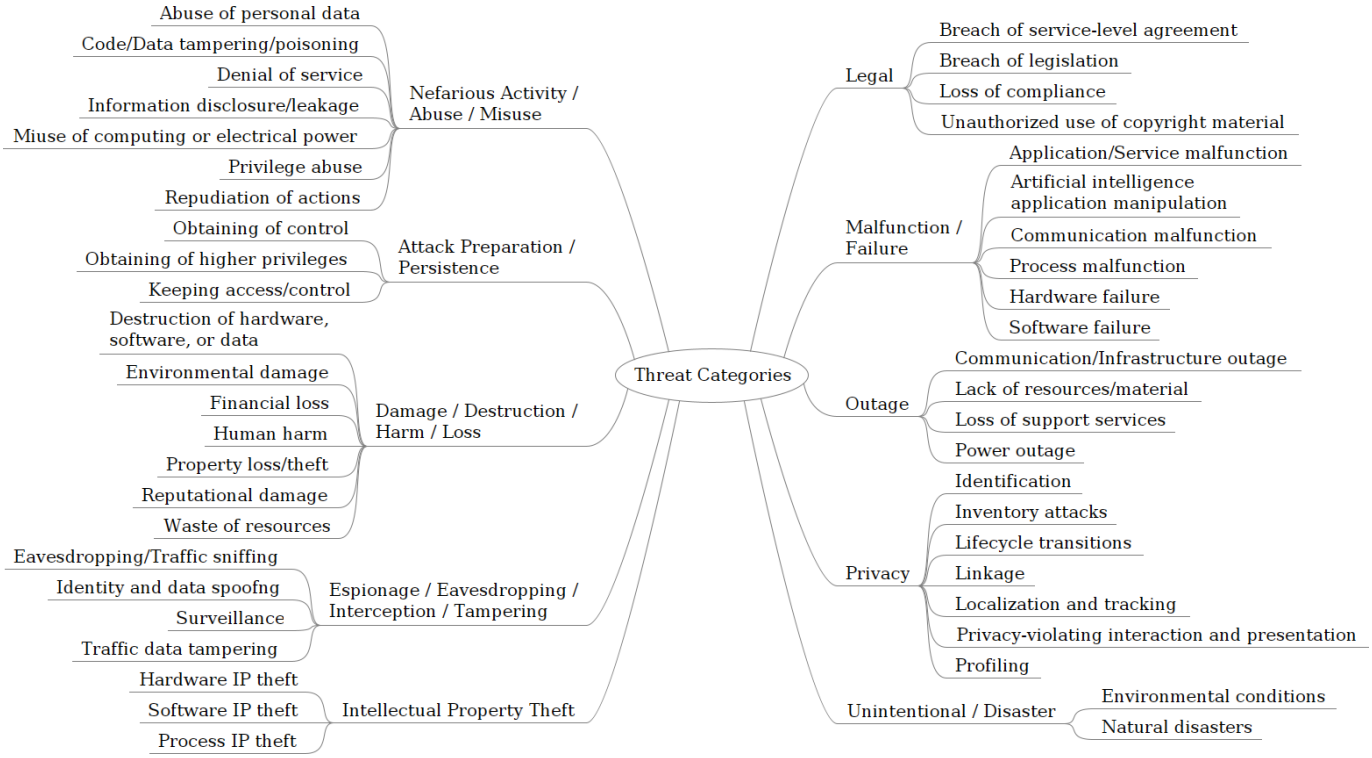


Figure 4. An overview of the grouped threat categories.

The last category of this group is repudiation of actions. In case of an incident, investigators try to reconstruct the exact procedure, for example, by analyzing log files. Attackers could manipulate or delete them in order to remain undetected. However, many IIoT devices do not uniquely identify users so far, which makes it easier for them to deny it.

B. Attack Preparation / Persistence

This group summarizes threats in which attackers gain unauthorized access to the device or a resource to prepare further attacks. A common technique is privilege escalation to obtain either initial access or higher privileges. Afterwards, adversaries might try to keep access across restarts, updates, or factory resets.

C. Damage / Destruction / Harm / Loss

Damage can arise in many ways. Attackers can destroy hardware, software, or data of the device. For example, ransomware can destroy data by encrypting it, software can be erased by deleting it requiring a reinstallation (e.g., the malware Brickerbot did it this way [25]), and hardware can be destroyed by actuator malfunction, a short circuit, or vandalism. Damage can also exceed device borders. Humans can be injured by attacks compromising safety such as the aforementioned example actuator malfunction and by consequential damage from attacks on the power grid. Additionally, environmental damage can result, as demonstrated by an attack with simple radio signals on Maroochy Water Services that discharged 800,000 liters of sewage to local parks and rivers [26]. Furthermore, financial loss can occur to manufacturers through product piracy, to owners through production downtime, and to users through theft of credit card information. Manufacturers and operators also fear damage to their company's image. Many manufacturers do not disclose vulnerabilities, operators publish incidents sketchily, and it is also suspected that only a fraction is made public.

D. Espionage / Eavesdropping / Interception / Tampering

In ICSs, encrypted communication is still rather rare. Industrial espionage by eavesdropping on communications is thus simpler. It also facilitates tampering with data. Payload data such as sensor values and commands can be manipulated, and identities can be spoofed. For instance, attackers could masquerade as the device and send false data to PLCs or cloud services. In addition, the variety of sensors such as microphones or cameras enables surveillance, for example, to monitor the behavior and activities of employees.

E. Intellectual Property Theft

The significance of IP has already been described in Section V. The primary source of threats to hardware, software, and process IP is from competitors.

F. Legal

This group considers legal implications including breach of service-level agreements, breach of legislation, and loss of compliance. An increasing number of laws have recently been drafted for connected devices. For example, a law in California requires to have unique preprogrammed passwords for each

device [27]. Germany also passed a law requiring manufacturers of digital devices to provide updates [28]. Consequently, manufacturers must continuously check whether the legal situation for their devices has changed. Last, manufacturers use a variety of third-party components. They must be careful not to use protected material without authorization.

G. Malfunction / Failure

One of the most famous attacks that caused application manipulation is Stuxnet, in which the speed of centrifuges was changed, while hiding it from monitoring systems [29]. Two recent examples are TRITON [30] and Industroyer [31] that were specifically created for OT devices and protocols. AI methods are currently used in cloud services, edge devices and even smart sensors. Special attention should be paid to them as they can increase the number of threats. Attacks could compromise and limit AI results, reduce their effectiveness, and lead to misclassification by providing adversarial examples. In addition to the application, the communication can also be manipulated. In this case, it is not about single bytes, but rather about entire packets. Gateways, for example, are the link between sensors and cloud services. Occasionally, packets may be redirect or not forwarded at all. Further threats may arise from hardware and software failures; components can fail due to age or quality issues and software may contain bugs.

H. Outage

This group summarizes outages that affect large parts of an ICS. The power supply is a basic requirement; an abrupt interruption could lead to serious consequences. An outage in communication between the numerous devices could have similar outcomes. However, it is sufficient if a single production support system, such as a logistics service, fails. Furthermore, downtime can occur due to lack of materials.

I. Privacy

Privacy in the IoT remains a hot research topic. Due to the complexity of the topic, this section follows the highly regarded paper by Ziegeldorf et al. about privacy threats in the IoT [32]. The threats, that mainly concern users, are briefly summarized below. First, the threat identification arises when an identifier can be linked with an individual and data about him/her. This includes the identification of humans as well as devices, for example, by fingerprinting. Second, as devices become more interconnected, they can be queried over the network, allowing attackers to gather information and characteristics about existing devices, called inventory attacks. The resulting inventory list of a factory could be interesting for competitors, for example. Another threat occurs during life cycle transitions. When device users change, the (sensitive) data of the previous user is often still there; a function for disposal is also often missing for full data wipe. The possibility to link two or more previously separated systems poses a further threat. The linkage of their data sources may reveal (truthful or erroneous) information without consent of the user. There are several ways to locate and track users through the device or an associated service. This may be necessary for the functionality of the application, but it also introduces threats; for example, the performance of employees can be tracked. The way users interact with IoT devices and the information they present in response can also result in the disclosure of sensitive data. For example, it might not be possible to privately interact with a voice-controlled device in public space. Last, there is a threat of user profiling to correlate their interests or behavior with other profiles and data.

J. Unintentional / Disaster

Unfavorable conditions can affect the functionality of devices. It matters little whether these are caused by environmental factors or disasters, if only the impact is considered. Threats can arise from the operation of a device outside of the specified parameters, for example, in unapproved temperature range (e.g., due to fire or lack of switching cabinet cooling) or voltage range (e.g., due to lightning strike or fluctuations in the power supply). Other categories are mechanical stress (earthquake or misuse by employees), pollution (dust), and corrosion.

VII. Attack and Weakness Categorization

There are countless ways to attack an IIoT device and new attack techniques are also constantly being discovered. Therefore, we categorized common attack techniques and weaknesses (see Figure 5). The previously conducted decomposition of a device into its individual components in Section IV was leveraged here to get a comprehensive understanding of the attack surface. Below, these categories are described by exemplary attack vectors throughout the life cycle of an IIoT device, namely design, production, distribution, setup, operation, maintenance, and end of life. Note that not all attacks can be clearly assigned to a single category.

A. Hardware Attacks

A closer examination of hardware attacks revealed that they can be subdivided into chip-, PCB- and device-level attacks. It should be noted here that this category is different from the commonly used designation *physical attacks*, as physical access is not always necessary.

a) Chip-Level Attacks: Attacks against chips, such as microcontrollers and custom ASICs, start at the design stage and also during production. Various actors have the ability to maliciously modify the design and insert a hardware Trojan or backdoor, for example. This can be done by an untrusted IP vendor, foundry, or design facility. Another threat source may be a compiler or Computer-Aided Design (CAD) tool. For example, a modified version of Apple's Xcode infected thousands of iOS apps with malware [33]. The facilities involved also have the ability of IP theft, which enable product piracy. There is still the possibility of reverse engineering in the foundry or later in operation, if someone has no access to the design files. There are further attack opportunities from distribution to the end of life. First, it is possible to tamper with data in chips by exploiting physical access. Such an attack

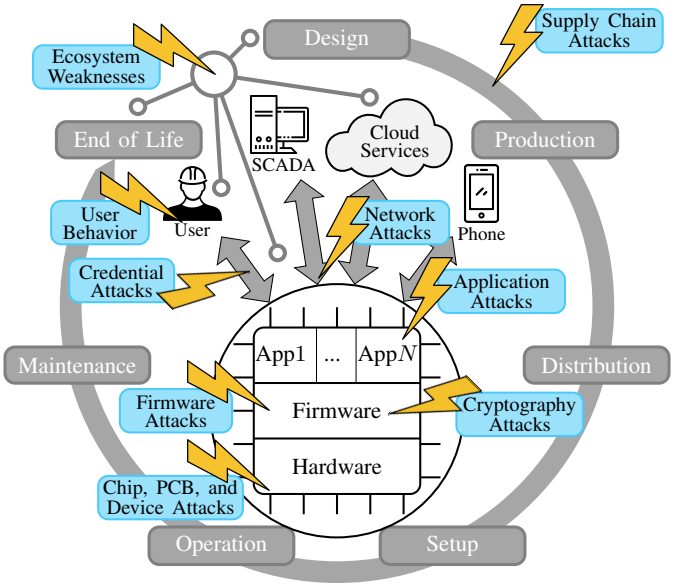


Figure 5. IoT devices can be attacked during the entire product life cycle. The various types of attacks on an IoT device, illustrated in the circle, are shown in the blue boxes.

was demonstrated at the Chaos Communication Congress in late 2019, when attackers were able to reroute orders from a microcontroller distributor. After the installation of a backdoor, the controllers could be shipped to the actual customers without them noticing [34]. Furthermore, people with (temporary) physical access to the chip can conduct microprobing or fault injection attacks. The former uses microscopic needles to probe internal wires, the latter deliberately injects a fault in a system to change its behavior. Both use invasive attack methods, although fault injection is also possible with semi-invasive (e.g., focused laser beam) and non-invasive (e.g., clock and voltage glitching) attacks [35]. These techniques are used to gain secret information or bypass security features, for instance. Another possibility to extract secrets are side-channel attacks. They observe parametric behaviors, such as power consumption or timing information, of a specific implementation of an algorithm to leak encryption keys, for example.

b) PCB-Level Attacks: First, attackers might be interested in the PCB design to understand or clone a product causing a threat to IP. They can get the design by stealing the CAD files or by utilizing reverse engineering techniques. Second, a backdoor can be implanted by inserting a malicious component on the PCB or by replacing an existing one. There are several ways to accomplish the former [36]. Attackers must initially insert an extra component into one of the different design files. In the second step, it must be mounted on the PCB. This can take place, for example, during production, at repair and rework stations, or after the PCB has been delivered to a warehouse. Another option is to add a completely new component. The project TPM Genie demonstrated that TPMs can be attacked by an interposed device [37]. PCBs usually contain interfaces for verifying the design and testing. The commonly integrated protocol JTAG allows to program memory or debug controllers, among others. Attackers can exploit such interfaces to dump the firmware, resulting in an IP theft, or overwrite local storage, enabling firmware and data manipulation.

c) Device-Level Attacks: The third group contains more general attack vectors and weaknesses. It includes sabotage attacks that can be any physical change to hardware that has a malicious impact. Threats can also arise unintentionally. A service technician might replace a burnt out PCB or a defective engine with a spare part that was not purchased from the original manufacturer for price reasons resulting in faulty operation, for example. There are other attack possibilities depending on the specific component, e.g., some are vulnerable to magnetic field attacks. One such component are displays. A common weakness occurring in IIoT devices is permissive displayed information to everyone (e.g., the firmware version), if authentication is required at all. Another component are USB ports, which allow insertion of malicious USB sticks that either imitate a keyboard to inject commands or destroy the port or entire device by putting a high voltage on the lines.

B. Firmware Attacks

The firmware enables several ways to attack a device. Attackers can exploit vulnerabilities in one of its components, for example, a network stack. Within a short time, dozens of vulnerabilities were discovered in famous IoT and OT TCP/IP stacks leading to DoS or Remote Code Execution (RCE); the findings were named as INFRA:HALT, NAME:WRECK, NUMBER:JACK, AMNESIA:33, and Ripple20 [38]–[42]. If a vulnerable firmware is already fixed, attackers may try to downgrade it to a previous version with security flaws. Another possibility is to utilize the firmware update mechanism to install a manipulated version including a backdoor, for example. Especially the hardware infrastructure including the boot process is often vulnerable, as concluded by [43].

C. Application Attacks

There are numerous ways to attack applications. However, the techniques depend on the utilized technologies such as web servers, databases, and used programming language. A few of these techniques are described below. A (remote) code execution attack exploits vulnerabilities in a running process, allowing the execution of any instruction on a system. Recently, dozens of vulnerabilities were found in IoT and OT RTOSs exploiting bad memory allocations [44]. Code injection attacks that take advantage of insufficiently sanitized input have a similar goal, e.g., SQL injection. Web applications running on the IIoT devices are also exposed to typical vulnerabilities such as Cross-Site Scripting (XSS), broken access control, and insecure deserialization [45]. Furthermore, for CPSs, data injection attacks should be considered, e.g., from spoofed sensor values. There are also attacks against AI applications. It is conceivable to attack machine learning models and data sets during their design or operation. Models

can be sabotaged or poisoned and data sets or their labels can be manipulated in order to reduce accuracy, for example.

D. Cryptography Attacks

Cryptography is a rather cross-disciplinary domain, as it is used for encrypted local storage, authentication, and secure communication, among others. In order to abstract from the various applications, attacks against cryptography have been grouped into this category. A common weakness is the use of deprecated cryptographic algorithms. Especially OT devices with a long lifetime employ these obsolete algorithms, such as the Data Encryption Standard (DES). This can also be suspected from the fact that still many modern microcontrollers implement hardware accelerators for those algorithms. In addition, algorithm-specific parameters, such as the key size, are often selected incorrectly. Examples include a poorly chosen curve in elliptic curve cryptography, and a weak block cipher mode for symmetric algorithms. Cryptography depends heavily on random numbers, which potentially introduces further weaknesses. These include weak pseudo-random number generators, lack of sufficient entropy, and the reuse of nonces (i.e., number that can be used only once). When handling passwords, design flaws can also occur that facilitate brute force or rainbow table attacks.

E. Network Attacks

The integration of network interfaces into OT devices has significantly increased their attack surface. One of the major concerns of ICS operators are (D)DoS attacks, as unavailability may cause production downtime, which in turn results in financial damage. In addition to the classic attack techniques, attacks on wireless communication technologies should also be considered, e.g., radio jamming. IIoT devices can be the target of attacks themselves as well as being used to launch attacks against SCADA systems or cloud services. Another risk are the various automation protocols used within OT systems that were designed decades ago and do not support security controls. As a result, man-in-the-middle and replay attacks can often be conducted because of missing authentication and unencrypted communication. Last but not least, network interfaces enable attack preparation attacks, such as port scans or device fingerprinting.

F. User Behavior

Users can introduce threats at different stages in the device life cycle. During setup, technicians might misconfigure the device, for example, by disabling security features, or neglect to create a configuration backup and keep this secure. Afterwards, operators might install malware, for instance, to mine cryptocurrencies. The mishandling of warnings and errors, as well as the incorrect use of the device in general, can also lead to further threats. Likewise, errors can occur during maintenance. Technicians are usually responsible for installing firmware updates, as automatic over-the-air updates are problematic due to safety requirements. They are also responsible for deleting all data at the end of life.

a) Credential Attacks: Credentials are an important asset of users and, therefore, a major target of attackers. With some exceptions, such as improper storage and cleartext transmission, users are often the weak point to get them. Contrary to recommendations, default and simple passwords are still used, as well as shared passwords with colleagues. Many users also disregard social engineering attacks such as phishing or shoulder surfing.

G. Ecosystem Weaknesses

This category summarizes weaknesses in the ecosystem that arise due to incomplete system design and the heterogeneous structure of the IIoT. Some manufacturers still do not provide software updates, and if they do, they are rare. Especially IIoT device manufacturers rarely implement vulnerability disclosure policies for fear of damaging their reputation. Further bad practices include hard-coded passwords, developer backdoors, and compromisable procedures for lost credentials. Another vulnerability is the implicit trust between components within the device and also in the entire ecosystem. The latter are still far from implementing the goal of zero trust. A huge challenge is interoperability; solutions are complicated by the numerous legacy standards in ICSs. Other devices and (third-party) IoT services for device life cycle management, telemetry, and analytics are therefore blindly trusted.

H. Supply Chain Attacks

Several supply chain attacks have already been mentioned in the previous categories. For the sake of clarity, these are again summarized and expanded in this section. A frequent target is malicious modification of functionality by hardware, software or data. Hardware logic insertion occurs during design and production stage, and replacing a hardware component with a malicious one is feasible during production and in all later stages. The manipulation or insertion of software code is conceivable in all life phases, e.g., by malicious third-party libraries or by a manipulated firmware. Data used to train machine learning models, for example, can also be compromised. IP theft of hardware and software is also possible in all stages; in the case of hardware, this is particularly achievable during design and production, as these steps are often outsourced to external contractors. Outsourcing the production additionally enables cloning of provisioning data and unauthorized overproduction; discarded or defective equipment could moreover end up on the gray market. A further threat to production is the use of counterfeit components or generally inferior material.

VIII. Recommended Procedure for Analyzing Threats

Last, we recommend a procedure for the threat analysis of IIoT devices. For this purpose, the previously discussed aspects are revisited and put in order. Note, this recommendation can be seen as complementary to existing Threat Analysis and Risk Assessment (TARA) methodologies, such as IEC 62443-3-2, in order to facilitate a device-oriented procedure.

1. System Analysis

The first step is to analyze the IIoT device in depth. All hardware and software components and data files, as described in Section IV, should be collected. This enables a component-by-component threat and vulnerability analysis later on. In addition, requirements and (environmental) conditions should be gathered. Requirements may originate, for example, from the operating location, industrial sector, or use in critical infrastructures.

2. System Interaction Overview

As previously stated, connectivity is a central element of IIoT devices. Therefore, an overview of all interactions with the device is necessary for a thorough threat analysis. This diagram should contain all actors including humans, cloud services, and other machines. This can be supported by an additional use case diagram to capture when and why an actor interacts with the system.

So far, we have only referred to users in general terms; especially in the case of industrial systems, it is sensible to differentiate between them according to their role. For example, the commissioning, operation, and maintenance personnel are often not the same. The goal is to grant each group only the least required privileges. The authorization of certain network interfaces should also be considered. This is especially important for industrial protocols, such as PROFINET. While most IoT applications allow the implementation of security measures manually, it is not possible with these proprietary protocols, as compatibility with other manufacturers must be maintained.

3. Asset Identification

The third step is the identification of assets based on the various types presented in Section V. The assets can be best identified by considering the interests of the different stakeholders; again, the different roles of users should be distinguished. This step helps to understand what values to each party. It also allows to rank the criticality of assets, which can be used for defining the impact of attacks.

Furthermore, it is useful to determine the security goals in general and also per asset. This can support the development of countermeasures later. For CPSs, availability is often most important for safety reasons. For edge devices, confidentiality could be rated higher, as they aggregate information. However, in low-power IIoT devices that must comply with real-time requirements, it is not always feasible to implement the most secure countermeasure. Thus, this initial assessment can be used to choose the right measure. For example, sensitive data in transit (e.g., credentials) will be encrypted, less critical data (e.g., sensor values) will only be protected against manipulation using a message authentication code.

4. Threat Source Identification

Threat sources can be best identified from the perspective of the attacker to find out what they are targeting and why. There are several types of attackers with different capabilities, attack techniques and motives. We classified various types of threat sources and their respective intentions in [1]. This is useful for deliberately including or excluding types of attacks. For IIoT devices in critical infrastructures, the more complex hardware and supply chain attacks should be addressed.

5. Threat and Vulnerability Identification

The primary task of a threat analysis is the identification of threats and vulnerabilities. However, the preparatory work from the previous four steps should significantly accelerate and enhance this process. As said before, the system analysis should allow the detection of threats to single hardware components, such as PCBs and actuators. The list of software components can also be used to search for publicly known vulnerabilities. Numerous ways to attack them were presented in Section VII. Additionally, it is possible to reflect on how the various threats, presented in Section VI, may arise. Using attack trees, it is also possible to graphically show how assets can be attacked. Figure 6 shows a sample tree for attacking the manufacturer key that is used for firmware updates, for instance. One option is to dump the memory, which can be achieved either by (remote) code execution or by exploiting physical access. In the latter case, it also depends on where the key is stored; for microcontroller internal memory, a read-out protection may have to be broken.

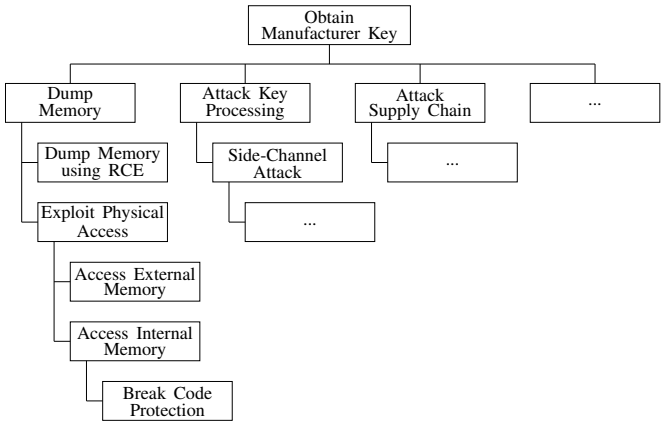


Figure 6. Fraction of an example attack tree on an asset.

The system interaction and use case diagrams can be utilized to create data flow diagrams in order to identify threats to data in transit. These can be used in combination with threat modeling techniques such as STRIDE [46], a mnemonic for threats against the security goals authenticity, integrity, non-repudiation, confidentiality, availability, and authorization. Furthermore, penetration testing can be used to discover additional vulnerabilities as well as to verify those already identified and show their severity.

Documenting the discovered threats is also crucial. We have had the experience that detailed attack scenarios can be better understood in retrospect. Table I shows an excerpt of a potential attack scenario description. This example lists threats resulting from a default PIN used in an electrical actuator controlling a valve. The PIN is required for authentication

TABLE I. EXCERPT OF AN EXAMPLE LIST OF ATTACK SCENARIOS.

No.	Vulnerability	Threat (Category)	Attack Vector Interface	Action	Note
1	Default PIN (users did not change it)	Login as administrator (obtaining of control)	Bluetooth app, local HMI	Authenticate using default PIN	PIN can be found in the manual
1.1		Moving actuator (application malfunction)	Bluetooth app	Open/close valve	
1.2		Blocking remote control (denial of service)	Bluetooth app	Change control system communication parameter	
1.3		Manipulating user database (data tampering)	Local HMI	View/change/add/delete user accounts	

at the attached HMI and in a Bluetooth mobile app. An unchanged PIN would allow attackers to authenticate as the administrator.

6. Vulnerability and Risk Assessment

The final step is to gather all the information to evaluate the vulnerability and assess the risk. A popular method for vulnerability assessment is the Common Vulnerability Scoring System (CVSS). It incorporates exploitability metrics, such as the attack vector, and the impact on confidentiality, integrity, and availability. The example in Table I indicates that several threats arise from a single vulnerability. As the impact can vary per interface, an overview of all scenarios is important for a proper scoring. Finally, the risk assessment may include further criteria such as likelihood and (business) impact.

IX. CONCLUSIONS

Targeted attacks on ICSs, including those in critical infrastructures, increased recently. IIoT devices that merge the previously separated areas of IT and OT additionally increase the attack surface. The consequences of an attack can be dramatic, as OT equipment controls physical processes that, in case of compromised safety, can cause harm to humans, machines, and the environment. Therefore, it is even more essential that IIoT devices are properly secured. However, this is not often the case in reality. One reason is that manufacturers are extensively provided with literature on best practices rather than threat analysis techniques for their devices.

In this paper, we presented a systematic and holistic procedure for analyzing the attack surface and threats of IIoT devices throughout the product life cycle. First, an arbitrary IIoT device was decomposed into its components for this purpose. This itemization of hardware and software components as well as data types is essential to ensure that no attack vectors are overlooked. Afterwards, the assets were analyzed from the perspective of different stakeholders in order to identify everything that is valuable and worth protecting. The provided comprehensive categorization of threats shows an overview of possible threats and their consequences. The attack techniques are almost innumerable and are also constantly expanding. Therefore, we categorized attack techniques and weaknesses that are frequently exploited. This included not only common attack vectors in communications and web applications, but also attacks against the supply chain and the various hardware components. Finally, the threat analysis procedure was described, which enables manufacturers and operators of IIoT devices to identify and evaluate attack vectors. Since the threat categorization considers all assets and the attack categorization addresses all components, the proposed analysis technique seems to be valuable. In the next steps, the procedure will be further validated.

ACKNOWLEDGMENT

The research project “Intelligent Security for Electrical Actuators and Converters in Critical Infrastructures (iSEC)” is a collaboration of SIPOS Aktorik GmbH, Grass Power Electronics GmbH and OTH Amberg-Weiden. It is supported and funded by the Bavarian Ministry of Economic Affairs, Regional Development and Energy.

REFERENCES

[1] S. Liebl, L. Lathrop, U. Raithel, M. Söllner, and A. Aßmuth, “Threat Analysis of Industrial Internet of Things Devices,” in Proceedings of the Eleventh International Conference on Cloud Computing, GRIDs, and Virtualization, Nice, France, Oct. 2020.

[2] “Threat landscape for industrial automation systems: Statistics for H2 2020,” Kaspersky ICS CERT, Mar. 2021. [Online]. Available: https://ics-cert.kaspersky.com/media/Kaspersky-Threat-landscape-for-industrial-automation-systems-statistics-for-H2-2020-En.pdf [accessed: 2021-12-01]

[3] J. Santagate, R. Glaisner, and R. Westervelt, “Operational Cybersecurity for Digitized Manufacturing: Emerging Approaches for the Converged Physical-Virtual Environment,” IDC, Aug. 2019. [Online]. Available: https://www.fortinet.com/content/dam/fortinet/assets/white-papers/wp-idc-operational-cybersecurity-for-digitized-manufacturing.pdf [accessed: 2021-12-01]

[4] M. Bakuei, R. Flores, Lord Remorin, and F. Yarochkin, “2020 Report on Threats Affecting ICS Endpoints,” Trend Micro Research, Jun. 2021. [Online]. Available: https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/2020-report-ics-endpoints-as-starting-points-for-threats [accessed: 2021-12-01]

[5] “DarkSide Ransomware: Best Practices for Preventing Business Disruption from Ransomware Attacks,” Cybersecurity & Infrastructure Security Agency (CISA), May 2021. [Online]. Available: https://us-cert.cisa.gov/ncas/alerts/aa21-131a [accessed: 2021-12-01]

[6] “Compromise of U.S. Water Treatment Facility,” Cybersecurity & Infrastructure Security Agency (CISA), Feb. 2021. [Online]. Available: https://us-cert.cisa.gov/ncas/alerts/aa21-042a [accessed: 2021-12-01]

[7] B. Dorsemaine, J.-P. Gaulier, J.-P. Wary, N. Kheir, and P. Urien, “Internet of Things: A Definition & Taxonomy,” in 2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies, Cambridge, United Kingdom, 2015, pp. 72–77.

[8] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, “Industrial Internet of Things: Challenges, Opportunities, and Directions,” IEEE Trans. Ind. Inf., vol. 14, no. 11, pp. 4724–4734, Nov. 2018, doi: 10.1109/TII.2018.2852491.

[9] “Industrial Internet of Things (IIoT) leading use cases worldwide as of 2019*,” PTC, Chart. May 1, 2019. [Online]. Available: https://www.statista.com/statistics/1102202/industrial-iot-worldwide-use-cases/ [accessed: 2021-12-01]

[10] H. Boyes, B. Hallaq, J. Cunningham, and T. Watson, "The industrial internet of things (IIoT): An analysis framework," Computers in Industry, vol. 101, pp. 1–12, Oct. 2018, doi: 10.1016/j.compind.2018.04.015.

[11] A. Hahn, "Operational Technology and Information Technology in Industrial Control Systems," in Cyber-security of SCADA and Other Industrial Control Systems, 2016, pp. 51–68, DOI: 10.1007/978-3-319-32125-7_4.

[12] K. Tange, M. De Donno, X. Fafoutis, and N. Dragoni, "A Systematic Survey of Industrial Internet of Things Security: Requirements and Fog Computing Opportunities," in IEEE Communications Surveys & Tutorials, vol. 22, no. 4, pp. 2489-2520, Fourthquarter 2020, doi: 10.1109/COMST.2020.3011208.

[13] P. Varga, S. Plosz, G. Soos, and C. Hegedus, "Security threats and issues in automation IoT," in 2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS), Trondheim, Norway, May 2017, pp. 1–6. doi: 10.1109/WFCS.2017.7991968.

[14] A. W. Atamli and A. Martin, "Threat-Based Security Analysis for the Internet of Things," in 2014 International Workshop on Secure Internet of Things, Wroclaw, Poland, Sep. 2014, pp. 35–43. doi: 10.1109/SIoT.2014.10.

[15] M. Abomhara and G. M. Køien, "Cyber security and the internet of things: Vulnerabilities, threats, intruders and attacks," Journal of Cyber Security and Mobility, vol. 4, no. 1, pp. 65–88, 2015.

[16] J. Wurm, K. Hoang, O. Arias, A. Sadeghi, and Y. Jin, "Security analysis on consumer and industrial IoT devices," 2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC), Macau, 2016, pp. 519-524, DOI: 10.1109/ASPDAC.2016.7428064.

[17] "Mapping of IoT security recommendations, guidance and standards," Department for Digital, Culture, Media & Sport, Oct. 2018. [Online]. Available: https://www.gov.uk/government/publications/mapping-of-iot-security-recommendations-guidance-and-standards [accessed: 2021-12-01]

[18] "Internet of Things (IoT) – ENISA," European Union Agency for Cybersecurity. https://www.enisa.europa.eu/topics/iot-and-smart-infrastructures/iot [accessed: 2021-12-01]

[19] "Baseline Security Recommendations for Internet of Things in the context of Critical Information Infrastructures," European Union Agency for Cybersecurity, Nov. 2017. [Online]. Available: https://www.enisa.europa.eu/publications/baseline-security-recommendations-for-iot [accessed: 2021-12-01]

[20] A. K. Sikder, G. Petracca, H. Aksu, T. Jaeger, and S. Uluagac, "A survey on sensor-based threats and attacks to smart devices and applications," IEEE Communications Surveys & Tutorials, vol. PP, pp. 1–1, Mar. 2021, doi: 10.1109/COMST.2021.3064507.

[21] IEEE (Internet of Things), European Commission (Agile IoT), and Eclipse IoT Working Group, "Distribution of operating systems used for Internet-of-Things (IoT) devices, as of 2016," Apr. 2016. [Online]. Available: https://www.statista.com/statistics/659581/worldwide-internet-of-things-survey-operating-systems/ [accessed: 2021-12-01]

[22] P. Kietzmann, L. Boeckmann, L. Lanzieri, T. C. Schmidt, and M. Wählisch, "A performance study of crypto-hardware in the low-end IoT," in Proceedings of the 2021 international conference on embedded wireless systems and networks, USA, 2021, pp. 79–90.

[23] "Glossary – ENISA," European Union Agency for Cybersecurity. https://www.enisa.europa.eu/topics/threat-risk-management/risk-management/current-risk/risk-management-inventory/glossary [accessed: 2021-12-01]

[24] "DBIR: 2021 Data Breach Investigations Report," Verizon, 2021. [Online]. Available: https://enterprise.verizon.com/resources/reports/2021-data-breach-investigations-report.pdf [accessed: 2021-12-01]

[25] "'BrickerBot' Results In Permanent Denial-of-Service," radware, Apr. 04, 2017. [Online]. Available: https://www.radware.com/security/ddos-threats-attacks/brickerbot-pdos-permanent-denial-of-service/ [accessed: 2021-12-01]

[26] M. Abrams and J. Weiss, "Malicious Control System Cyber Security Attack Case Study– Maroochy Water Services, Australia," MITRE, Aug. 2008.

[27] "SB-327 Information privacy: connected devices," California Legislative Information, Sep. 28, 2018. https://leginfo.legislature.ca.gov/faces/billNavClient.xhtml?bill_id=201720180SB327 [accessed: 2021-12-01]

[28] M. Henschke, "Update-Pflicht für digitale Geräte: Was haben Kunden davon? (Mandatory updates for digital devices: What's in it for customers?)," Hamburger Abendblatt, Jun. 27, 2021. [Online]. Available: https://www.abendblatt.de/ratgeber/article232640687/update-pflicht-digitale-geraete-smartphone-verbraucher.html [accessed: 2021-12-01]

[29] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," Security & Privacy, IEEE, vol. 9, no. 3, pp. 49–51, 2011.

[30] S. Miller, N. Brubaker, D. Kapellmann Zafra, and D. Caban, "TRITON Actor TTP Profile, Custom Attack Tools, Detections, and ATT&CK Mapping," FireEye, April 10, 2019. [Online]. Available: https://www.fireeye.com/blog/threat-research/2019/04/triton-actor-ttp-profile-custom-attack-tools-detections.html [accessed: 2021-12-01]

[31] A. Cherepanov and R. Lipovsky, "Industroyer: Biggest threat to industrial control systems since Stuxnet," welivesecurity, June 12th, 2017. [Online]. Available: https://www.welivesecurity.com/2017/06/12/industroyer-biggest-threat-industrial-control-systems-since-stuxnet/ [accessed: 2021-12-01]

[32] J. Ziegeldorf, O. Morchon, and K. Wehrle, "Privacy in the Internet of Things: Threats and Challenges," Security and Communication Networks, vol. 7, Dec. 2014, doi: 10.1002/sec.795.

[33] C. Xiao, "Novel Malware XcodeGhost Modifies Xcode, Infects Apple iOS Apps and Hits App Store," paloaltonetworks.com, Sep. 17, 2015. https://unit42.paloaltonetworks.com/novel-malware-xcodeghost-modifies-xcode-infects-apple-ios-apps-and-hits-app-store/ [accessed: 2021-12-01]

[34] T. Roth, "36C3 - TrustZone-M(eh): Breaking ARMv8-M's security," Dec. 28, 2019. [Online]. Available: https://media.ccc.de/v/36c3-10859-trustzone-m_eh_breaking_armv8-m_s_security [accessed: 2021-08-04]

[35] N. Beringuier-Boher et al., "Voltage Glitch Attacks on Mixed-Signal Systems," in 2014 17th Euromicro Conference on Digital System Design, Verona, Italy, Aug. 2014, pp. 379–386. doi: 10.1109/DSD.2014.14.

[36] S. H. Russ and J. Gatlin, "Ways to hack a printed circuit board: PCB production is an underappreciated vulnerability in the global supply chain," IEEE Spectrum, vol. 57, no. 9, pp. 38–43, 2020, doi: 10.1109/MSPEC.2020.9173902.

[37] J. Boone, "TPM Genie: Interposer Attacks Against the Trusted Platform Module Serial Bus," NCC Group, White Paper, Mar. 2018. [Online]. Available: https://github.com/nccgroup/TPMGenie/blob/master/docs/NCC_Group_Jeremy_Boone_TPM_Genie_Whitepaper.pdf [accessed: 2021-12-01]

[38] D. dos Santos, S. Dashevskyi, A. Amri, J. Wetzels, A. Karas, S. Menashe, and D. Vozniuk, "INFRA:HALT - Jointly discovering and mitigating large-scale OT vulnerabilities," Forescout, 2021. [Online]. Available: https://www.forescout.com/resources/infrahalt-discovering-mitigating-large-scale-ot-vulnerabilities/ [accessed: 2021-12-01]

[39] D. dos Santos, S. Dashevskyi, A. Amri, J. Wetzels, S. Oberman, and M. Kol, "NAME:WRECK - Breaking and fixing DNS implementations," Forescout, 2021. [Online]. Available: https://www.forescout.com/company/resources/namewreck-breaking-and-fixing-dns-implementations/ [accessed: 2021-12-01]

[40] "NUMBER:JACK - Weak ISN Generation in Embedded TCP/IP Stacks," Forescout, 2021. [Online]. Available: https://www.forescout.com/company/resources/numberjack-weak-isn-generation-in-embedded-tcpip-stacks/ [accessed: 2021-12-01]

[41] D. dos Santos, S. Dashevskyi, J. Wetzels, and A. Amri, "AMNESIA:33 - How TCP/IP Stacks Breed Critical Vulnerabilities in IoT, OT and IT Devices," Forescout, 2020. [Online]. Available: https://www.forescout.com/company/resources/amnesia33-how-tcp-ip-stacks-breed-critical-vulnerabilities-in-iot-ot-and-it-devices/ [accessed: 2021-12-01]

[42] M. Kol and S. Oberman, "Ripple20," JSOF, White Paper, 2020. [Online]. Available: https://www.jsof-tech.com/wp-content/uploads/2020/06/JSOF_Ripple20_Technical_Whitepaper_June20.pdf [accessed: 2021-12-01]

[43] G. Hernandez and D. Buentello, "Smart nest thermostat a smart spy in your home," 2014.

[44] "'BadAlloc' – Memory allocation vulnerabilities could affect wide range of IoT and OT devices in industrial, medical, and enterprise networks," Microsoft Security Response Center, Apr. 29, 2021. https://msrc-blog.microsoft.com/2021/04/29/badalloc-memory-allocation-vulnerabilities-could-affect-wide-range-of-iot-and-ot-devices-in-industrial-medical-and-enterprise-networks/ [accessed: 2021-12-01]

[45] "OWASP Top 10 - 2017," OWASP, 2017. [Online]. Available: https://owasp.org/www-pdf-archive/OWASP_Top_10-2017_(en).pdf.pdf [accessed: 2021-12-01]

[46] A. Shostack, Threat modeling: designing for security. Indianapolis, IN: Wiley, 2014.

An Access Control Architecture for Securing Multi-Tenancy Cloud Environments

Ronald Beaubrun
Department of Computer Science and Software Engineering
Laval University
Quebec, Canada
e-mail: ronald.beaubrun@ift.ulaval.ca

Alejandro Quintero
Department of Computer and Software Engineering
Polytechnique Montreal
Montreal, Canada
e-mail: alejandro.quintero@polymtl.ca

***Abstract*— In multi-tenancy cloud environments, physical resources are transparently shared by multiple Virtual Machines (VMs) belonging to multiple users. Implementing an efficient access control mechanism in such environments can prevent unauthorized access to the cloud resources. In this paper, we propose an access control mechanism called CloudGuard that provides scalable and secure access control to the cloud in the context of multi-tenancy cloud environments. Such a mechanism prevents malicious tenants from generating and sending unauthorized traffic to the cloud network. Numerical results show that CloudGuard offers better system throughput in critical or high-risk multi-tenancy cloud network environments, where the amount of intrusion traffic is high.**

Keywords-access control; cloud computing; hypervisor; multi-tenancy; security, virtual machine.

I. INTRODUCTION

Cloud computing is a flexible and cost-effective platform for providing business and consumer services over the Internet [1][2][3]. Such a platform is utilized by multiple customers who share computing resources, including CPU time, network bandwidth, data storage space, with other users, which refers to multi-tenancy [4]. By multi-tenancy, Clouds provide simultaneous, secure hosting of services for various customers utilizing the same infrastructure resources [5][6]. However, in multi-tenancy cloud environments, one customer can gain unauthorized access to the information of other customers. In this context, it is important to control the access of network entities to such information.

Access control is a security feature that controls how users and systems communicate and interact with other systems and resources. In general, there are three types of access control: physical access control, technical access control and administrative access control [7][8]. Physical access control refers to the implementation of security measures in a defined structure in order to prevent unauthorized access to sensitive materials. Examples of such control include: security guards, picture IDs, locked and dead-bolted steel doors, biometrics, closed-circuit surveillance cameras and motion or thermal alarm systems. Technical access control employs the technology as a basis for controlling the access to sensitive information throughout a physical structure and over a network. Examples of technical access control are: encryption, smart cards, network authentication, Access Control Lists (ACLs), and file integrity auditing software. Administrative access control defines the human factors of security. All levels of the personnel within an organization are involved in such control. Administrative access control also determines which users have access to which resources and information.

The above types of access control can be integrated into security architectures in order to preserve the integrity, confidentiality and availability of resources that are collocated in multi-tenancy cloud environments. In this paper, we investigate the use of technical access control for proposing a secure access control mechanism in the context of multi-tenancy cloud environments. Such a mechanism will prevent malicious insiders from generating and sending unauthorized traffic to the cloud network.

The rest of the paper is organized as follows. Section II introduces the context and background related to access control in multi-tenancy cloud environments. Section III discusses the main existing methods and models for controlling access in multi-tenancy cloud environments. Section IV presents the main assumptions and principles of the proposed architecture. Section V illustrates and explains a use case scenario. Section VI evaluates the performance of the proposed access control architecture in terms of latency and system throughput. Section VII gives some concluding remarks and perspectives.

II. CONTEXT AND BACKGROUND

As illustrated in Figure 1, a multi-tenant cloud service provider has three essential elements: the cloud manager, the hypervisor and the Virtual Machines (VMs) [9]. The cloud manager is a console of management provided for clients in order to manage their cloud infrastructure, which means creating, shutting down, or starting the instances. The hypervisor, also called Virtual Machine Manager (VMM), allows multiple operating systems (guests or virtual machines) to run concurrently on a host server. Its main responsibility is to manage the application's operating systems (OSs) and their use of the system resources (e.g., CPU, memory and storage). Its role is to control the host processor and resources, and also to allocate what is needed to each operating system.

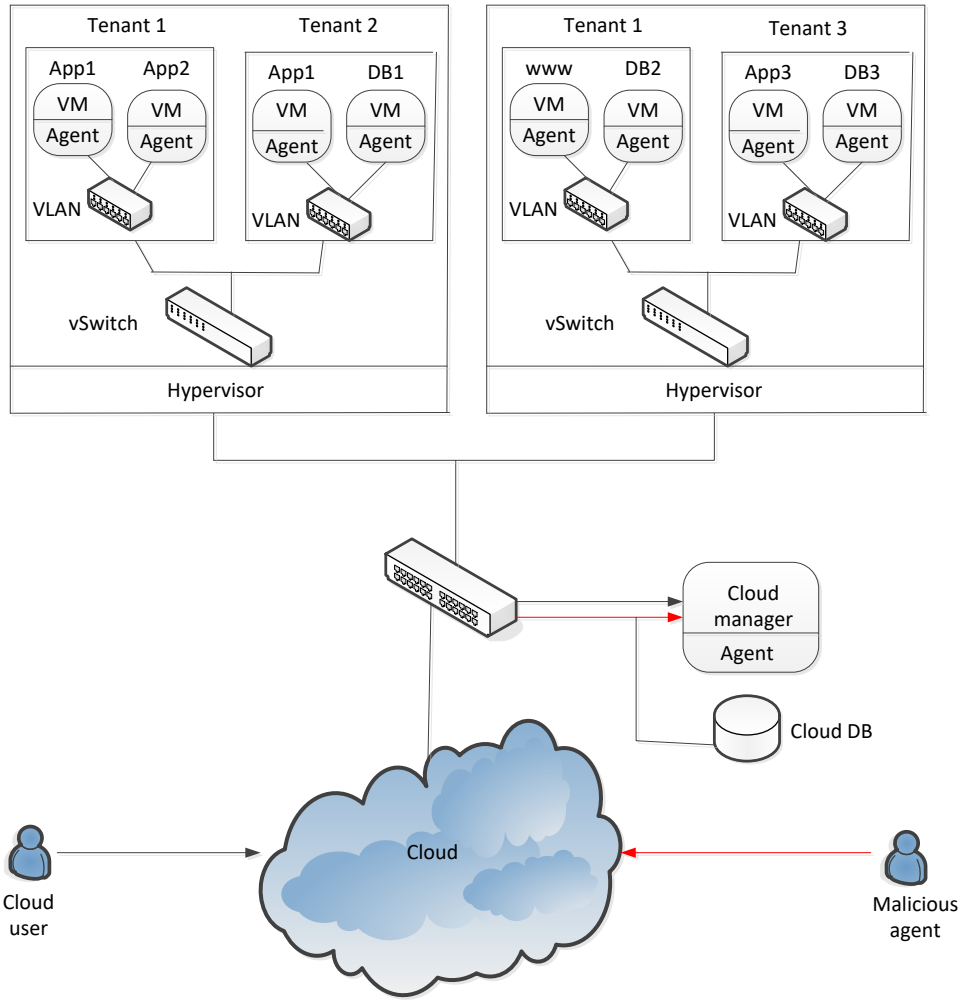


Figure 1. A model for a multi-tenant cloud service provider [9].

A VM is an isolated guest operating system installation within a normal host operating system. In this context, each client may have one or more VMs, as one physical server can host several VMs. In such an environment, one client can send unlimited amount of traffic to another client. Accordingly, a malicious agent can rent a VM on the same host where the target VM resides. This malicious agent can send unauthorized traffic to the target VM and violate the security of the target VM [10]. Intrusion traffic is the amount of traffic that is generated by a malicious VM, and the target of such traffic is to penetrate the vulnerabilities of the destination VM. In principle, intrusion traffic is unpredictable in a multi-tenancy cloud network.

The unauthorized traffic may contain some script or malware which violates the confidentiality or the integrity of the target VM data. Sending such traffic to another VM makes it possible to perform other sorts of attacks. For instance, a malicious agent who owns a VM can perform VM Hopping over another user who is co-located at the same host. With VM hopping, an attacker has the control of one VM and tries to gain the control of another VM. VM hopping allows an attacker to move from one virtual server to the next one, or even to gain the root access to the physical hardware. VM hopping is a considerable threat because several VMs can run on the same host, which makes them the targets for the attacker. By performing this attack, a malicious user can violate the security and steal the data of other users who are located at the same server while compromising the hypervisor file system [11].

In addition, the malicious insider can perform Denial of Service (DoS) attacks. These kinds of attacks exhaust the resources of the cloud network, such as bandwidth and computing power, by sending large amount of unauthorized traffic to other VMs.

III. EXISTING METHODS AND MODELS

In this section, we discuss the main existing methods and models for controlling access in the context of multi-tenancy cloud environments.

A. Distributed access control

The Distributed Access Control (DAC) architecture was proposed by Thomas *et al*. [12]. As illustrated in Figure 2, such an architecture has three main components: the Cloud Service Provider (CSP), the Cloud Service Consumer (CSC), and the Identity Provider (IdP). The CSC requests the resources or services hosted by the CSPs. In this stage, the CSC should be first authenticated to ensure that unauthorized users do not access the services from the CSP. The main responsibility of the CSP is to host and to provide various services or resources to the CSCs. As a result, for avoiding illegal and unauthorized access by CSCs, proper authorization and authentication of CSCs are required.

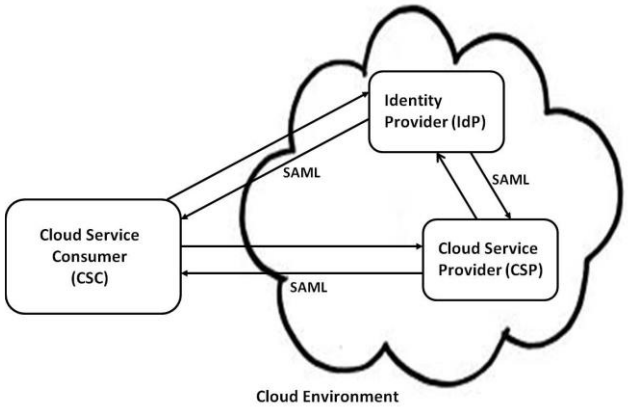


Figure 2. Distributed Access Control architecture [12].

Moreover, in DAC architecture, the IdP plays a great role since it generates identity tokens to the users. By using this identity token, a user can request the access to the cloud. Such a user may subscribe to services from multiple CSPs to meet the resource requirements. In this case, a federated identity management approach is required. The CSCs can use the identity tokens generated by the IdPs and these cloud users can exchange such tokens with various CSPs in the federation [12].

Analysis and results of DAC architecture reveal that using such an architecture is important in the domain of distributed applications or service computing. However, this model has some limitations. In particular, there is no effective mechanism which meets all access control requirements.

B. Adaptive access algorithm

Wenhui *et al*. [13] added trust management to the Role-Based Access Control (RBAC) in order to propose an adaptive access algorithm for cloud environments. This model is based on loyalty, *i. e.*, a user is restricted only when its behavior contains malicious behavior. More specifically, the user request is first analyzed, and based on trust evaluation, the user becomes dynamically authorized. Here, user's trust is calculated according to user's behavior. In other words, the user access to the resource is dynamically based on calculation. As a result, by establishing dynamic mapping between roles and trust values, this model is able to determine the security level and control the user's access to the resources.

The trust-role-based-access control model claims that it can efficiently control user's malicious behavior. However, this model depends on the trust values, as the trust evaluation process needs to be improved in order to become widely used.

C. Multi-tenancy access control model

Multi-Tenancy Access Control Model (MTACM) is a security architecture which embeds the security duty separation principle in multi-tenancy cloud environments [14]. The main idea of MTACM is based on limiting the management privilege of CSP and letting the customers manage the security of their own business. In this model, the duty separation mechanism between cloud service provider and cloud customer is handled by a management module. However, the management module is not user-friendly for customers, as the cloud customer has to take care of the data security.

D. Role-based multi-tenancy access control

Role-Based Multi-Tenancy Access Control (RB-MTAC) applies identity management to determine user's identity and applicable roles [15]. Such a model combines two important concepts in access control under multi-tenancy access environment: identity management and role-based access control. In this context, Yang *et al*. [15] believe that this combination makes it easier to manage privileges that protect the security of application systems and data privacy. Providing a set of privileges and identity management schemes for corporations in cloud computing environment is the main contribution of this security model.

This scheme can be used to easily change employee privileges when a personnel member leaves an organization or when we want to grant employees more access without the need to modify all employee privileges one by one. However, RB-MTAC is not independent, and for implementing it in a cloud computing system, a directory service is needed.

E. CloudPolice

Popa *et al*. [16] proposed *CloudPolice*, a system that implements a hypervisor-based access control mechanism for multi-tenancy cloud environments. CloudPolice operations are illustrated in Figure 3. More specifically, when a source VM initiates a new flow, the source hypervisor sends a control packet to the destination hypervisor. This control packet specifies the security group to which the source VM belongs (Step 1). As soon as the control packet reaches the source hypervisor, it will be checked by the destination hypervisor to verify the policy for the group of the destination VM (Step 2). If the policy allows the traffic, then the state of the traffic will be created for this flow by the destination hypervisor. However, if the traffic is not allowed or should be rate-limited, the control packet will be sent back to the source hypervisor to block or rate limit the flow or the VM (Step 3).

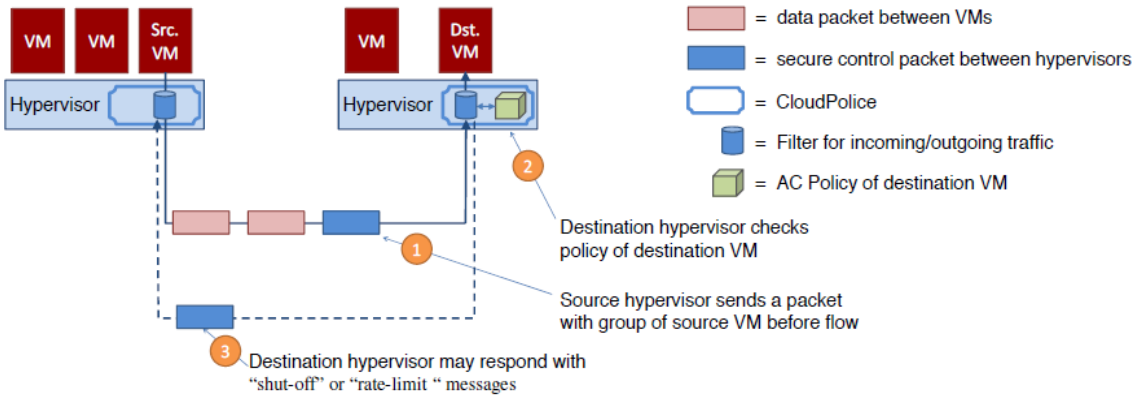


Figure 3. Overview of CloudPolice operations [16].

Since hypervisors are generally trusted, network-independent, close to VMs and fully software programmable, CloudPolice seems to be effective to prevent denial of service (DoS) attacks from malicious agents who send unauthorized traffic to their targets. As a result, CloudPolice acts as stateful firewalls and creates a state for each flow.

However, there are several major concerns for the feasibility of CloudPolice. The first concern is the ability for the hypervisor to act on per flow state, as the hypervisor should be ready to act on every single flow. The second concern is the ability to install new state with low enough latencies for new traffic flows, as we should make sure that the hypervisor is able to create a state for each new incoming flow very fast. As a result, the hypervisor should be able to create states for all new flows without latency (or at least with acceptable latency) and also act on the states that already exist in the buffer. Also, CloudPolice imposes overheads in the system, as the destination hypervisor receives all the traffic and decides to pass or drop the traffic based on the security attributes of the target virtual machines.

IV. THE PROPOSED ARCHITECTURE

This section defines the main assumptions, as well as the design and principles of the proposed architecture.

A. Main assumptions

The proposed architecture deals with the concept of Inter-VM traffic, which is the transmission of any data packet to and from one virtual machine. In other words, when the hypervisor encounters inter-VM traffic, the traffic does not pass through the physical switch or router, as the virtual switch that is located at the hypervisor forwards the packet to the destination VM. At this point, the following assumptions need to be done:

- The virtual machines and physical servers are co-located at the same cloud provider. If the entire system is not part of the Cloud, then for sending traffic to another Cloud, the traffic should pass through a real router or firewall. In this case, the policies that are implemented in the firewall should be enforced.
- Each physical server has only one hypervisor. In this case, the security attributes and access control lists of all virtual machines that belong to a physical server are located at one hypervisor. If we have multiple hypervisors on a physical server, we should apply an extra process for realizing which hypervisor contains the access control lists of certain virtual machines.
- Each physical server is hosting at least one tenant, and each tenant has at least one virtual machine. Since each virtual machine should be registered as a tenant, if a tenant is registered in the Cloud, a virtual machine should be assigned to that tenant.
- All access control lists are defined and stored in the hypervisor.
- In its startup process, a hypervisor sends an update message to the other hypervisors that are located at the same Cloud. This update message contains the IP address and the ID of virtual machines that are located at that hypervisor.

B. Architecture principles

The principles of the proposed architecture are based on control packets, which is the core element for verifying security permissions of virtual machines in multi-tenancy cloud environments. In the following, we explain the main elements of the proposed access control architecture, which is illustrated in Figure 4:

- Source (Src.) VM is a virtual machine that is installed on the source hypervisor, as the latter is located at the physical source server. The source VM is then sending traffic packets to a virtual machine in the same Cloud called Dst. VM.
- Destination (Dst.) VM is installed at the destination hypervisor, and this hypervisor is located at the destination physical server.
- A data packet is a packet that the source VM wants to send to the destination VM.

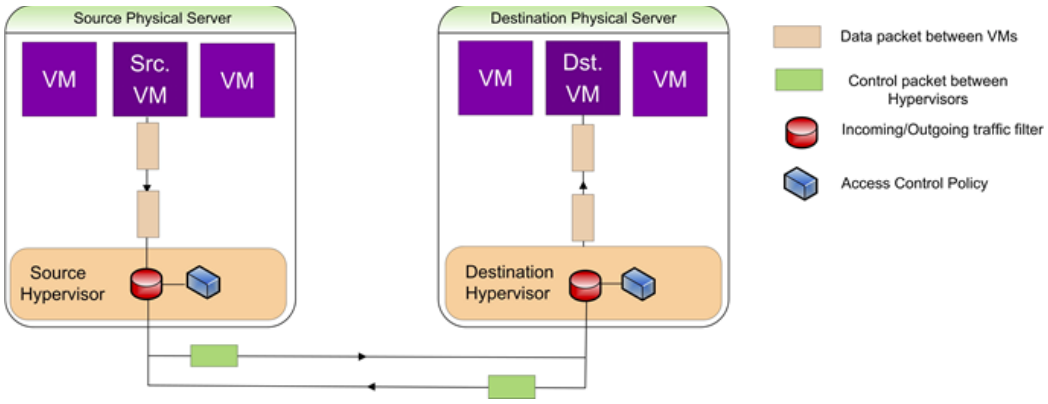


Figure 4. Principles of the proposed architecture.

- A control packet is a special packet that is generated by the source hypervisor. Its content represents the specifications of the source and destination VMs.
- Incoming/outgoing traffic filter is a lightweight IDS that is integrated in the hypervisor. It compares the control packet with the access control lists of destination VM.
- An access control list is a set of security permission that defines the level of security of each virtual machine.

C. Architecture design

The main goal of the proposed architecture is to block and drop undesired packets as close as possible of the source hypervisor. As illustrated in Figure 4, when the source VM sends traffic to the destination VM, such traffic has to pass through the source hypervisor. As soon as a data packet reaches the hypervisor, it generates a control packet which consists of the necessary information for access control checking, such as the source IP address, the destination IP address, the port numbers, as well as the protocol type. Such a control packet has to be sent to the destination hypervisor which checks its content and decides whether the traffic can be delivered to the destination hypervisor. If the source VM is permitted to send the so-called traffic to the destination VM, the destination hypervisor adds a pass or drop value to the control packet payload, and sends it back to the source hypervisor. According to this value, the source hypervisor threats the awaiting traffic.

As illustrated in Figure 5, the process starts when a VM initiates to send some traffic to another VM. As soon as such traffic is received by the source hypervisor, it checks the packet and looks for the destination address that is located at the inserted IP packet header. If the destination address belongs to a virtual machine in the same cloud, we will have two possibilities. The first case considers that the destination address is located at the same physical server. In this case, the architecture checks the access control policy of the destination VM, and can decide whether to pass or drop the traffic. The second case occurs when the destination address is located at a different physical server. In this case, the source hypervisor generates and sends the control packet to the destination hypervisor. Then, it waits for the response control packet.

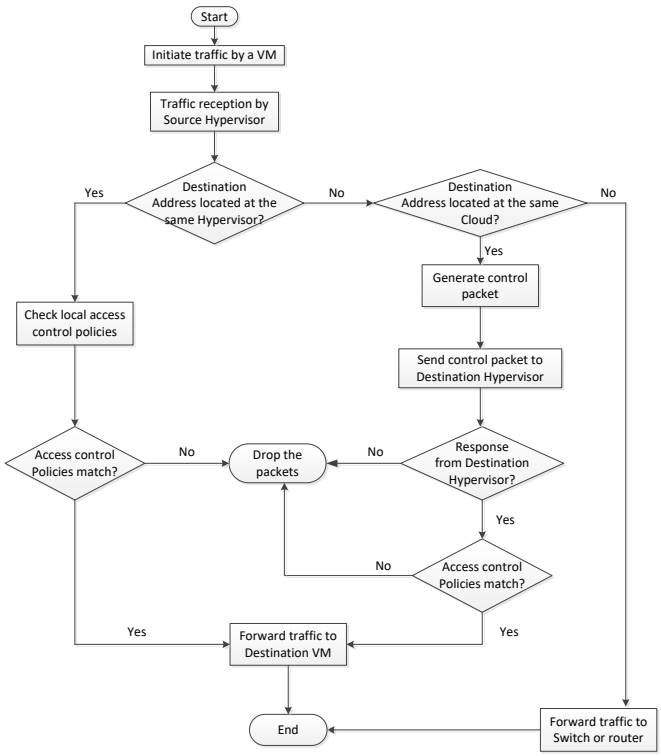


Figure 5. General mechanism flowchart.

Beside such possibilities, there may be an exception, when the destination address does not belong to any VM in this Cloud, which means that the source and destination addresses belong to two devices that are not co-located at the same Cloud. In this case, the architecture only has to pass the traffic to the default gateway of the source hypervisor (router, switch or firewall).

The main part of the mechanism starts if the destination address belongs to a VM that is located at a destination hypervisor. In this case, the whole traffic should wait until the source hypervisor generates and sends a control packet to the destination hypervisor. Hence, the decision will be made based on the response control packet. Figure 6 shows the main tasks of the destination hypervisor when it receives the control packet from the source hypervisor. More precisely, the destination hypervisor selects one of the following actions:

- Insert a pass value to the control packet if the access control policy of the destination VM matches, and accept the traffic from the source VM.
- Insert a drop value to the control packet if the access control policy of the destination VM does not match, as the source VM is not authorized to send the traffic to the destination VM.
- Insert a null value to the control packet if the destination address is not found in the destination hypervisor. This may happen if the control packet is sent to the hypervisor by mistake, or if the VM destination is migrated to another hypervisor, whereas the source hypervisor is not informed about such migration.

After inserting the proper value to the control packet, the destination hypervisor returns the edited control packet to the source hypervisor. The response control packet contains the decision and the action to be taken for the traffic. In the case of a drop value, the source hypervisor drops the traffic right away, as such traffic will not even exit the hypervisor, which means no wasted and unnecessary traffic in the network. Consequently, the network bandwidth does not suffer from extra and unwanted traffic. Finally, the pass value indicates that the access control policy matches between the source and destination, whereas the source VM and the traffic will pass throughout the destination hypervisor.

V. A USE CASE SCENARIO

In this section, we analyze a use case scenario which enables to tackle the problem of sending unauthorized traffic to a VM in the context of multi-tenancy cloud environments. This scenario is illustrated in Figure 7, where a public Cloud is connected to the Internet, using a router and three physical servers that are connected to a layer-2 switch. In this scenario, the function of the router is to route the internal cloud traffic to the Internet. Apparently, the router serves as a controller, enabling the networked devices to talk to each other efficiently.

In this scenario, there are 3 physical servers, as well as 10 virtual machines. These virtual machines belong to 4 tenants. The multi-tenancy topology of this Cloud is as follows:

- Server 1: Tenant 1 (VM1, VM2) and Tenant 2 (VM3);
- Server 2: Tenant 1 (VM4, VM5) and Tenant 3 (VM6, VM7);
- Server 3: Tenant 4 (VM8) and Tenant 3 (VM9, VM10).

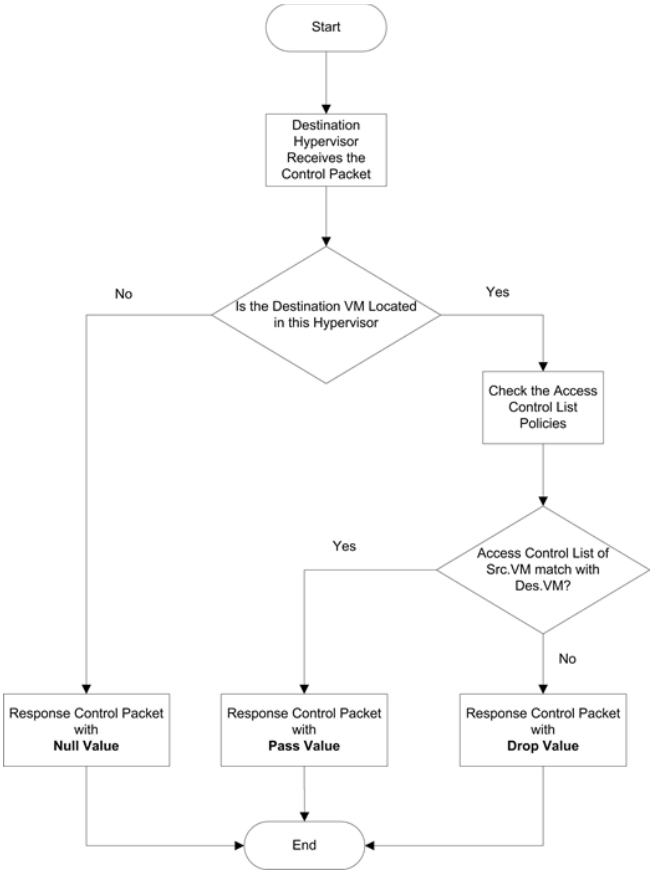


Figure 6. Destination hypervisor's tasks after control packet reception.

It is important to mention that the process of controlling the access is executed in the hypervisors. In this context, the scenario has two phases: the first phase consists of generating control packets, whereas in the second phase, the destination hypervisor investigates the information and decides about the destiny of the packet. More specifically, in phase one of the scenario, the VM Source sends a traffic flow to the hypervisor source, as illustrated in stage 1 of Figure 8. Then, the source hypervisor generates a control packet. The content of this control packet is based on the traffic to be sent from source VM3 to destination VM8.

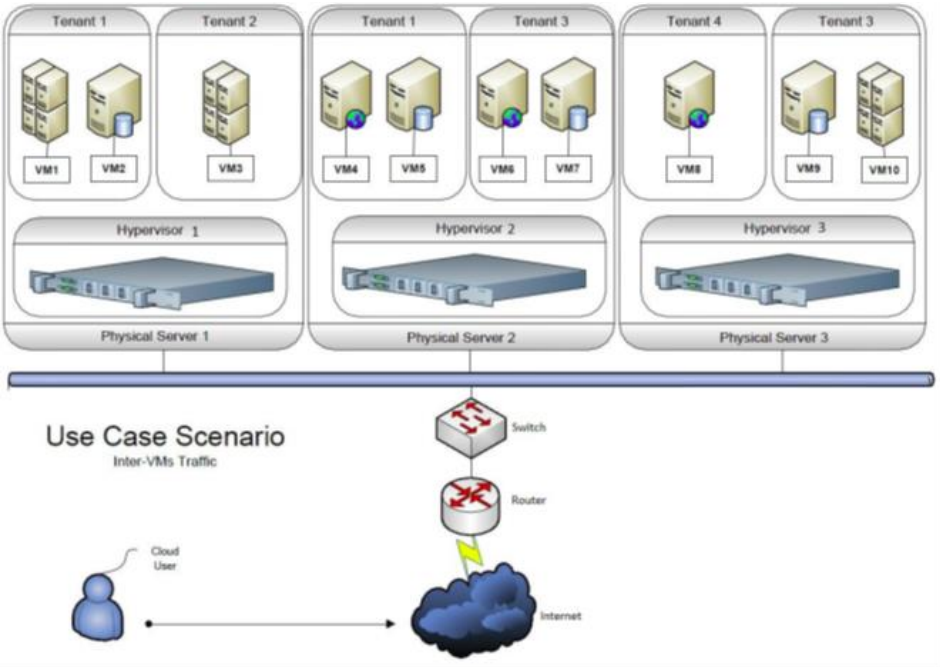


Figure 7. A use case scenario for multi-tenancy cloud access control.

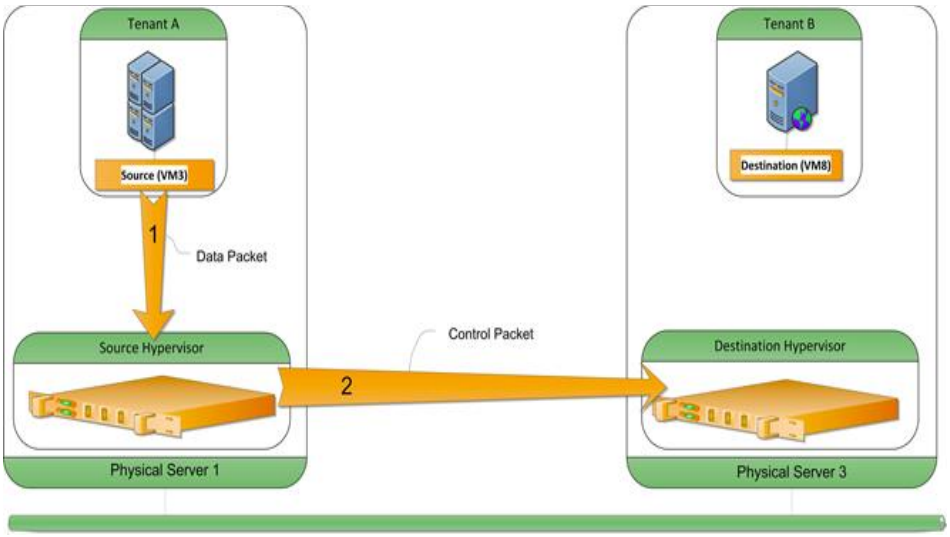


Figure 8. Illustration of phase one of the scenario.

As illustrated in stage 2 of Figure 8, the source hypervisor sends the control packet to the destination hypervisor in order to check the access control policy of the VM destination.

In phase two, the control packet arrives at the destination hypervisor which checks the access control lists (ACLs) to verify if VM3 is authorized to send traffic to VM8. If the ACLs related to VM8 match, the destination hypervisor sends back a pass value within the control packet (called response control packet) to the source hypervisor, as illustrated in stage 3 of Figure 9. The response control packet enables the hypervisor source to decide what to do with the traffic that is waiting in the source hypervisor. Hence, if the security attributes of VM8 do not match the data packet, then the destination hypervisor sends a drop signal to the source hypervisor.

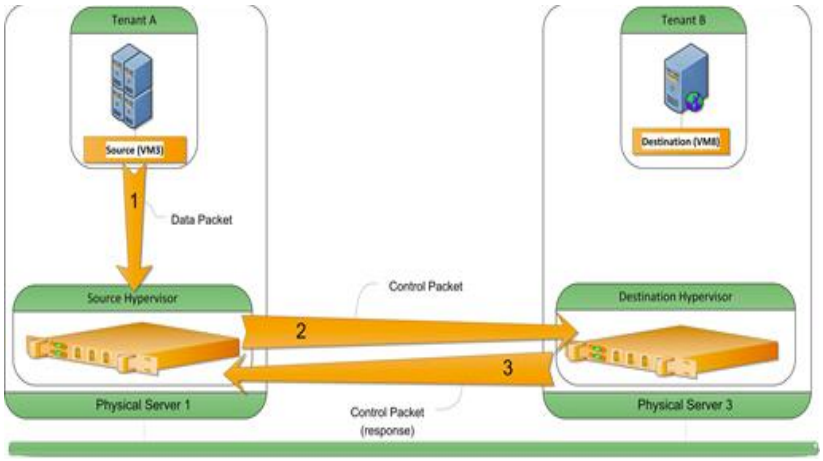


Figure 9. Illustration of phase two of the scenario.

VI. PERFORMANCE ANALYSIS

In this section, we evaluate the performance of the proposed access control architecture that we call *CloudGuard*. More specifically, we will define the parameters that will enable to mathematically express the latency and system throughput. Then, we will present numerical results and analysis.

A. Parameter definition

Let us first define the most important parameters that characterize the proposed access control architecture:

- N_p: the number of packets transmitted from source VM to destination VM;
- S_p: the data size of each packet (bits);
- N_f: the number of traffic flows that are being transmitted from source VM to destination VM;
- S_f: the size of traffic flow (bits);
- T: the time taken by the traffic flow to be sent from source VM to destination VM (seconds);
- RTT: the time taken by a control packet to be sent from a source hypervisor to a destination hypervisor, and received back by the source hypervisor (seconds). This parameter also refers to as *Round Trip Time*;
- BW: The amount of data that can be carried from one point to another point in a given period in a Gigabit Ethernet (1 073 741 824 bits per second);
- P_i: the probability of having intrusion traffic.

B. Packet latency

We may now evaluate the latency of the proposed access control architecture and compare it with that of CloudPolice [16] that we studied Section III. While other different solutions may be found in the literature for comparison with the proposed architecture, CloudPolice is a hypervisor-based access control system that implements, like CloudGuard, the access control only within the hypervisor. In other words, both architectures are designed to enforce the access control list in the hypervisor.

However, a major difference between such architectures is that with CloudGuard, the source hypervisor waits for the control packet response from the destination hypervisor. In this case, if the security policies of source VM and destination VM match, the source hypervisor passes the traffic toward the destination hypervisor. As a result, the destination hypervisor passes the traffic to destination VM. On the other hand, according to CloudPolice, the source hypervisor does not wait for the control packet response, as it sends the whole traffic to the destination hypervisor. When the traffic reaches the destination hypervisor, the security policies will be checked between the traffic and destination VM. In this case, if the traffic is authorized, it will continue its journey to destination VM. However, if the traffic does not match with the security attributes of destination VM, it will be ignored and dropped in the destination hypervisor.

From the differences between CloudGuard and CloudPolice, we can generate two expressions for evaluating the latency related to each packet: one for the latency obtained with CloudGuard, and the other one for the latency obtained with CloudPolice. With CloudGuard, the control packet is generated by the source hypervisor, whereas the control packet response is generated by the destination hypervisor. When one VM wants to communicate with another VM in the same cloud environment, the source VM first sends the traffic to the source hypervisor. Then, the source hypervisor checks the destination address, and finds out the destination hypervisor on which the destination VM is hosted on.

In the next step, the source hypervisor generates a control packet, and sends such control packet to the destination hypervisor. Then, it keeps the traffic, and does not let the traffic to be passed to the destination hypervisor, unless it receives back the control packet response from the destination hypervisor. In this stage, the responsibility of the destination hypervisor is to check the traffic security attributes with the security policies of destination VM. If the traffic is authorized,

the control packet response asks the source hypervisor to pass the traffic.

As a result, for each traffic flow, one control packet needs to be generated with CloudGuard. Based on that, to evaluate the average packet latency, we need to multiply the number of traffic flows N_f with round trip time *RTT*. More specifically, to derive an expression for the latency, RTT is first added to the time it takes to transmit one traffic flow, while only considering non intrusion traffic. Then, we add this value to the round trip time of the control packet for intrusion traffic. Such process is repeated for each traffic flow in order to obtain the packet latency. Therefore, the average packet latency obtained with CloudGuard is expressed as follows:

$$D_{CG} = \left(RTT + \frac{S_f}{BW}\right)(1 - P_i) + RTT \times N_f \times P_i \qquad (1)$$

where *RTT*, S_f, *BW*, P_i and N_f are defined in subsection A.

Let us now evaluate the average packet latency while considering CloudPolice. With such access control architecture, when a VM needs to communicate with another VM in the same cloud environment, the source VM sends the traffic to the source hypervisor. The source hypervisor checks the destination VM address and passes the traffic to the proper destination hypervisor where destination VM is hosted. Then, the destination hypervisor checks the traffic security attributes with the security policies of the destination VM. If the traffic is not authorized, the destination hypervisor generates a control packet, and sends it to the source hypervisor in order to inform the source hypervisor to block the next stream of the same traffic, or to limit the bandwidth that must be allocated to this traffic. Therefore, CloudPolice generates control packets only for intrusion traffic flows. As a result, the number of control packets that are generated depends on the probability of intrusion traffic.

More specifically, for evaluating the latency obtained with CloudPolice, we first need to multiply the number of intrusion traffic flows with RTT. Then, we need to evaluate the time it takes for intrusion traffic to pass to the network since the amount of intrusion traffic may technically consume the available network bandwidth. Based on that, we first calculate the round trip time RTT of all control packets. Then, RTT is added to the time that it takes to pass the intrusion traffic from source to destination. As a result, the average packet latency obtained with CloudPolice is expressed as follows:

$$D_{CP} = \left(RTT \times P_i + \frac{S_f}{BW}(1 - P_i)\right) \times N_f \qquad (2)$$

where *RTT*, P_i, S_f, *BW* and N_f are defined in subsection A.

C. System throughput

In order to evaluate the proposed access control architecture, the system throughput is also taken into consideration. In this context, we develop two expressions for the system throughput: one expression for the system throughput with CloudGuard and another one for the system throughput with CloudPolice. For evaluating the first expression, we consider that CloudGuard generates a control packet for each traffic flow before passing such traffic to the destination hypervisor. The traffic remains in the source hypervisor until the source hypervisor receives back the control packet response. In this case, only authorized traffic flows can pass through the network.

As a result, for evaluating the system throughput with CloudGuard, we first need to divide the number of traffic flows by the average latency. Then, we multiply the obtained results with the probability of not having intrusive traffic. The system throughput with CloudGuard is expressed as follows:

$$R_{CG} = \frac{S_f N_f}{D_{CG}}(1 - P_i) \qquad (3)$$

where S_f, N_f, and P_i are defined in subsection A, whereas D_{CG} is given by (1).

Moreover, with CloudPolice, no control packet is sent before passing traffic packets to the destination hypervisor. In other words, all traffic flows pass from the source hypervisor to the destination hypervisor. As a result, such traffic flows include unwanted and unauthorized traffic. In this context, when evaluating the system throughput with CloudPolice, we should deduct from such traffic flows the amount of unauthorized traffic, since such unauthorized traffic that is generated by an intruder may occupy the available network bandwidth. Then, the number of traffic flows is divided by the average latency obtained in (2), as we multiply such a result with the amount of intrusion traffic. The obtained expression is valid for the situations where the probability of intrusion traffic is between 0 and 0.5. Further, because of the nature of CloudPolice, we consider the system throughput equal to zero if the probability of intrusion is between 0.5 and 1.

As a result, the system throughput with CloudPolice is expressed as follows:

$$R_{CP} = \begin{cases} \frac{S_f N_f}{D_{CP}}(1 - 2P_i), if\ 0 \le P_i \le 0.5 \\ 0, \qquad\qquad if \quad 0.5 < P_i \le 1 \end{cases} \qquad (4)$$

where S_f, N_f, and P_i are defined in subsection A, whereas D_{CP} is given by (2).

D. Numerical results and analysis

For numerical results, we will compare the system throughput obtained with CloudGuard with that obtained with CloudPolice. For such comparison, MATLAB Release 2021b (R2021b) [17] will be used as a fundamental research tool that can mathematically computes the system throughput for both architectures according to several scenarios. In this context, a number of parameters must be calculated.

First, it is important to evaluate the size of traffic flows S_f. Based on [18], S_f may be expressed as follows:

$$S_f = S_p \times N_p \qquad (5)$$

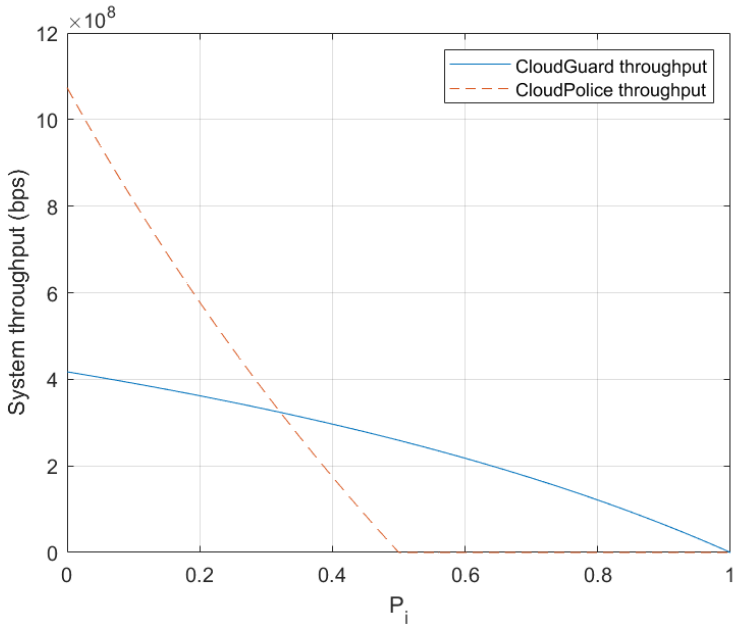


Figure 10. System throughput comparison (CloudPolice vs CloudGuard).

where S_p represents the packet size, and N_p represents the number of packets transmitted from source VM to destination VM. We then consider packet sizes of 128 bytes (*i.e.*, 1 024 bits), and we assume that each traffic flow contains 1 000 packets. As a result, the flow size considered for numerical results will be set to 1 024 000 bits.

Another important parameter for numerical results is the number of traffic flows N_f that are sent from source VM to destination VM. In order to verify how CloudGuard and CloudPolice performs in different situations, we consider N_f as random numbers from 1 000 to 10 000, and we choose the same N_f value for system throughput comparison.

Moreover, in [16], *RTT* is set to 1.5 *ms* with CloudPolice. Since the process of generating control packets with CloudPolice is similar to that of CloudGuard, we assume that RTT is the same for both access architectures, *i.e.*, 1.5 *ms*. Further, as intrusion traffic is unpredictable in a multi-tenancy cloud network, the probability of intrusion traffic P_i will be chosen between 0 and 1 for numerical results.

Figure 10 illustrates the system throughput comparison (CloudPolice vs CloudGuard) in function of P_i. We realize that, if the probability of intrusion is low (P_i < 0.3215), CloudPolice offers better system throughput than CloudGuard. Moreover, for P_i = 0.3215, both architectures have the same throughput, whereas when the amount of intrusion traffic is higher in the cloud system (P_i > 0.3215), results show that CloudGuard offers better system throughput than CloudPolice.

Also, it is important to mention that a lot of intrusion traffic (P_i > 0. 5) completely paralyzes the cloud system with CloudPolice, since the system is unable to offer any throughput for P_i > 0. 5. Such results prove that CloudGuard is more appropriate than CloudPolice in high-risk multi-tenancy cloud computing environments. In other words, CloudGuard is an effective security architecture which is suitable for high-risk environments. A high-risk multi-tenancy cloud computing environment refers to a cloud environment in which the system is the target of many attacks, as lots of intrusion attempts and unwanted traffic are being sent to the VMs. Financial institutions and game centers are examples of networks that are always targeted by attackers and malicious agents. On the other hand, one can hardly imagine a network environment with zero intrusion traffic, which means that P_i = 0.

VII. Conclusion

The access control architecture (called CloudGuard) proposed in this paper for multi-tenancy cloud environments satisfies a number of requirements, such as scalability and security. This architecture is scalable in the sense that, if the number of VMs grows, we only need to implement this architecture in the hypervisor of each physical server without any extra changes in the system. Besides that, the architecture enables to maintain the security of information in the cloud system by controlling the traffic sent from one hypervisor to another hypervisor and enforcing the security policies in the hypervisor.

Using MATLAB R2021b, we built a mathematical model in order to evaluate CloudGuard performance. More specifically, we compared the system throughput obtained with CloudGuard with that obtained with CloudPolice. Numerical results prove that CloudGuard obviously offers better system throughput in critical or high-risk multi-tenancy cloud network environments, where the amount of intrusion traffic is high. As a result, CloudGuard leads to better

performance by avoiding unnecessary traffic and dedicating the cloud resources to necessary traffic.

Future works will focus on intra-cloud traffic. In real world cloud environment, the traffic may pass through lots of intermediate devices, such as switches, routers and firewalls. In this context, when a VM needs to send traffic to another VM that is located at another Cloud, we need more complex access control updates between hypervisors that are located at different cloud environments. Hence, in the context of intra-cloud traffic, the process of propagation of security attributes and updates between hypervisors should be taken into consideration. Future works will also focus on implementing a prototype of the proposed architecture on a real cloud environment.

REFERENCES

[1] R. Beaubrun and A. Quintero, “A Secure Access Control Architecture for Multi-Tenancy Cloud Environments,” CLOUD COMPUTING 2021: The Twelfth International Conference on Cloud Computing, GRIDs, and Virtualization, 18-22 Apr. 2021, pp. 48-53.

[2] K. Hashizume, D. G. Rosado, E. Fernández-Medina, and E. B. Fernandez, "An analysis of security issues for cloud computing," Journal of Internet Services and Applications 4:5, vol. 4, pp. 5-18, 2013.

[3] M. Auxilia and K. Raja, “Dynamic Access Control Model for Cloud Computing,” Sixth International Conference on Advanced Computing (lCoAC), 17-19 Dec. 2014, pp. 47-56.

[4] Z. Minqi, Z. Rong, X. Wei, Q. Weining, and Z. Aoying, "Security and Privacy in Cloud Computing: A Survey," in 6th International Conference on Semantics Knowledge and Grid (SKG), Beijing, 1-3 Nov. 2010, pp. 105-112.

[5] C. J. Guo, W. Sun, Y. Huang, Z. H. Wang, and B. Gao, "A framework for native multitenancy application development and management," in 9th International Conference on E-Commerce Technology/4th International Conference on Enterprise Computing, Ecommerce and E-Services, Tokyo, 23-26 July 2007, pp. 551-558.

[6] G. Kappes, A. Hatzieleftheriou, and S. V. Anastasiadis., “Multitenant Access Control for Cloud-Aware Distributed Filesystems,” IEEE Transactions on Dependable and Secure Computing, Vol. 16, No. 6, pp. 1070-1085, 2019.

[7] S. Harris, CISSP All-in-One Exam Guide, Sixth ed. New York: McGraw-Hill, 2013.

[8] K. Albulayhi, A. Abuhussein, F. Alsubaei, and F.T. Sheldon, “Fine-Grained Access Control in the Era of Cloud Computing: An Analytical Review,” 10th Annual Computing and Communication Workshop and Conference (CCWC), 6-8 Jan. 2020, pp. 748–755.

[9] K. Benzidane, S. Khoudali, and A. Sekkaki, "Autonomous Agent-based Inspection for inter-VM Traffic in a Cloud Environment," in 7th International Conference for Internet Technology and Secured Transactions (ICITST-2012), London, 10-12 December 2012, pp. 656-661.

[10] S. J. De and S. Ruj, “Efficient Decentralized Attribute Based Access Control for Mobile Clouds,” IEEE Transactions on Cloud Computing, Vol. 8, No. 1, pp. 124-137, 2020.

[11] A. Jasti, P. Shah, R. Nagaraj, and R. Pendse, "Security in Multi-Tenancy Cloud," in International Carnahan Conference on Security Technology (ICCST), San Jose, CA, 5-8 October, 2010, pp. 35-41.

[12] M. V. Thomas and K. C. Sekaran, "An Access Control Model for Cloud Computing Environments," in 2nd International Conference on Advanced Computing, Networking and Security (ADCONS), Mangalore, 15-17 Dec 2013, pp. 226-231.

[13] W. Wenhui, H. Jing, S. Meina, and W. Xiaohui, "The design of a trust and role based access control model in cloud computing," in 6th International Conference on Pervasive Computing and Applications (ICPCA), Port Elizabeth, 26-28 Oct. 2011, pp. 330-334.

[14] X.-Y. Li, Y. Shi, Y. Guo, and W. Ma, "Multi-Tenancy Based Access Control in Cloud," in International Conference on Computational Intelligence and Software Engineering (CiSE), Wuhan, 10–12 Dec. 2010, pp. 1-4.

[15] S.-J. Yang, P.-C. Lai, and J. Lin, "Design Role-Based Multi-tenancy Access Control Scheme for Cloud Services," in International Symposium on Biometrics and Security Technologies (ISBAST), Chengdu, 2–5 Jul 2013, pp. 273-279.

[16] L. Popa, M. Yu, S. Y. Ko, S. Ratnasamy, and I. Stoica, "CloudPolice: Taking Access Control out of the Network," in ACM Workshop on Hot Topics in Networks. HotNets, Monterey, CA, USA, 20-21 Oct. 2010, pp. 1-6.

[17] Available from https://www.mathworks.com/

[18] Internet Engineering Task Force (IETF), “IPv6 Flow Label Specification”, in Request for Comments (RFC): 6437, ISSN: 2070-1721, 2011.