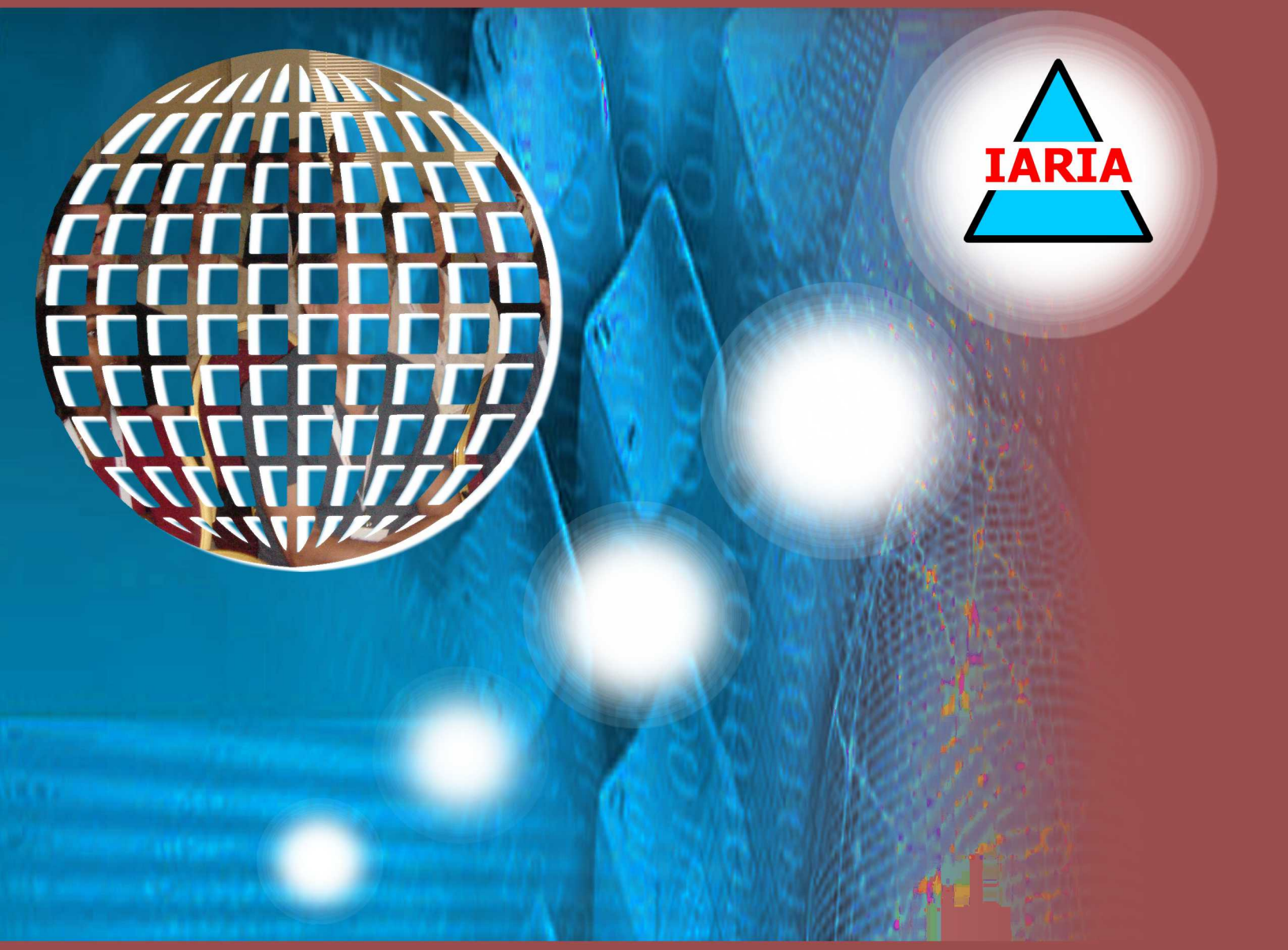
International Journal on Advances in Security



2009 vol. 2 nr. 4

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 2, Number 4, 2009

- Aljosa Pasic, ATOS Origin, Spain
- Vladimir Stantchev, Berlin Institute of Technology, Germany
- Michiaki Tatsubori, IBM Research - Tokyo Research Laboratory, Japan
- Ian Troxel, SEAKR Engineering, Inc., USA
- Hans P. Zima, Jet Propulsion Laboratory/California Institute of Technology - Pasadena, USA // University of Vienna, Austria

Security in Internet

- Evangelos Kranakis, Carleton University, Canada
- Clement Leung, Victoria University - Melbourne, Australia
- Sjouke Mauw, University of Luxembourg, Luxembourg
- Yong Man Ro, Information and Communication University - Daejon, South Korea

International Journal on Advances in Security
Volume 2, Number 4, 2009

CONTENTS

Self-organization supported algorithms for wireless sensor networks

Jian Zhong
School of Computer Science and Information Technology
Royal Melbourne Institute of Technology
Melbourne, Australia
E-mail: jian.zhong@rmit.edu.au

Peter Bertok
School of Computer Science and Information Technology
Royal Melbourne Institute of Technology
Melbourne, Australia
E-mail: peter.bertok@rmit.edu.au

***Abstract*—*Self-organization is an important issue in wireless sensor networks because of the inherent unreliability of the network. Besides, variable threats in the networks can not be ignored. Extending battery life and enhancing robustness under variable threats are two essential aspects which need to be considered when a self-organization scheme is explored. In order to address these issues, a Redundant Nodes Selection scheme and a variable Threats Probability Estimation scheme are proposed in this paper. RNS is able to select redundant nodes that can be switched off without affecting overall sensing coverage. TPE is able to help a sensor node to choose the most suitable path and avoid high-threat neighbors in order to reduce packet loss. The scenario with RNS extended battery life by 30% to 50%, and postponed the occurrence of first partitioning in the network by 27% to 140%. TPE decreased packet loss by 225% to 400% when a high threat level was involved.*

Keywords-wireless sensor networks; self-organization; variable threats; battery life; robustness

I. INTRODUCTION

For the constraint of wireless sensor networks (WSNs), some threats can not be ignored, such as environment changes, sensor damage, information lost and sensor attacks etc. There are some key areas which need to be explored, such as [2] network organization, routing, security, node localization, clock synchronization, power management and key management etc. This focuses on network self-organization.

For wireless sensor networks, organizing typically begins with neighbor discovery [3]. Nodes send rounds of messages (packets), build local neighbor tables and organize clusters centered around a cluster head. The tables include information on each neighbor's ID and location. However, during operation some sensors become inactive due to battery exhaustion which may result in network partitioning, and packets can be lost due to various threats. Extending battery life, postponing the occurrence of first partitioning and reducing packet loss are significant aspects of self-organizing.

A self-organization scheme supported by a redundant nodes selection algorithm (RNS) and variable threats probability estimation (TPE) is proposed here to extend battery life and reduce packet loss. RNS is designed to scan all sensor nodes and select redundant nodes that can be switched off so that the whole area will still be covered. The redundant nodes will be used as backups and replacements to extend the effective network lifetime without any coverage loss.

The second method TPE, improves the scheme originally proposed in [7], by allowing nodes to choose a more reliable neighbor as a default path to the data sink and blocking high-threat nodes to reduce packet loss. The whole proposed scheme provides a solution for WSNs to extend battery life and avoid variable threats.

This paper is based on the conference article "A Variable Threats Based Self-Organization Scheme for Wireless Sensor Networks", presented at the SENSORCOMM 2009 conference.

II. BACKGROUND

Due to the physical constraints of wireless sensor networks, sensors organization, resilience to node capture attack and power-saving are essential aspects. This chapter discusses the work done by some of the researchers on wireless sensor network self-organization, coverage exploration, static attack probability and related aspects. In the first section, deployment and topologies will be presented. Then the self-organization issue will be discussed. Furthermore, previous works related to power saving and resilience will be detailed. In the end of this chapter, a summary of the essential literature is given.

A. Topologies and Network Architecture

For a wireless sensor network, the topology and network architecture always need to be considered first. In the literature, there are some wireless sensor network topologies and architectures proposed for the uniform and non-uniform deployment. The most common topologies and architectures are described by Bhaskar Krishnamachari [13], which are shown in Fig. 1.

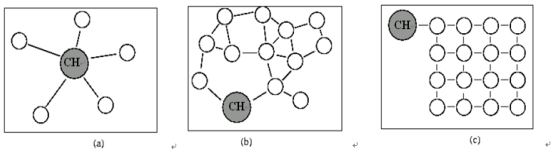


Figure 1. Different topologies in wireless sensor networks [13]

Fig. 1(a) shows the simplest topology, in which all sensors directly report the collected data to the cluster head

(CH). Fig. 1(b) shows a tree topology and the collected data is sent to the data sink via different paths depending on some factors, such as power-save path, most-secure path, reputation-based path etc. Fig. 1(c) shows a grid topology which is also used in an experimental model [7]. A more complex scenario is depicted in [13] with two-tiered architecture, as shown in Fig. 2.

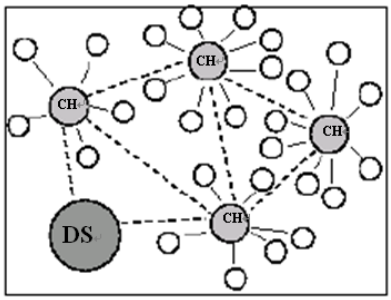


Figure 2. Topology in WSNs [13]

The biggest circle denotes the data sink (DS), which gathers reports from the cluster heads (CH). The small white circles indicate wireless sensors, which can collect data from their sensing ranges. In this typical topology, sensor nodes can directly connect to a cluster head which acts as a group leader.

In our research, we consider a clustered organization, when individual nodes are connected to a cluster head, and data from the cluster is relayed by the head towards the destination. However, in a number of cases there is no guarantee that all cluster heads can directly connect to the data sink or all sensor nodes can directly connect to a cluster head. For such cases, a random non-uniform architecture has been mentioned in [9], which is shown in Fig. 3.

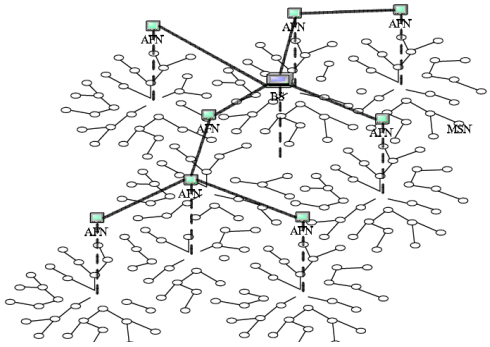


Figure 3. A two-tiered architecture [9]

In Fig. 3 [9], a small number of high-end nodes, called *Aggregation and Forwarding Nodes* (AFNs), are deployed together with numerous low-end sensor nodes, called *Micro Sensor Nodes* (MSNs). In addition, the network includes a globally trusted *base station* (BS), which is the ultimate destination for data streams from all the AFNs. The BS has powerful data processing capabilities, and is directly connected to an outside network. Each AFN is equipped with a high-end embedded processor, and is capable of communicating with other AFNs over long distances.

The deployment and topology in [9] are more reasonable for random scattering, and foremost, this architecture can adjust to the changes in topology during runtime, i.e. new nodes can be added into the network or some working nodes can be compromised.

Accordingly, a two-tiered wireless sensor network model will be used in our proposed method, and AFN will be called Cluster Head (CH), BS will be called Data Sink (DS) and MSN will be called Sensor Node (SN) in the rest of the paper.

B. Threats and Threat Model

There are many kinds of threats for WSNs, such as mentioned in [20]. In this paper, node failure will be discussed, including node capture, physical damage, battery exhaustion and any condition making the sensors unavailable. My proposed attack model does not include the scenario in which adversaries not only steal the data stored in the sensor nodes but also put the captured sensor nodes back into the WSNs as agents for collecting messages.

C. Self-Organization

After a sensor network has been deployed, self-organization comes. It will be separated into four different aspects: clustering and neighboring, power consumption, resilience, and sensor addition.

1) Clustering and Neighboring

To organize wireless sensors, Falko Dressler et al. [14] described a solution which involved a mobile robot that helps to organize the network. The model is shown in Fig. 4.

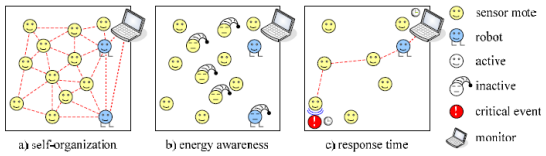


Figure 4. Challenges in the sensor networks [14]

Fig. 4(a), Self-organization, shows that the active sensor nodes can transfer their collected data to the robot via different paths and the monitor merely need to receive data from two robots. Fig. 4(b) shows that, for energy awareness, some sensors are switched to an inactive mode. A critical event is still can be gathered and transferred to the monitor, which is illustrated in Fig. 4(c). The method relies on mobile robots to maintain the whole network. However, in many cases, a mobile robot is not available.

Another self-organization mechanism for both uniform and non-uniform was described. The location-based mechanism [3] is relying on a special node named “server node”. This will raise the cost of the whole network and if this server node is compromised or damaged, all nodes under its control will be affected. If this function is embedded in each cluster head, the power consumption will increase, due to most of the key management and delivery being

performed by the server node. To reduce the memory overhead as well as maintain security for the network, a new approach is proposed in [9], which is called Survivable and Efficient Clustered Keying (SECK). However, a new issue, high-threat networks cannot be ignored in the path selection algorithms.

2) Power Consumption

To reduce WSN power consumption switching off some nodes has been introduced. Nevertheless, when switching off redundant nodes, how to maintain wireless network coverage and rerouting existing connections become new issues.

In [6], the proposed notion is that "if any parameter on a point can be reliably estimated, then this point can be claimed to be information covered".

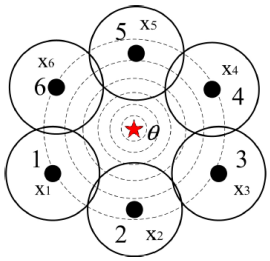


Figure 5. Illustration of physical and information coverage [6]

As shown in Fig. 5, although the sensing area of the node marked by a star is not covered, it may be "information covered" as long as the estimation error is small enough. The information coverage is based on parameter estimation, which means that for an unknown or uncovered point/area θ, there is a set of K sensor nodes for estimation. Each related sensor node has estimation for the non-physical covered point, which is (1),

$$x_k = \frac{\theta}{d_k^{\alpha}} + n_k, k = 1,2,\ldots K. \quad (1)$$

Where x_k denotes the output of the estimation; θ denotes the parameter of the non-physical covered point; d_k denotes the distance between a sensor k and a location with parameter θ; d_k^{α} denotes the attenuation with the distance where α denotes the attenuation exponent; n_k denotes the additive noise. The *mean squared error* (MSE) is used for the evaluation of the estimation for K related sensor nodes.

To try to achieve the sleep/wake up scheme, the authors [15] provided an information coverage estimation for redundant nodes selection named *Distributed Node and Rate Selection* (DNRS). In the proposed scheme, the redundant nodes can be switched off to save power by measuring the distortion. Distortion is any undesired change in the data transmitted such as signal strength and signal format etc. As described in the paper [15], the proposed scheme focused on an area. However, for some applications, such as tracking etc, the proposed scheme will not work properly. Besides, if two sensors are both information covered and rely on each other, which one will be switched off should be discussed.

These proposed methods [6, 15] are based on information coverage and can reduce the impact from transmission distortion. For large density of sensing nodes, a number of adjacent nodes may be available, but the computation overhead will rise. How many samples are enough for one event and how to choose the reporting sensors need to be considered. Besides, in [6] the balance of power-saving and information distortion should be carefully evaluated. In my proposed method, physical coverage will be considered and this situation will be improved.

3) Resilience

Being deployed in a hostile environment, failure/attack probability can not be ignored. Besides, researchers have pointed out that there is no sure and efficient way to readily detect a node capture [16, 17]. Thus, threat probability should be concerned.

In [7], the authors proposed a non-uniform sensor deployment algorithm based on static attack probability to improve the resistance of sensor nodes to node capture. The main algorithm is for estimating the number of keys, called the degree of each sensor node. Let $d_{i,j}$ denote the sensor nodes in the deployment group $G_{i,j}$. In the proposed method [7], the authors claimed that "the higher probability that a deployment group to be attacked implies that $d_{i,j}$ should be set to be a higher value". In other words, the degree of a node in each deployment group should be proportional to its attack probability. Thus, $d_{i,j}$ calculation is the core algorithm of the proposed method. In [7], $D_I(p_{i,j})$ denotes the inner group degree determination function used to calculate the degree of a node in group $G_{i,j}$, which means $d_{i,j} = D_I(p_{i,j})$. When $D_I(\cdot)$ takes input as $p_{i,j}$, it maps $p_{i,j}$ into one of $(\Omega+1)$ values. Formally, $D_I(\cdot)$ can be represented as

$$D_I(p_{i,j}) = \begin{cases} \pi_1, & \text{if } p_{i,j} < \varpi_1 \\ \pi_2, & \text{if } \varpi_1 < p_{i,j} < \varpi_2 \\ \vdots & \vdots \\ \pi_{\Omega+1}, & \text{if } \varpi_\Omega < p_{i,j}. \end{cases} \quad (2)$$

Therefore, if $p_{i,j} \le p_{i',j'}$ holds then $d_{i,j} \le d_{i',j'}$ holds. In [7], $D_I(p_{i,j})$ was designed to be a threshold function. $\{\varpi_1, \varpi_2, \cdots, \varpi_\Omega\}$ is a set of threshold values and $\{\pi_1, \pi_2, \cdots, \pi_{\Omega+1}\}$ is a set of $(\Omega+1)$ values. The values $\{\pi_1, \pi_2, \cdots, \pi_{\Omega+1}\}$ are given such that after the assignment of keys based on the setting, sensor nodes can resist attacks in the corresponding groups. These values are assigned based on experience. In the formula (2), $p_{i,j}$ denotes the normalized attack probability with respect to a deployment group $G_{i,j}$ and $p_{i,j}$ is defined as:

$$p_{i,j} = \frac{\omega_{i,j}}{\sum_{i',j'} \omega_{i',j'}} \quad (3)$$

In equation (3), $\omega_{i,j}$ denotes the attack coefficient associated with $G_{i,j}$. It is a value by considering all of factors and can be calculated as:

$$\omega_{i,j} = b_{i,j} + b_{i',j'} + \sum_{\rho=1}^{\alpha} \sum_{(b_{i',j'}, \tilde{b}_{i',j'}) \in \Psi(G_{i,j}, \rho)} (b_{i',j'} \times g_\rho + \tilde{b}_{i',j'} \times \tilde{g}_\rho) \quad (4)$$

where $\Psi(G_{i,j}, \rho)$ is a set of pairs of the base coefficient $b_{i',j'}$ for $G_{i',j'}$ and base coefficient for data sink $\tilde{b}_{i',j'}$ satisfying that $G_{i,j}$ and $G_{i',j'}$ are at a distance from ρ deployment groups. In [7], g_ρ denotes attack influence factors for sensor nodes and $\tilde{g}_\rho$ denotes attack influence factors for the data sink.

Also, $b_{i,j}$ denotes the basic attack coefficient for sensor nodes, representing the threat from adversaries and $\tilde{b}_{i,j}$ is that for the data sink. Accordingly, the coefficient $b_{i,j}$ and influence factors g_ρ will be used for computing the parameter $d_{i,j}$ that was specified at the beginning of this paragraph.

In [7], $b_{i,j}$ and g_ρ are both static. The proposed method had a good result when attack probability remained unchanged. An obvious improvement was presented not only on memory overhead but also on connectivity maintenance. For the algorithm, the higher attack the probability of the deployment group, the more keys were kept by group members to maintain connectivity. Although it had good performance in the described scenario without node failures, the paper did not consider the effect of nodes being compromised. Besides, static attack probability is not good enough for real scenarios, variable attack and failure probability should be considered. Thus, threat probability should be a concern.

D. Assumptions: The model considered in this thesis

In this paper, the deployment area is set to 2-dimensional and all sensors are randomly scattered. There are one data sink and three cluster heads with fixed location. In the model, a point *u* is covered (monitored) by a node *v* if their *Euclidian distance* is equal or less than the sensing range *R*. Sensor density (*D*) is defined as $D = m/(\pi R^2)$. Assume there are *m* nodes on average in each sensor's signal range and *R* is the radius of the signal range [18].

High threat probability estimation indicates nodes with "high threat" status, which have a high probability to lost packets and may affect their neighbors. In the model, four threat levels are used which are "no threat", "low threat", "high threat" and "compromised". Node attacks may raise the sensors' threat level. A "no threat" node will pass all packets without any packet loss. A "low threat" node may lose packets in a very low probability while sensors with "high threat" may lose packets in a very high probability. "Compromised" sensors will not forward any packets. Also, a sensor connecting to a "high threat" node may raise its threat level. Besides, no transmission error is included in the model.

In this paper, three different sensor distribution and three traffic distribution models are employed. Uniform distribution is used for normal environment monitoring. Normal distribution (*Gaussian* distribution) is used for some special usage such as monitoring forest fire. Zipf distribution is used for simulating some environment changing such as sensors are blown away by strong wind etc.

E. Summary

For WSNs organization, a self-organization mechanism for non-uniform distribution of sensor nodes is proposed in [14], and a location-based scheme for both uniform and non-uniform was described [5]. The mechanism in [14] offers good performance for non-uniform distribution, and an isolated part of the network can be reconnected by mobile sensors/devices. The method can find the optimum path to connect to the data sink efficiently. The solution in [14] relies on mobile robots to maintain connectivity in the network, but in some cases a mobile robot is not available. The location-based mechanism [5] is relying on a special node named "server node". This will raise the cost of the whole network and if this server node is compromised or damaged, all nodes under its control will be affected. If this function is embedded in each cluster head, the power consumption will increase, due to most of the key management and delivery being processed by the server node. To reduce the memory overhead as well as maintain security for the network, a new approach is proposed in [9], which is called *Survivable and Efficient Clustered Keying* (SECK). However, a new issue, default path selection becomes a problem in high-threat networks.

For the WSNs power consumption issue, an information coverage concept [6] has been proposed. In [6], a balance between coverage and sensor density has been explored. In some cases data can be estimated reliably, in other cases it cannot, and estimation cannot be a replacement of actual data.

For resilience, the attack probability estimation algorithm in [7] has a good experimental result for static attack probability estimation and connectivity maintenance. As mentioned before, a variable attack and failure probability is more realistic for WSNs.

For SN addition, a cluster and network-oriented scheme is proposed in [9]. However, the issue of connection between new sensors and existing cluster heads needs to be considered.

This paper focuses on the sensor node organization issues, and a neighbor-oriented self-organization mechanism will be proposed. Four aspects will be described respectively and then the integrative proposed scheme will be detailed.

III. PROPOSED METHOD

To extend the life for wireless sensor networks, redundant nodes can be switched off to save power, and later switched back on to replace failure nodes. Redundant nodes are those that can be switched off and the whole area will still be covered.

In this chapter, the proposed self-organization scheme is examined from three aspects, namely clustering and neighboring, battery life extension, network resilience and new sensor node addition. The self-organization mechanism addresses network maintenance, and is based on Redundant Nodes Selection scheme (RNS) and variable Threats Probability Estimation scheme (TPE). RNS is employed to select the redundant nodes in order to save power. TPE is used to help a sensor node to choose the most suitable path and avoid high-threat neighbors in order to reduce packet loss.

The proposed scheme not only extends battery life, but also enhances network robustness and maintains connectivity. In addition, the proposed scheme can be applied to both uniform and non-uniform distributions.

A. Redundant Node Selection (RNS)

As mentioned in the background chapter, the authors [13] employed a similar system named Distributed Node and Rate Selection (DNRS). The method in [13] is aiming to measure whether a node is redundant by calculating the distortion. However, this method has a limited effect when the objects are appearing random, or sensor density is high.

All in all, less than full coverage will bring in some new problems. For instance, when tracing an object, in the non-covered area we will lose track of the object. Although it may be estimated from related information, it is still not precise.

RNS algorithm has two steps. One is to select redundant nodes and the other is to check whether the redundant nodes can be switched o.

Notion 1: *It is a redundant node, if and only if there are no changes in the covered area when it has been switched to sleep mode.*

A simple sketch map is shown in Fig. 6. In the simulation, the sensor density is much higher. In the Fig. 6, SN_1 can be switched to sleep mode because its original sensing area is covered by other sensor nodes, namely by SN_2, SN_3, SN_4 and SN_5. Thus, even if SN_1 has been switched off, there are no changes in the covered area.

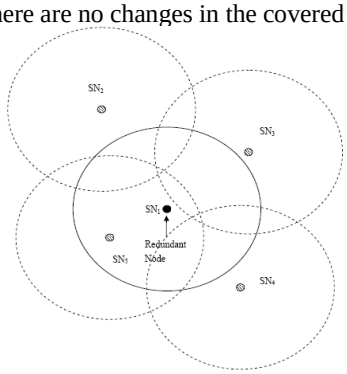


Figure 6. The redundant node

1) The RNS Algorithm and Proof

Assume that every node has its own location information which will be the coordinate in this algorithm and all radio range (radius) will be R. Assume that all sensor nodes are in the same 2-dimensional area. In this thesis, assume a point u is covered (monitored) by a node v if their Euclidian distance is equal or less than the sensing range, i.e., $|uv| \leq R$. Define the sensing circle C(u) of node u as the boundary of u's coverage region.

Notion 2: Let SN_x be a set of sensor nodes. A sensing area is fully covered if and only if for $\forall Z(x,y) \in C(SN_1)$, there exists at least one $C(SN_k)$ $(k > 1)$, that $Z(x,y) \in C(SN_k)$ is true.

Theorem 1: A sensing area of SN_1 is fully covered by a group of nodes if and only if, for $\forall Z(x,y) \in C(SN_1)$, there exists a set of nodes group $G_N = \{SN_2, SN_3, \cdots SN_k\}$ that $Z \in C(SN_2) \bigcup C(SN_3) \bigcup \cdots \bigcup C(SN_k)$ is true.

Proof: Assume for $\forall Z(x,y) \in C(SN_1)$, there exists a set of nodes group $G_N = \{SN_2, SN_3, \cdots SN_k\}$ that $Z \in C(SN_2) \bigcup C(SN_3) \bigcup \cdots \bigcup C(SN_k)$. Without loss of generality, there must be a $C(SN_j), 2 \leq j \leq k$ that $Z \in C(SN_j)$ is true.

The algorithm has the following steps:

1. Put a node SN1 at the origin of the coordinate system.
2. Assume there is a node SN2 in C(SN1) which means node SN2 is in node SN1's radio range, vice versa. Without loss of generality, let SN2 be on the X-axis, shown in Fig. 7.
3. The X-axis and circle C(SN2) intersect at P'. C(SN1) and C(SN2) intersect at Q' and Q".
4. To find the next circle.

4.1 If there is a node SN3 in the area Q'SN1W and SN2Q' ≤ R, then go to step 3 and replace SN2 by SN3. If SN2 could be the next circle, then node SN1 can be switched to sleep mode, as shown in Fig. 8.

4.2 If there is a node SN3 in the area Q'SN1W and SN2Q' > R, as shown in Fig. 9, there will be a small area A'B'Q' which is not covered. If there exists a node SNE that SNEA' ≤ R, SNEB' ≤ R, SNEQ' ≤ R, then return to step 3 and replace SN2 by SN3. If there is no such SNE, the node SN1 will not be switched off, as shown in Fig. 10.

4.3 If there is not any nodes in the area Q'SN1W: Assume that there is a node SNA which is outside the area Q'SN1W, as it is shown in Fig. 11, and SNASN1 < 2R, SNASN2 ≤ 2R, and SNAQ' < R. The C(SNA) and the C(SN1) intersect at C' and C' is in the area Q'SN1W. Then, go to step 4.3 and replace Q' and SN2 by C' and C". If Q" is in the next circle, then node SN1 can be switched to sleep mode. If there is no such node SNA, the node SN1 can not be switched off.

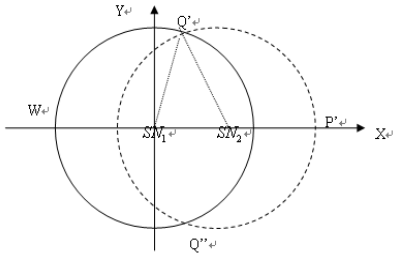


Figure 7. Put both nodes in the coordinate system

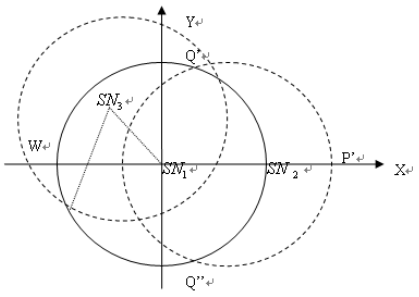


Figure 8. Step 4.1

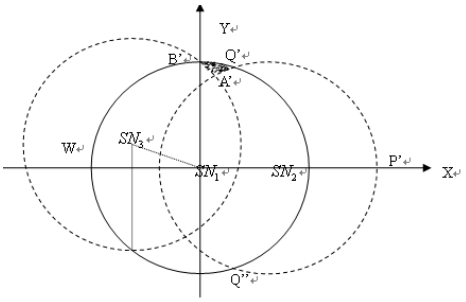


Figure 9. Step 4.2

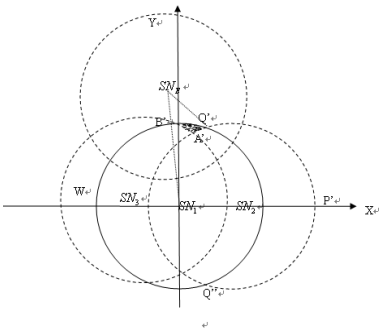


Figure 10. Step 4.2

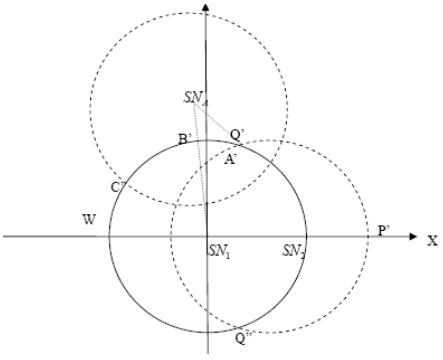


Figure 11. Step 4.3

Notion 3: If there exists a set of nodes $R = \{SN_{m1}, SN_{m2},\ldots SN_{mn}\}, n \in N$, for any $SN_{mk}(1 \le k \le n)$ that SN_{mk} is a redundant node, but if any node $SN_{ml}(1 \le l \le n)$ from among them is switched off, there will be at least one node $SN_{mp}(1 \le p < n)$ that is no longer redundant, then the node SN_{ml} and all SN_{mp} are defined as redundant related nodes (RR nodes or RRNs) and SN_{ml} is defined as redundant related seed (RRS).

Especially, there may be only one SN_{mp} related with SN_{ml} and both of them are RRS for each other. Then, these two nodes are defined as twin redundant related nodes (TRRNs), shown in Fig. 12.

Theorem 3: Only one of the TRRNs can be switched off.

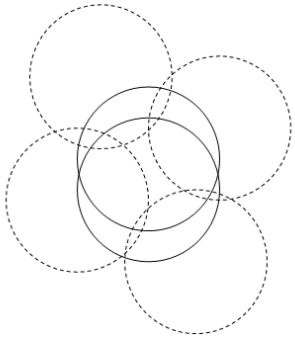


Figure 12. Twin redundant related nodes

The algorithm of switching the RR nodes is to separate them into TRRNs. First, select a RRS from RR nodes and search all of the SN_{mp} that whether there is a twin node for the RRS. If there is, switch one of them off and put the other one back. Besides, the selection algorithm, from step one to step four, does not cover all coverage probabilities because the computation overhead still need to be considered.

The selection for the TRRNs is based on the Variable Threats Probability Estimation algorithm which is specified in the following section. The higher threat probability estimation node will be switched off.

Besides, for any sleeping node SN_s, let a point in the deployment area be Z=(x, y). If there exists $\exists Z,\ |ZSN_i| > R, |ZSN_s| \le R, i \ne s$, SN_s will be switched on.

B. Variable Threats Probability Estimation

"There is no sure and efficient way to readily detect a node capture." The authors mentioned in [10] and [11]. Accordingly, a threats probability estimation algorithm is employed in the self-organization scheme.

The algorithm proposed here is an extension to the one originally described in [7], in which variable attack and failure probability will be involved. Consider that sensor nodes are deployed in groups, as shown in Fig. 3. The authors [7] proposed an algorithm based on static failure or attack probability for key predistribution. In their study, they also consider that a sensor node can play the role of data sink.

In our proposed method, a new deployment model is used. Compared with that in [7], the attack and failure probability does not focus on groups but on individual sensors. A new algorithm is proposed to keep the whole network connected and avoid key threats.

In the paper, we assume that the data sink is chosen before the network is deployed and will not be replaced. It is the same with cluster heads. Moreover, my proposed algorithm focuses on sensor nodes rather than on deployment groups which was proposed in [7]. The sensor nodes will not distinguish between an attack from a neighbor and one from an adversary. For security of Trusted Neighbors, I refer to *Reputation-based Framework for High Integrity Sensor Networks* [10].

In our proposed algorithm, the sensor nodes reporting to the same cluster head are defined as in one group. Assume that all deployment groups are in the same 2-dimensional area. Assume that all sensor nodes will automatically record threats' times and levels. Assume that only sensor nodes will be under threat. The algorithm for cluster heads can be derived similarly and will be discussed in the future work section.

In the following, the estimation of node threat level is described. It follows the general PID controller principle [19], and adjusts the estimate threat level according to three correction factors. The whole algorithm can be described as: if there are no threats detected by a node, the threat value of this node will drop gradually; if threats are detected by a node, there will be a correction value (derived from three correction factors) added on this node's threat value to estimate the threat level.

In my proposed algorithm, a sensor node $S_i (i \in N)$ is associated with a basic failure coefficient b_{S_i} representing the threat from the environment and b_{S_i} is predistributed value by experience. Let w be the variable attack and failure weight, which is obtained by experience. Let φ be threshold threat level and let D_φ be the detected attack level. Then w is defined as

$$w = \begin{Bmatrix} w_1 & , D_\varphi = \varphi_1 \\ w_2 & , D_\varphi = \varphi_2 \\ \vdots & \vdots \\ w_n & , D_\varphi = \varphi_n \end{Bmatrix}, n \geq 1 \tag{5}$$

Here n denotes the number of threat levels. Let $\{\varphi_1, \varphi_2, \cdots, \varphi_n\}$ be a set of threshold values and N_T shows how many times a certain threat level has been detected. For a certain threat level, the failure estimate is $F = wN_T$. The difference between the failure estimates at time k and that at time (k-1) can be defined as

$$e(k) = F_{S_i}(k) - F_{S_i}(k-1) \tag{6}$$

which is the first correction factor. Here $F_{S_i}(k)$ denotes the estimate failure of S_i at time k. Then the second correction factor M_c, relative threat value, can be defined as

$$M_c(k) = \theta \frac{F_{S_i}(k)}{F_{C(S_i)}(k)} \tag{7}$$

where θ is a constant coefficient. $F_{C(S_i)}(k)$ denotes the sum of failure estimate of S_i 's neighbors, which can be defined as

$$F_{C(S_i)}(k) = F_{S_i}(k) + \sum_{S_j \in C(S_i)} F_{S_j}(k) \tag{8}$$

The third correction factor is connectivity detection, which is the number of sensors that have at least an available path to a cluster head. It can be defined as

$$CD_{S_i}(k) = (1 - \frac{CNT_{S_i}(k)}{CNT_{S_i}})(CNT_{S_i} - CNT_{S_i}(k)) \tag{9}$$

Here $CNT_{S_i}(k)$ denotes the connectivity of S_i at the time k and CNT_{S_i} denotes the connectivity when the network first deployed. If $CD_{S_i}(k) > 0$, it means some nodes are compromised or unavailable at the time k.

The first correction factor measures the diversity between different time intervals. The second factor measures the relative threat. The third factor measures the connectivity for a sensor and its neighbors. The total correction measurement derived from (6), (7) and (9) can be defined as

$$M_A(k) = \omega \cdot (e(k) + M_c(k) + \gamma CD_{s_i}(k) + \frac{1}{\alpha}\int (e(k)) + M_c(k))dk + \beta \frac{d}{dk}(e(k) + M_c(k)) \tag{10}$$

Here α, β, γ are constants and will be set based on experience. Let δ , a constant, be the decreasing threat value, which means if no failure/attack problems are detected, b_{si} will gradually drop down. Then b_{si} can be defined as

$$b_{S_i}(k) = b_{Si}(k-1) - \delta + M_A(k) \tag{11}$$

If $b_{S_i}(k) < 0$, then let $b_{S_i}(k) = 0$. The $b_{S_i}(k)$ is the real time threat estimate and this will also be used in RNS algorithm for TRRNs.

TABLE I. PARAMETERS IN TPE

Parameters		Formula
0	Sensor failure coefficient	$M_c(k) = \theta \frac{F_{S_i}(k)}{F_{C(S_i)}(k)}$
ω	TPE coefficients based on experience	$M_A(k) = \omega \cdot (e(k) + M_c(k) + \gamma CD_{s_i}(k)$
β		$+ \frac{1}{\alpha}\int (e(k) + M_c(k))dk + \beta \frac{d}{dk}(e(k) + M_c(k)))$
γ		
ω	Amendment coefficient	

After real time threat estimate b_{si} is calculated, the degree of a sensor node can be derived. A degree of a sensor node denotes the number of available connections for a sensor node (number of shared keys with neighbors). Given a set of threshold $\{\varpi_1, \varpi_2, \cdots \varpi_n\}$, $n \geq 2$ for any sensor nodes, the degree can be calculated by

$$K(S_i) = \begin{Bmatrix} \kappa_1 & , b_{s_i} = \varpi_1 \\ \kappa_2 & , b_{s_i} = \varpi_2 \\ \vdots & \vdots \\ \kappa & , b_{s_i} = \varpi_n \end{Bmatrix}, n \geq 2 \tag{12}$$

Here $K(S_i)$ denotes the degree of a sensor node in a deployment group. In the formula (12), for $\forall i, j, i \neq j$, if $\varpi_i \geq \varpi_j$, then $\kappa_i \leq \kappa_j$. In my proposed method, differently from that in [5], the higher attack/failure probability a sensor has the fewer keys it has.

C. Organizing

In this section, a neighbor-oriented clustering and neighboring scheme is described which is supported by RNS and TPE. As it has been mentioned in the background chapter [13], the sensor nodes may be scattered randomly e.g. scattered from an airplane. Thus, there is no guarantee that all cluster heads can directly connect with the data sink or all group nodes can directly connect with their respective

cluster head. Routing with no threats is described in section *1)*, while in section *2)* variable threats are involved.

1) Normal Routing

In this section, the thesis considers the routing algorithm only between sensor nodes without any threats. To discover a primary cluster head (CH), each sensor node (SN) wants to discover the ID of the closest CH.

In our proposed scheme, let SNID be the ID of a SN, DP be Default Path, DD be the Depth of Default Path, DTD be Distance to Default Path, SLP be the status (sleeping or active), TL be Threat Level, NL be a Neighbor List. Then, we give each SN an expression (13):

$$SN_i = \{SNID, DP, DD, DTD, SLP, TL, NL(\omega)\} \quad (13)$$

Where ω is the index of SN's neighbors. SN's parameters are expressed with dotted notation, for instance, SN.DP denotes the SN's Default Path.

After deployment, the RNS algorithm (detailed in the next section) is activated. SNs which are redundant will be set to SLP = 1. At the same time, all SNs discover all their neighbors and store in NL. Then they will wait for connection from their neighbors which can connect to cluster heads and the algorithm is described as follows. At first, each CH will search SNs within its sensing range. Each SN within CH's sensing range will update its expression (14).

$$SN = update\{DP = CH, DD = 0, DTD, SLP = 0, NL(\omega)\} \quad (14)$$

Then these SNs continue to tell their neighbors they can communicate with a cluster head by sending a path message (15):

$$PathInfo = \{SNID = SN, DP = CH, DD = 0, DTD, SLP = 0\} \quad (15)$$

The SNs who receive the path message (15) will update their expression (16).

$$SN = update\{DP, DD, DTD, SLP)\} \quad (16)$$

If a SN receives more than one path message, it will calculate the power consumption (PC) on these different communication paths. As mentioned above, the proposed scheme is neighbor-oriented. The default path selection algorithm is described in (17).

$$SN = update\{PC(DP, DD, DTD)\} \quad (17)$$

Here PC denotes the power consumption function. Before give the expression for total power consumption, assume power consumption is proportional to the square of distance with a coefficient θ and each hop will consume λ. Thus, the power consumption function can be described as

$$PC(SN, NL(index)) = \theta \cdot (DTD^2 + NL(index).DTD^2) + \lambda \cdot DD \quad (18)$$

NL(index).DTD denotes the distance between a neighbor's default path and the neighbor. The paths from a SN to a CH are called communication links. For instance, SN_i need to send messages to SN_j, and SN_j need to forward to SN_k then finally to the CH, the link $i \to j \to k \to CH$ is called a communication link.

Theorem 2: For any SNs, if there exists $PC_1(SN,NL(index_1)) < PC_2(NL(index_2))$, the power consumption of the communication link on $NL(index_1)$ is less than that on $NL(index_2)$ is true.

Proof:

$$\begin{aligned} PC(SN, NL(index_1)) &= \theta \cdot (DTD^2 + NL(index_1).DTD^2) + \lambda \cdot DD \\ &= (\theta \cdot DTD^2 + \lambda) + (\theta \cdot NL(index_1).DTD^2 + (\lambda \cdot DD - 1)) \\ &= PC(SN) + PC(NL(index_1)) \end{aligned}$$

PC(SN) denotes the power consumption between the SN and its default path and $PC(NL(index_1))$ denotes the power consumption between the node of SN's default path and the node of SN's default path's default path. For any available communication links, we define a searching function (SF), that SF(SN) = $SN.NL(index_1)$ if for any n ($1 < n \leq N$, N is the number of SN's neighbor), there is $PC_1(SN,NL(index_1)) < PC_n(NL(index_n))$, where $SN.NL(index_1)$ denotes the SN's neighbor with $index_1$. We define $SF(SF(SN)) = SF^2(SN)$, then the most power-saving path will be the communication link $SN \to SF(SN) \to SF^2(SN) \to \cdots \to SF^{SN.DD+1}$. Thus, for any SNs, if there exist $PC_1(SN,NL(index_1)) < PC_2(NL(index_2))$, that the power consumption of the communication link on $NL(index_1)$ is less than that on $NL(index_2)$ is true.

Lemma 1: For any SNs, if there exists $PC_1(SN,NL(index_1)) < PC_n(NL(index_n))$,($1 < n \leq N$, N is the number of SN's neighbors), the $NL(index_1)$ is the minimum power-saving link for SN.

Based on Lemma 1, all SNs can find the minimum power path. If a sleeping node is on a minimum power path, then it will be switched on. An example of the normal routing is shown in Fig. 13.

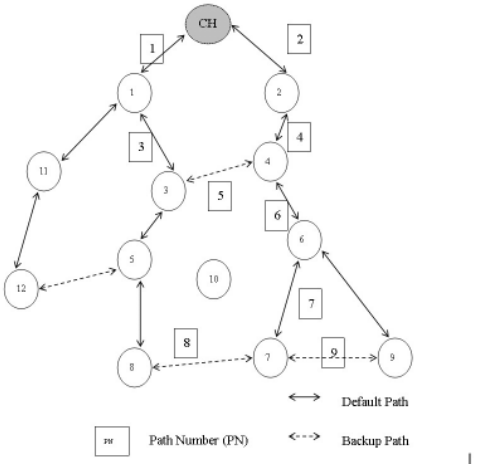


Figure 13. Normal routing

Fig. 13 shows the normal routing scheme. In RNS, SN3, SN4, SN5 and SN10 are candidates of redundant nodes. SN1 and SN2 are both in CH's sensing range and they have the default path DP = CH and DD = 0. Then, they start to send path messages to neighbors. Both SN3 and SN11 receive a path message from SN1 and a path message from SN4 is received by SN3 as well. Then based on Lemma 1, SN1 is set as the default path for SN3 and SN4 is set as backup. The rest may be deduced by analogy. After the default path searching, SN3, SN4 and SN5 are all on the minimum power link, thus, only SN10 will be switched off.

2) Routing with Variable Threats

In this section, variable threats are assumed, and attack threats and sensor failure will be discussed.

Based on TPE, each SN has a property called Threats Level (TL). In the routing scheme under variable threats, we assume there are three threat levels. Let level 0 be no threats, level 1 be low threats, level 2 be high threats. Thus, there are three different cases.

SN.TL = 0. Under this circumstance, the neighbor with TL = 0 and lowest power consumption is a priority selection. If there is no SN with TL = 0, the one with TL = 1 and lowest power consumption is a priority selection.

SN.TL = 1. Under this circumstance, the solution is the same with SN.TL = 0. For a more complex scenario, the balance between threats level and power consumption is mentioned in the future work.

SN.TL = 2. Under this circumstance, the SN has a high probability of failure or of being compromised. In the proposed method, TPE, this "high threat" SN may be totally isolated because it will be disconnected from "low threat" or "no threat" neighbors and only neighbors with TL = 2 can be used in the default path to the sink.

Besides, a SN will delete neighbors with TL = 2 from SN.NL.

For a failed SN, its neighbor will delete its index from SN.NL. If a CH becomes unavailable, such as due to physical damage or battery exhaustion, the SNs in its group will join another group using a backup path. Fig. 14 shows the scheme under variable threats.

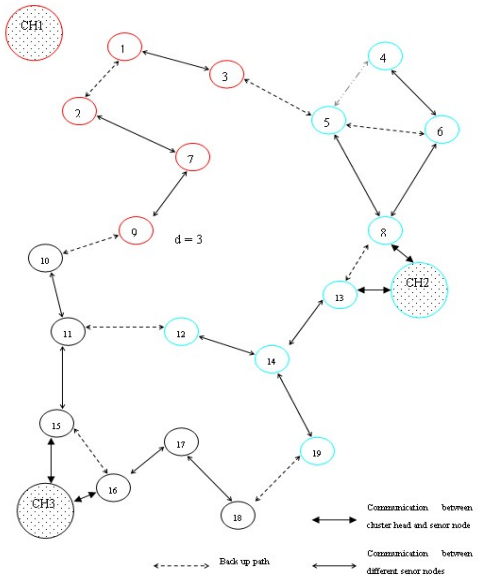


Figure 14. Routing with variable threats

In Fig. 14, we assume SN5 has a higher threat level than SN4, thus, SN4 will select SN6 as default path. If SN5 has a TL = 2, it will be isolated by SN4, SN3, SN6 and SN8 in order to avoid key and packet loss. If assume that CH1 is unavailable and SN5 has a normal threat level, then the backup path between SN1 and SN2, SN3 and SN5, SN9 and SN10 will be set as default path and all SNs in group 1 will join group 2 or group 3.

D. Sensors Addition

Throughout the lifetime of a WSN, it may be necessary to deploy additional SNs. In my proposed scheme, the network is flexible to receive additional SNs. We assume the additional SNs are randomly deployed in the monitored area. Based on the scheme specified and Lemma 1, new SNs will join a suitable group.

E. Proposed Scheme

The *3.A*, *3.B* and *3.C* will work as an integrative and the entire proposed scheme is outlined as follows: after scattering on the wild area, the RNS is activated. Each cluster head will search the data sink within its communication range. When a cluster head receives an available path to the data sink, 1) the cluster head will start to search other cluster heads within its sensing range (neighbors); 2) the cluster head will start to search sensor nodes around it to organize a group; 3) the sensor nodes will calculate the threshold $b_{S_i}(0)$. If a sensor node receives no available path or message to become a group member, it will be switched to sleep mode. If a redundant node is the only available path or the minimum power path for a sensor node, it will not be switched off.

If the environment changes: 1) if a cluster head becomes unavailable, all of its group members will join other groups via the algorithm specified; 2) if a sensor node SN has detected a high threat level, all sensor nodes connecting to SN will reroute to the first backup path; 3) if new sensor nodes join the network, they will follow *3.C*; 4) in case of a node pair (S_i and S_j), when only one of the pair can be switched to sleep mode, the following calculation is used. If S_i's $b_{S_i}(k)$ is higher than the sleep one S_j's, S_i will be switched to sleep mode and S_j will be waked up, vice versa.

Algorithm: Proposed Scheme

Sensors do RNS algorithm;
for $CH_i (1 \le i \le N)$ **do**
  search data sink;
  if CH_i find data sink **then**
    register at the data sink(CH_i);
    search neighbors(CH_i);
    search sensor nodes(CH_i);
  end if
end for
for $S_j (1 \le j \le M)$ **do**
  if S_j can access to a cluster head **then**
    search neighbors(S_j);
    organize network;
  end if
end for

for $S_j(1 \le j \le M)$ **do**

if RNS(S_j)is true and S_j is not the only path for another node **then**

switch(S_j,sleep);

end if

end for

if $CH_i(1 \le i \le N)$ cannot access to data sink **then**

switch(CH_i,sleep);

end if

if $S_j(1 \le j \le M)$ cannot access to cluster head **then**

switch(S_j,sleep);

end if

for $S_j(1 \le j \le M)$ **do**

if S_j default path is sleep or unavailable **then**

load the top of the stack and set as default path;

end if

end for

Thus, after the deployment, the network will be automatically organized and redundant nodes will be switched to sleep mode to save power. When some nodes become unavailable, some of the sleep nodes will be set as replacements and maintain the network.

IV. Analytical and Simulation Results

In this section, simulation results are given for total energy consumption and total packages lost in the network. The latter indicates network robustness, that is, the ability of the network to continue operating after variable threats.

In the simulation, a JAVA based wireless sensor network simulator was used. Compared with other simulators, such as OMNeT++, the JAVA simulator proved to be more flexible for environment configuration and implementation of the proposed solutions.

A. The Impact of Sensor Failure on Network Integrity

The proposed model, RNS can reduce partitioning in the network. An example illustrating the RNS algorithm when some sensor nodes are unavailable is shown in Fig. 16. In this example, there are 15 sensor nodes, the topology and deployment group is shown in Fig. 16. If the sensor nodes in the ellipse are unavailable, the default path will be blocked. However, the node which is switched to passive and sleep mode is not affected by the attack, and when the default path for the first forwarding sensor is blocked the sleeping node will be woken up to reconnect to the cluster head.

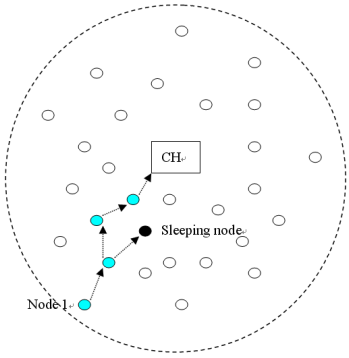


Figure 15. Message delivery from a sensor to the cluster head

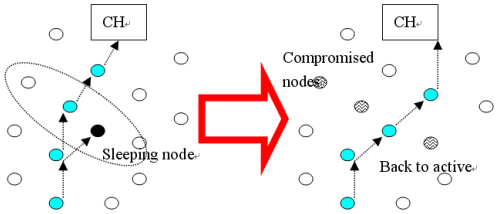


Figure 16. The impact of sensor failure

However, the problem of resistance to attacks becomes more delicate when the location of cluster heads is taken into consideration, and this will be detailed in the following section.

B. Resilience Against Node Failure

In this section, resilience against node failure between deployment groups is described.

An example illustrating the RNS algorithm in the case a cluster head fails is shown in Fig. 17. In this example, first, cluster head 1 is damaged and default paths for node 1 and 2 are blocked. In my proposed scheme, the isolated nodes will search for an available path to a new cluster head in order to join a new deployment group. As shown in Fig. 17, node 9 is switched to its backup path that connects to node 10, and node 7 and node 9 will join to cluster head 3, meanwhile, node 2, node 1 and node3 will join to cluster head 2. If cluster head 2 fails, all these sensor nodes will join group 3 via the backup path. Although this may raise communication overhead, it can maintain coverage over the whole monitoring region.

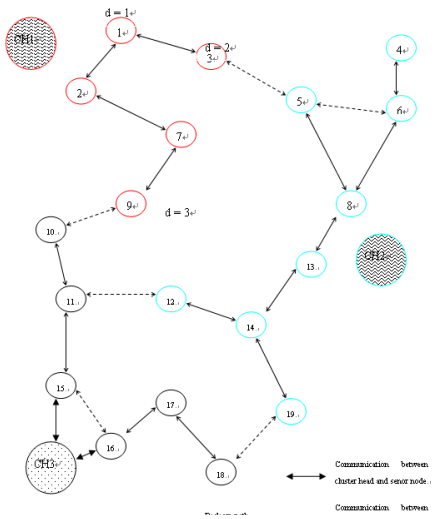


Figure 17. Resilience against node failure

C. Test Environment

In this section, simulation results are given for total energy consumption and total packages lost in the network. The latter indicates network robustness, that is, the ability of the network to continue operating after variable attacks.

In the simulation, a JAVA based wireless sensor network simulator was used. Compared with other simulators, such as OMNeT++, the JAVA simulator proved to be more flexible for environment configuration and implementation of the proposed solutions. The tests can be grouped into two parts. The first part describes the total energy consumption before and after implementing the Redundant Nodes Selection scheme (RNS). The second part depicts the robustness of the sensor network under variable threats with the proposed Threats Probability Estimation Scheme and with the shortest path first (SPF) scheme [9]. Robustness is examined with three different kinds of sensor distributions. Sensor Density in uniform distribution is 8 and in other distribution is 6, because in the simulator, the uniform distribution has a fixed template, and the number of sensors was the same in all simulation.

D. Power Consumption

Uniform sensor distribution and uniform traffic distribution was used when examining the total energy consumptions between before and after implementing proposed method. In the simulator, each hop costs 0.0005% battery life, around 0.00006% per simulation meter and 0.001% per working sensor.

Simulations with different parameters produced similar results. A typical set of results is presented here. There were 100 sensors with three cluster heads and sensor density is 8. The sensors were arranged in a 10 × 10 matrix. Fig. 18 shows the total battery consumption without proposed scheme and Fig. 19 shows the total consumption under the same circumstance but with self-organization scheme activated.

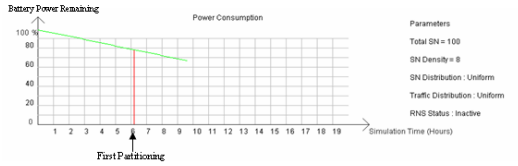


Figure 18. Total power consumption in uniform distribution without RNS

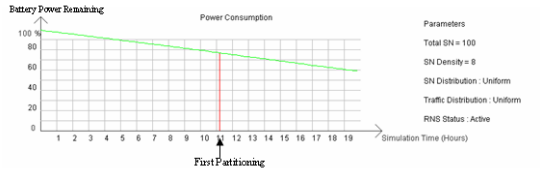


Figure 19. Total power consumption in uniform distribution with RNS

Then we look at the differences in normal distribution. Fig. 20 shows total power consumption without RNS scheme. Fig. 21 shows the total consumption under the same circumstance but with RNS activated. In the scenario with Zipf distribution, Fig. 22 and Fig. 23 show the improvement.

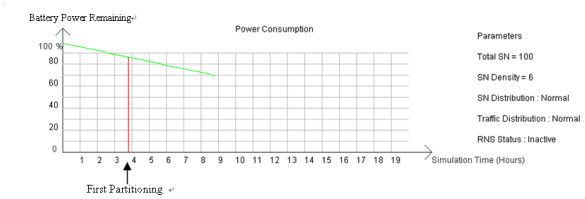


Figure 20. Total power consumption in normal distribution without RNS

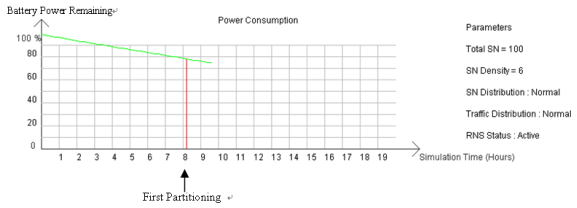


Figure 21. Total power consumption in normal distribution with RNS

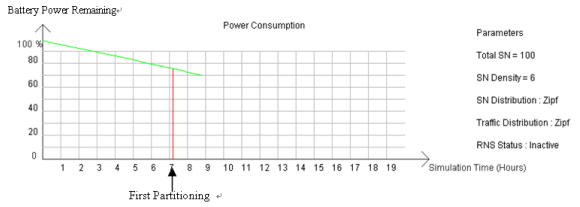


Figure 22. Total power consumption in Zipf distribution without RNS

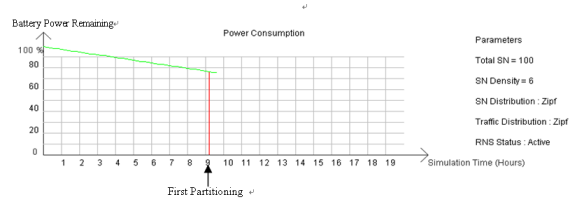


Figure 23. Total power consumption in Zipf distribution with RNS

The charts prove that the RNS scheme could reduce energy consumption of the network, and thereby extend the network's lifetime.

E. *Robustness*

Node captures in hostile environments are inevitable. Robustness is a kind of ability to help WSN recover from variable threats. This thesis describes a Threats Probability Estimation (TPE) scheme to support the key management method described in Chapter *3.B.* For comparison, the default key management scheme is shortest path first (SPF). There are total 100 sensors were deployed in such area and average sensor density is 6. One fourth of the sensors are set as "high threats" and random attacks are launched six times in each scenario. In the simulator the process of detecting an attack was not modeled, but rather the attack event was directly passed on to the sensors.

1) Normal sensor distribution with normal traffic distribution

First, we look at the scenario when sensors are deployed in a normal distribution (Gaussian distribution). The traffic distribution is also normal, to simulate centralized events, such as fire in a forest. The Fig. 24 shows the robustness of the network with SPF scheme and Fig. 25 shows that with TPE algorithm.

The curves illustrated above indicate the TPE scheme could enhance the robustness of the network, by reducing packet loss from 15% to 2%.

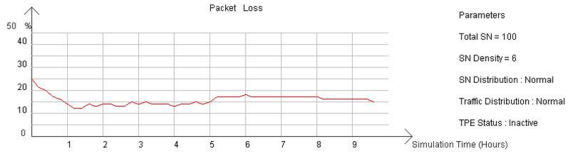


Figure 24. Robustness of the network with SPF in normal distribution

Figure 25. Robustness of the network with TPE in normal distribution

2) Uniform sensor distribution with uniform traffic distribution

Fig. 26 shows robustness of the network with SPF scheme when sensor deployed and network traffic follows uniform distribution. Fig. 27 shows the robustness of the same circumstance but with the same distribution, but with TPE activated.

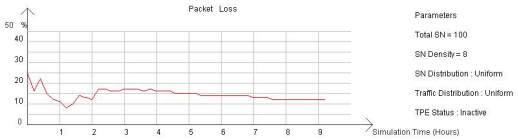


Figure 26. Robustness of the network with SPF in uniform distribution

Figure 27. Robustness of the network with TPE in uniform distribution

In uniform distribution, the TPE scheme improves the robustness by reducing packet loss from 14% to 4%.

3) Zipf sensor distribution with Zipf traffic distribution

Fig. 28 shows the robustness of the network with SPF when sensor deployed and network traffic follow Zipf distribution. Fig. 29 shows the robustness of the same circumstance but with the same distribution, but with TPE activated.

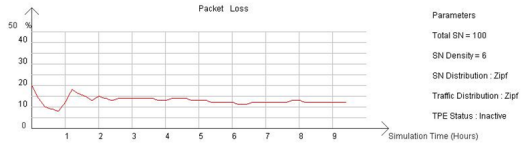


Figure 28. Robustness of the network with SPF in Zipf distribution

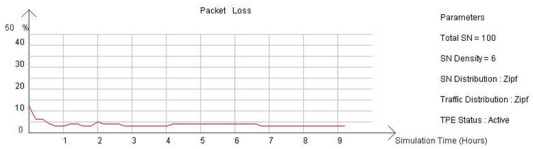


Figure 29. Robustness of the network with TPE in Zipf distribution

In Zipf distribution, the TPE scheme improves the robustness by reducing packet loss from 13% to 4%.

With high threats in the network, TPE can be considered as an effective scheme to improve the robustness by reducing packet loss.

4) Sensor Addition

In the scenario with proposed method, new sensors were successfully added to the network as long as there was at least one available "no threat" neighbor or "low threat" neighbor. Obviously, adding new sensors will improve network connectivity and coverage.

Discussion and evaluation will be detailed in the following Chapter.

V. DISCUSSION AND CONCLUSION

Power consumption and robustness are two important aspects of self-organization for wireless sensor networks. Throughout the lifetime of a wireless sensor network, a partitioning inevitably occurs after a certain period of time, because of finite battery power. Extending the battery life and postponing the occurrence of the first partitioning are the problems of the former aspect. The latter aspect describes the ability of a WSN to recover from different attacks.

In the proposed method, Redundant Nodes Selection scheme (RNS) and variable Threats Probability Estimation scheme (TPE) are designed to improve network performance on these two aspects. RNS is proposed to select redundant nodes and switch them off to save power, and set them back to active mode when a partitioning occurs. A node is redundant if after switching it off, its sensing area is still covered by neighbors. TPE is designed to enhance the routing algorithm and avoid high threats sensors in order to reduce packet loss.

The experimental results presented in the previous chapter showed that the battery life is extended dramatically. In uniform distribution, the total remaining battery power dropped to 80% after around 6 simulation hours without RNS, while under the same circumstance but with RNS activated, it took approximate 9 simulation hours, which represents a 50% improvement. The figures show that the battery consumption curve for the scenario with RNS has a gentler slope. Besides, from about 6.25 simulation hours, in the scenario without RNS a partitioning appeared because of sensor exhaustion. After that, connectivity and coverage also went down. However, in the scenario with RNS, sleeping nodes replaced the exhausted ones and maintained connectivity, and the first partitioning occurred at around 11.25 simulation hours, which is an 80% improvement.

Similarly, in normal distribution, the total remaining battery power dropped to 80% after around 5.75 simulation hours without RNS, while under the same circumstance but with RNS activated, it took approximate 7.5 simulation hours, which represents a 30% improvement. In addition, it took approximate 3.75 simulation hours to meet the first partitioning in the scenario without RNS. Compared with that, scenario with RNS has around 8.25 simulation hours without partitioning, which is a 140% improvement.

In Zipf distribution, the total remaining battery power dropped to 80% after around 6 simulation hours without RNS, while under the same circumstance but with RNS activated, it took approximate 8 simulation hours, which represents a 33% improvement. In addition, it took approximate 7.25 simulation hours to meet the first partitioning in the scenario without RNS. Compared with that, the scenario with RNS has around 9.25 simulation hours without partitioning, which is a 27% improvement.

For the implementation of RNS, redundant nodes are switched off to save power and then switched on to replace power-exhausted sensors. This can help the WSN to extend battery life and postpone the occurrence of first partitioning. The calculation for the redundant nodes was only executed once, after network deployment. Thus, there was not much computation added during runtime and we ignored the computation overhead in the simulation.

On the other hand, variable threats, such as node capture attacks and nodes failure in hostile environments are inevitable. A sensor with high threat level indicates a high packet loss probability when packets are received or sent. TPE scans all neighbors and helps sensors to avoid high-threat nodes to reduce packet loss. The scenario with my proposed scheme, TPE, is also showed a distinct improvement. In the simulation with sensor normal distribution, the scenario without TPE had approximate 15% packet loss on average while in the network with TPE, it had around 3% packet loss during simulation times, which is a 400% improvement.

Similarly, in uniform distribution, the scenario without TPE had approximately 14% packet loss on average while in the network with TPE, it had around 4% packet loss during simulation times, which is a 250% improvement.

In Zipf distribution, the improvement is also significant. I observed 13% packet loss in the scenario without TPE while under the same circumstance but with TPE, it had 4% packet loss, which is a 225% improvement.

In the simulation, the level of packet loss in the scenario without TPE usually stayed at a high level, although it fluctuated sometimes. On the other hand, in the scenario with TPE it always dropped quickly to a low level and stabilized, despite sometimes having a small rise at the beginning. TPE helps routing by avoiding higher threat neighbors, thus, a lower packet loss is obtained when variable threats are involved.

In my proposed method, the improvement by RNS depends on sensor density, the higher the density, the more improvement. Low sensor density networks will not benefit significantly from RNS. TPE is designed to counter variable threats and there will not be much improvement on the scenario without variable threats. Also, TPE may slightly raise the communication overhead and memory overhead because of rerouting to a safer neighbor. When a network was in low threat, the communication overhead was the same as the scenario without TPE. However, as the threat level goes up, the communication overhead was rising. In the simulation, the communication overhead was approximately 0% to 12% more than in the scenario without TPE, which was the price for significantly lower packet loss.

ACKNOWLEDGMENT

I would like to thank my parents and my friend Rui Rui Zhang for their understanding and support.

REFERENCES

[1] Jian Zhong and Peter Bertok, "A Variable Threats Based Self-Organization Scheme for Wireless Sensor Networks," 3rd Intl. Conf. on Sensor Technologies and Applications, 2009 (SENSORCOMM '09), pp. 327-332, doi: 10.1109/SENSORCOMM.2009.57

[2] J. A. Stankovic, "Self-organizing Wireless Sensor Networks in Action," J. of Networking and Mobile Computing, vol 3619/2005, doi: 10.1007/11534310_1

[3] B. Karp, "Geographic routing for wireless networks," PhD Dissertation, Harvard University, October 2000. URL http:// actcomm.dartmouth.edu/papers/karp:paper.pdf

[4] R. Di Pietro, L. V. Mancini and A. Mei, "Energy efficient node-to-node authentication and communication confidentiality in wireless sensor networks," Intl. J. on Wireless Netowrks, vol 12, pp 709-721, doi: 10.1007/s11276-006-6530-5

[5] D. Liu and P. Ning, "Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks," In Proc. of the 10th Annu. Network and Distributed Sensor Networks (NDSS03) , 2003.

[6] B. Wang, K. C. Chua and V. Srinivasan, Wei Wang, "Sensor density for complete information coverage in wireless sensor networks," Intl. J. of EWSN2006,vol 3868/2006, pp. 69-82, doi: 10.1007/11669463.

[7] C. Yu, C. Li, C. Lu, D. Lee and S. Kuo, "Attack probability based deterministic key predistribution mechanism for non-uniform sensor deployment," In Proc. of the 27th Intl. Conf. on Distributed Computing Systems Workshops, pp 18, doi: 10.1109/ICDCSW.2007.24.

[8] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks," University of California at Berkeley, 2002. URL http://webs.cs.berkeley.edu/papers/sensor-route-security.pdf

[9] M. W. Chorzempa, "Key management for wireless sensor networks in hostile environments," Paper submitted to the Faculty of the Virginia Polytechnic Institute and State University, 2006. URL http://scholar.lib.vt.edu/theses/available/etd-05022006-171402/unrestricted/chorzempapaper_v2.pdf

[10] Saurabh Ganeriwal and Mani B. Srivastava, "Reputation-based Framework for High Integrity Sensor Networks," SASN'04, October 25, 2004, Washington, D.C., USA

[11] G. Jolly, M. Kusu, and P. Kokate. "A hierarchical key management method for low-energy wireless MSN networks," In Proc. of the 8th IEEE Symposium on Computers and Communication (ISCC), pp. 335–340. Turkey, July, 2003.

[12] S. Zhu, S. Setia and S. Jajodia. "LEAP: efficient security mechanisms for large-scale distributed MSN networks," In Proc. of the 10th ACM Conf. on Computer and Communication Security (CCS), pp. 62-72, Washington DC, October 2003.

[13] B. Krishnamachari, "An introduction to wireless sensor networks," Tutorial Presented at the Second Intl. Conf. on Intelligent Sensing and Information Processing (ICISIP), Chennai, India, 2005.

[14] F. Dressler, B. Krüger, G. Fuchs and R. German, "Self-organization in sensor networks using bio-inspired mechanisms," In Organic-computing, 2005. URL http://www.organic-computing.org/ conferences/arcs2005/dressler-article.pdf

[15] B. Atakan and Özgür B. Akan, "Immune system-based energy efficient and reliable communication in wireless sensor networks," In Next Generation Wireless Communications Laboratory, 2007. URL http://www.springerlink.com/index/14g2255603256jn6.pdf

[16] G. Jolly, M. Kusu, and P. Kokate, "A hierarchical key management method for low-energy wireless MSN networks," In Proc. of the 8th IEEE Symposium on Computers and Communication (ISCC), pp. 335–340. Turkey, July, 2003.

[17] S. Zhu, S. Setia and S. Jajodia, "LEAP: efficient security mechanisms for large-scale distributed MSN networks," In Proc. of the 10th ACM Conference on Computer and Communication Security (CCS), pp. 62-72, Washington DC, October 2003.

[18] D. Liu and P. Ning, "Location-based pairwise key establishments for static sensor networks," Cyber Defense Laboratory, Department of Computer Science, North Carolina State University, 2003

[19] *Documentation for the Bytronic Process Control Unit version 3*, Bytronic International Ltd., 2002.

[20] F. Xiujun, W. Fan, T. Bihua and L. Yuanan, "Wireless sensor networks security analysis," School of Telecommunication Engineering, Beijing University of Posts and Telecommunications, Beijing, 2007 URL http://www.paper.edu.cn/downloadpaper.php? serial_number=200704-597

Performance, survivability, and cost aspects of Business Continuity Processes According to BS 25999

Wolfgang Boehmer*
*Technische Universitaet Darmstadt, Germany, Hochschulstr. 10, 64289 Darmstadt
Email: wboehmer@cdc.informatik.tu-darmstadt.de

***Abstract*—A new model is presented for evaluating the performance of a Business Continuity Management System according to BS 25999. Performance is based fundamentally on the system's Business Continuity Plans and Disaster Recovery Plans. Typically, the performance of these plans is inadequately evaluated using a number of specific exercises at various intervals and, in many cases, with a variety of targets. Consequently, it is difficult for companies to give *ex-ante* statements of their survival in the case of a disaster.**

Two key performance indicators are presented that allow the performance of a Business Continuity Management System to be evaluated according to BS 25999. Using these key performance indicators, the probability of survival can be estimated before extreme events occur. However, the two key performance indicators compete and their use invokes a trade-off: an alignment in favor of one key performance indicator is necessarily done at the expense of the other. A key performance matrix with four ranges is presented according to the Business Continuity Management System. The best range is the strategic balance in which both key performance indicators support the economic strategy and a suitable cost/benefit relationship is achieved. Moreover, if a company is already in the range of the strategic balance, a further improvement, which yields minimal turnover, may be possible. This improvement can be obtained via a combinatorial optimization between the two competing key performance indicators.

***Index Terms*—BS 25999; BCMS; Business Continuity Plan (BCP); Knapsack-Problem; Branch & Bounding.**

I. Introduction

This contribution is inspired by [1] which addresses measurement of performance indicators for effectiveness and economic efficiency of a Business Continuity Management System (BCMS). However, in this article the evolution of key performance indicators is interpreted as a trade-off between the conflicting goals of effectiveness and economic efficiency. This trade-off is analogous to a 0-1 knapsack problem. Furthermore, it is proposed that a management system (BCMS) can be interpreted as a control loop with a steady state based on systems theory for Discrete Event Systems (DEVS). If this interpretation is acceptable, then DEVS knowledge can be transferred to a management system.

The BS 25999-1:2006 standard sets out the code of practice for a BCMS [2]. After extensive review by the British Standard Institution (BSI), specifications for Business Continuity Management (BCM), BS 25999-2:2007, were published in November 2007 [3]. During that review, more than 5000 industrial ideas and suggestions were integrated into the standard, thereby establishing a high level of maturity. The standard BS 25999-2:2007 provides requirements that a management system must meet if critical business processes (value chain) are to remain stable in the face of acceptable levels of disasters. The fundamental idea is that BCM aims to manage various types of uncommon business risks that would have a huge impact on a company. A BCMS is capable of responding satisfactorily in extreme situations (catastrophic events) with pre-defined plans (Business Continuity Plans; BCP). The continuation of the value chain at an acceptable level for a defined period (Δt) is then ensured. A BCMS includes vital business processes. Recovering only the working infrastructure, e.g., replacing a failed IT infrastructure by an emergency one, will not meet a BCMS, as an IBM report clearly points out [4].

The BS 25999 standard requires implementation of a management system in accordance with the PDCA cycle (Plan–Do–Check–Act) as well as those systems already required in other standards, such as ISO/IEC 27001, ISO/IEC 20000, ISO/IEC 9001, and ISO/IEC14001. However, those standards describe only *what* to do rather than *how* to do it.

The PDCA[1] cycle is based on the idea of imperfection and thus follows a continuous improvement process. For example, in the Check phase, managers examine whether the plan's objective set is still in agreement with the rest of the system. If it is not, corrections are implemented in the Act stage.

During the initial Plan phase of a PDCA cycle, BS 25999 requires identification of critical business processes and analysis of dependencies between key stakeholders and key services. Following this, a risk analysis must be performed. For each risk of high impact and low probability, a response [i.e., a Business Continuity Plan (BCP)] must be developed. The response is aimed, on the one hand, at continuing business processes on a defined level (BCPs) and, on the other, at initiating those countermeasures that will restore the original state (Disaster Recovery Plan, DRP).

As in the ISO 27001 standard, risk plays a central role in BS 25999 [5][3]. However, the measures for implementing ISO 27001 are oriented toward risk-prevention, while those for BS 25999 (BCP and DRP) are reactive. That is, BCM is a reactive model that becomes active only after a disaster has occurred.

[1]The PDCA cycle was developed by W. E. Deming in the late 60s of the last century.

In this context, the maximum allowable downtime (Maximum Tolerable Period of Disruption; MTPD), which starts after a disaster occurs, increases considerably in importance. The MTPD is determined from the length of time that critical activities in the value chain require to begin working again after a disaster. This period of time is an ultimate boundary for a company and decides the company's survival. If this ultimate limit is exceeded, the company is irretrievably lost. Consequently, the goal of every company must be to optimize the performance of the restoration so that the time required for restoration (Recovery Time of Objective; RTO) is reduced. Thus, a company must do everything to ensure that RTO $\leq$ MTPD can be achieved. However, the efficiency of restoration measures must not be ignored.

The relation between critical activities and the value chain is determined by the Business Impact Analysis (BIA). Within the BIA, the dependent critical resources (key stakeholders, key products, key services) and their importance to critical activities (core processes of the value chain) are analyzed. Any BCMS includes those business processes that are vital. A BCMS is capable of responding satisfactorily in extreme situations (catastrophic events) with pre-defined plans and emergency processes (Business Continuity Processes, BCP). This raises the question as to how well the performance of emergency processes are. However, performance, as in CobiT and in ISO/IEC 9004:2009, is also understood here [6].

In the literature, three basic methods are generally available for measuring the performance of processes.

1) It one method, performance is related to the maturity of processes, and tools such as Spice (ISO/IEC 15504) or CMMI, developed at Carnegie Mellon University, are used to measure process maturity. This maturity approach is gaining support across a wide range of technical environments, including production environments as well as management systems such as ITIL (ISO/IEC 20000) and ISO/IEC 27001.
2) A somewhat newer method is to describe the state space of processes using process algebra. In the development carried out at the TU Eindhoven, classical process algebra was not used, but a tool (mcrl2) has been developed by which process algebra can be applied in a simple manner. Then, if modal logic is applied in the form of a μcalculus on the state-space, the process algebra generated by state sequences and state transitions of all processes allows safety issues and the liveness of processes to be analyzed. Essentially, model checking is performed based on a specification. First thoughts were published in [7] and a study of this method applied to a business continuity process (BCP) has been published [8]. An extensive explanation can be found in the Technical Report of the TU Eindhoven [9]. Note that this method is distinct from symbolic model verification (SMV) used in such tools as νSMV and SMV.
3) Another method is to estimate performance on the basis of appropriate indicators (KPI). The challenge is to define appropriate metrics, which have a corresponding significance. Suggestions for the handling of such indicators are to be found in [10] and in [11]. A performance measurement system for a BCMS is developed in [12]. It rests upon the methodology of the BORIS, which contains a set of different tools. In that article, traditional security variables (integrity, availability, confidentiality) and business indicators, appear to be necessary [12]. A similar approach to a performance measurement system has been applied in [1] for business continuity management (BCM) and a forerunner of this approach was published in [13].

In a previous article by Boehmer, it was demonstrated how the management system of ISO 27001 can be measured using effectiveness and efficiency as indicators [13]. As mentioned above, a measurement takes place in the Check phase. In the present paper, this idea of measuring the quality of these two KPIs is applied to a BCMS. Measurements of these KPIs provide the status of a BCMS; that status maps into one of four quadrants. The worst state is one in which a BCMS is neither effective nor efficient; this is called a strategic dilemma [13]. In a strategic dilemma, the probability of a catastrophe occurring in which the company will not survive is very high. Conversely, the survival probability increases if the ratio of the effectiveness and efficiency of the KPI is ideal and the majority of all the exercises carried out has $RTO \leq MTPD$.

This paper is divided into seven sections. The following section integrates relevant work from the literature. The third section contains a discussion of the structure of a process-oriented evaluation system based on circumstantial evidence and key indicators. In the fourth section, the development of two KPIs is discussed; these KPIs are used in the fifth section to look at survival probability. Survival probability is closely linked to a functioning BCP. In the sixth section trade-offs between the KPIs of effectiveness and efficiency are discussed within the context of the 0-1 knapsack problem. Our contribution finishes with a brief summary and prospects for future work.

II. Related Work

An empirical study by Knight and Pretty shows that those companies with a BCMS are more likely to survive a disaster than those that have taken no precautions [14]. Nevertheless, the study also shows that, despite the use of a BCMS, a company's chance of survival is not guaranteed, and a small number of such companies have been reported as failing to survive. Conversely, the study also reports a very small number of companies that survived a disaster even though they had no BCMS [14]. This latter phenomenon may simply be luck.

Looking at those cases of companies that used a BCMS and still did not survive, it appears the quality of their BCMS or BCP and DRP needs to be taken into account. It is clear from the study that the application of a standard is not, by itself, enough: the standard must be applied properly.

The literature has so far focused on the topic of BCMS primarily in practical terms, e.g., [15],[16],[17]. Nemzow discusses the need for various strategies to protect an organization

from natural and manmade disasters [15]. He also explains the difference between a BCP and a DRP. Quirchmayr discusses the Business Continuity Management Lifecycle and its content [16]. Landry and Koger discuss the lessons learned from 2005's hurricane Katrina [17]. Again, the importance of a DRP is stressed. Similar ideas are set out in the study by IBM on hurricane Katrina, including claims that a BCP and DRP must contain more than simple aspects of the company. For example, company members remaining in the disaster area should also be taken into account in the BCP [4]. Similarly, Saleem et al. [18] note the importance of an adequate Business Continuity Information Network on an effective DRP. A similar issue is also highlighted by Shklovski et al. [19]. The importance of a Business Impact Analysis (BIA) and the restoration point of objective after a disaster is discussed by Quirchmayr et al. [20]. These issues are related to the MTPD. Meanwhile, many of these aspects influenced the BS 25999-2:2007.

However, solely from the results of Knight and Pretty, a more detailed review can be posited [14]. This review must relate to a BCMS as well as to the function and performance of its BCP and DRP. Only after the quality of performance has been measured can a statement be made concerning a business's survival probability.

III. Business Continuity Managment System (BCMS)

The fundamental idea of a BCMS is that Business Continuity Management (BCM) is meant to manage those rare business risks that can have a huge impact on a company. The BCMS is capable of responding adequately in extreme situations (catastrophic events) with pre-defined plans (BCP). In the next section, the goal of a management system is discussed using a simple example, then this goal is transferred to a BCMS.

A. Concrete example–a weight management system

A simple real-world example is used here to illustrate the concept of a management system. Consider a person who wants to manage his or her weight using a management system that focusses on consumed and burned calories. A possible objective could be to balance these values, as illustrated in Figure 1. Another objective could be to reduce the weight of a person. In this case more calories must be burned than consumed. The measuring instrument is the weighing machine (scales). The ideal state is maintained by burning as many calories as are consumed. Then the system is in a dynamic equilibrium and energetic costs are balanced. Equilibrium is a state of a system that does not change with respect to one or more state variables over some period of time; i.e., on average, the weight remains constant over a long period.

In Figure 2 the weight management system is interpreted as a feedback system. Every time the person diverges from the ideal weight, an adjustment is made by the actuator (calories are burned). This behavior of the weight management system can be interpreted as a linear feedback system. This linear feedback system can be described with a Discrete Event

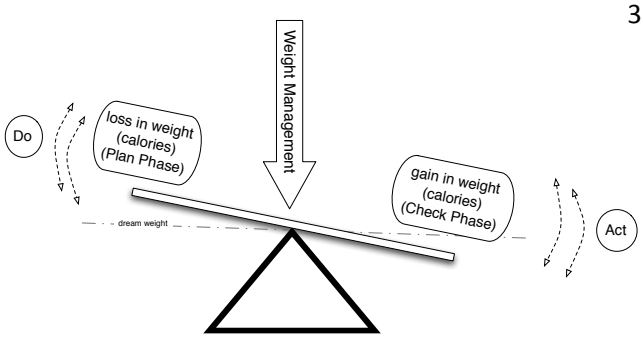


Fig. 1: Weight management system attempts to control gains and losses in a person's weight

System Specification (DEVS); it is a control loop that contains sensors (s), controller (c) and actuators (a) arranged to regulate in discrete (k)-steps a process variable (p) with respect to a reference signal $w(k)$ (see Figure 2).

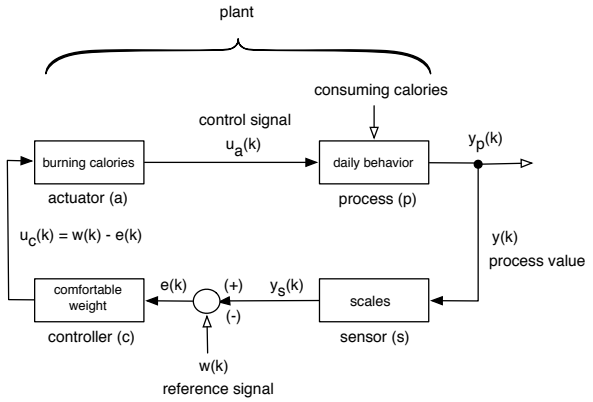


Fig. 2: Weight management system reinterpreted as a control loop for the weight of a person

The basic objective is to control the value of the process variable $y_p(k)$. This is done by measuring its value $y(k)$ and determining $e(k)$ its variance $(+/-)$ relative to the desired reference value $w(k)$. This variance $u_c = w(k) - e(k)$ is used by the actuator to generate an appropriate correcting control signal $u_a(k)$ that modifies the system's behavior and changes the value of $y_p(k)$ appropriately. A closed loop is created by the feedback of the controlled variable to the sensor and its conversion to a control signal, as in Figure 2.

Inside an atomic DEVS, an arbitrary formalism can be used. A DEVS can be viewed as a framework that unifies a number of other formalisms in a consistent, systems theoretic, state-centered fashion. Discrete Event System Specifications (DEVS) are dynamic systems whose state changes serve as a basis for discrete events.

A similar behavior is achieved through the PDCA cycle in a Business Continuity Management System (BCMS). As mentioned above, a PDCA cycle is based on imperfection and follows a continuous improvement process. The controlled

variables are the KPIs related to the effectiveness and efficiency of a business continuity process (BCP). The reference signal is the balance (equilibrium) between effectiveness and efficiency of each business continuity process (BCP) and each Diaster Recovery Process (DRP).

The PDCA cycle can be applied to each element of the BCMS; this results in a PDCA cycle for the BCP and DRP as well as for the BCMS itself. In Figure 3 a PDCA cycle is

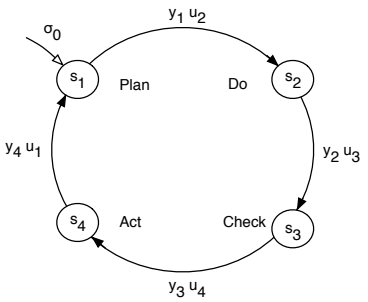


Fig. 3: PDCA cycle as a DEVS

modeled as a discrete deterministic finite automaton ($\mathcal{A}$). In this representation, the finite automaton can be defined as a 6-tuple,

$$\mathcal{A}(S, U, Y, \delta, g, s(0)) \tag{1}$$

Three finite sets occur:

$$S = \{s_1, ..., s_n\}\,; \text{ set of states} \tag{2}$$

$$U = \{u_1, ..., u_n\}\,; \text{ set of input alphabet} \tag{3}$$

$$Y = \{y_1, ..., y_n\}\,; \text{ set of output alphabet} \tag{4}$$

with two functions:

$$\delta : S \times U \longrightarrow S\,; \text{ transition function} \tag{5}$$

$$g : S \times U \longrightarrow Y; \text{ output function} \tag{6}$$

Furthermore, the initial state is called $s(0) \in S$. A single state is determined by $s \in S$, and its successor state s' is formed with the help of a transition function δ by $s' = \delta(s, u)$.

The four states in Figure 3 can be identified in accordance with BS 25999:

s_1= establishing and managing
s_2= implementing and operating
s_3= monitoring and reviewing
s_4= maintaining and improving

The state transition function is δ, k is the time independent counter, and g is the output function[2]. The automaton equations are then

$$s(k+1) = \delta(s(k), u(k))\ k = 0, 1, ... \tag{7}$$

$$y(k) = g(s(k), u(k))\ k = 0, 1, ... \tag{8}$$

Therefore an automaton ($\mathcal{A}$) is generated by an infinite state sequence and modeled by the continuous improvement of the PDCA cycle. If, after a certain time, alterations in the state no longer occur, so that a state change $(k+1)$ leaves the system in the old state with $s = y(s)$, then the state is an equilibrium one. This equilibrium condition expresses the balance between effectiveness and efficiency in the events of a BCMS for a BCP and DRP. In this case δ' is an extended state transition function for all absorbing states[3]. This condition is called a state of equilibrium [21].

Therefore, a BCMS with inherent PDCA cycles can be described with the system theory of discrete-event systems. States in the PDCA cycle for a BCP and DRP are measured by two key indicators, Efk and Efz. It is important to distinguish between an indicator and a key indicator. In the next section we show how this approach can be mapped onto the concept of a Business Continuity Management System.

B. Basic idea of a BCMS according BS 25999

A company that wants to safeguard its critical value chain should focus on securing revenues by taking adequate risk countermeasures. Since 2007, the BS 25999-2:2007 [3], published by the British Standard Institution (BSI), is available. It is an industry-wide recognized best-practice method that governs the creation of a BCMS. It encompasses a BCP and a DRP (Disaster Recovery Plan). The standard requires implementation of a management system in accordance with the PDCA cycle (Plan-Do-Check-Act), as well as those already required in standards ISO 27001, ISO 20000, and others.

Figure 4 illustrates the operational view of a PDCA cycle within an underlying BCMS. A BCMS is a framework that helps balance risks (potential disasters and impacts on the critical business process) against available countermeasures (business continuity processes and business recovery processes) while recognizing the MTPD as a real-world side constraint.

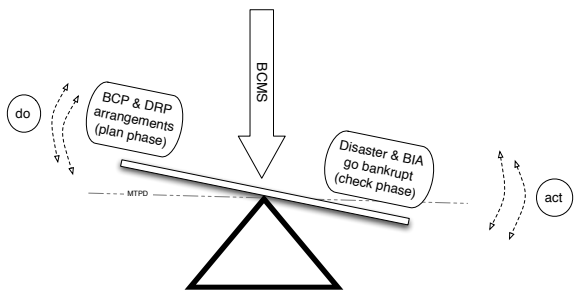


Fig. 4: Business Continuity Management System (BCMS) with ups and downs as a seesaw

Figure 5 shows a qualitative timeline of events following a disaster that strikes at time t_0. Immediately after occurrence of the disaster, turnover collapses. At time t_1 the processes of the BCP (emergency operation) begin, and turnover starts to increase. A little later, at time t_2, recovery processes start, and

[2]For more details we refer to the literature [21]

[3]Absorbing states are states that do not have successor states, and can be considered as final states.

at time t_4 the company is back to its normal level of operation. The dash dotted line in the figure shows the increase in costs after the disaster. In the event that no countermeasures (BCP, DRP) are taken, or that the countermeasures do not work, the costs continue to increase (see curve 2). The ideal situation is that the Business Continuity Plan and the Disaster Recovery Plan work so well that costs remain bounded, as in curve 1.

If no action (BCP, DRP) has been taken at or before the time t_3 in Figure 5, then costs will increase until insolvency is reached. Costs are determined by the obligations of the company. These consist of personnel, technical expenses, and the cost of delivery, performance, or possibly storage costs, etc. Thus t_3 identifies the maximum allowable downtime (Maximum Tolerable Period of Disruption; MTPD), $\Delta T_{max} = t_0 - t_3$.

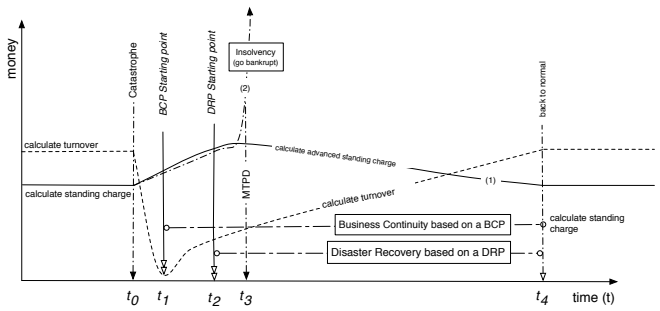


Fig. 5: Illustration of aspects of a catastrophe (t_0) and the reaction ($t_1, ..., t_4$)

It is in the self interest of a company to keep the BCPs and DRPs operational. Usually, this is tested on a regular basis by simulating that something goes astray within the ordinary business process. Because these tests are expensive, they are not executed very often and generally they only address certain aspects of the recovery plans. Such testing provides a rather haphazard prediction of the effectiveness of recovery plans when a true disaster strikes. To improve the quality of the analysis of BCPs and DRPs, one should model these and the ordinary business process such that they can be simulated. The first ideas of how to do this have been presented in [7].

IV. Performance Indicator of a BCMS according to BS 25999

This section shows how the key indicators of effectiveness and economic efficiency are developed. The controlled variables are the KPIs related to the effectiveness and efficiency of a business continuity process (BCP). The reference signal is the balance (equilibrium) between effectiveness and efficiency of each business continuity process (BCP) and each Diaster Recovery Process (DRP). In Figure 6 we see how a BCMS acts as a control loop, but to measure the performance of a BCP and a DRP, a number of indicators are required.

A number of indicators will be formed for each key indicator. An indicator and a key indicator can be defined as follows:

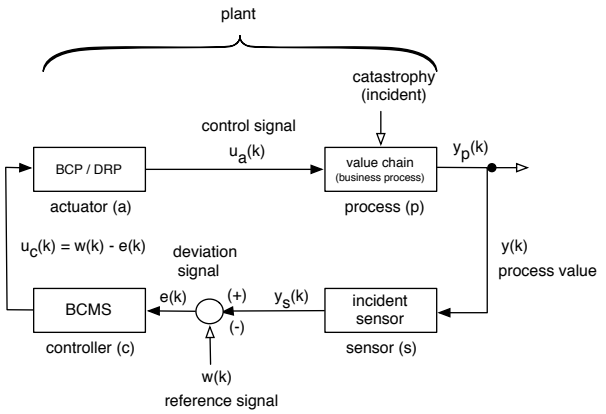


Fig. 6: BCMS Control loop for the emergency processes

Def. 1: An indicator (I) is a variable subject to a metric.

Def. 2: A Key Performance Indicator (KPI) is a key indicator formed from several more general indicators and provides a significant statement about a certain set of circumstances (see Eq. 18 and Eq. 22).

It is possible to make a significant statement using a key indicator, but this statement is supported by several more general indicators. The quality of a BCMS is reflected in the preparation, handling, and testing of the BCP and DRP in the Check phase (see Figure 4). For the system's effectiveness, this means that the indicator's

- existence (I_{ex}),
- enforcement (I_{op}) and
- completeness (I_{co})

form a set on the system effectiveness (Efk):

$$Efk = \{I_{ex}, I_{op_{(BCP,DRP)}}, I_{co}\}. \quad (9)$$

These indicators are derived about $\lambda_1, ..., \lambda_4$ from the pyramid-level documents (see Figure 7). This pyramid structure was derived by Alan Calder from practical experience and published in the ISMS Toolkit [22].

For the assessment system, performance values (KPI) can be defined for a BCMS. The documentation required by the standard plays a crucial role. From the required documentation, success measurements can be derived, and a lower boundary can be defined for the implementation of a BCMS. Below this boundary, a BCMS is inadequately implemented, and the effectiveness *(are we doing the right things?)* cannot be measured. Furthermore, an upper boundary is defined by the economic efficiency of the BCMS *(are we doing things right?)*. This consists of a cost/benefit relationship and follows the standard requirement (Clause 2.1.4 of the standard). This limit postulates that no more than the value of the critical business process should be invested in countermeasures.

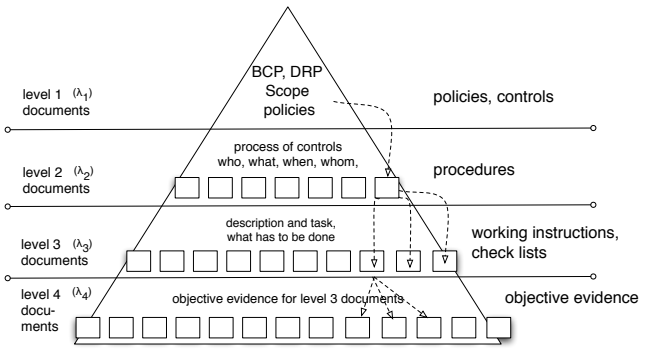


Fig. 7: Pyramid-level documents of a BCP and DRP

Figure 7 shows how the volume of documents from the top (λ_1) (peak) down increase. This structure shows the natural history based on a directive toward their technical implementation (procedures (λ_2), checklists (λ_3)), which provides a series of activities for implementing the directive. At the lowest level (λ_4) is the evidence *(objective evidence*), as described by Alan Calder [22]. This pyramid structure is now a condition for the existence of a lower boundary, as recommended in [13]. Below this boundary, the implementation of the management systems is not measurable. If the lower limit is exceeded, the quality of the BCMS and BRP and DRP can be measured on the basis of indicators.

The first key performance indicator (KPI_1) relates to the effectiveness (see Eq. 9) and can be determined by three indicators. On the one hand, the existence of the policies per BCP (Business Continuity Plan) can be evaluated with indicator I_{ex}. On the other hand, the degree of enforcement of policies is considered using indicator I_{op} relative to the BCP and DRP. Completeness (coverage) will be used as the third indicator, I_{co}. This indicates the coverage of the BIA as compared with the resources in relation to the scope of the BCMS.

The indicator I_{ex} evaluates the existence of control points (checkpoints; CP) or non-existent control points (NoCP) relative to a BCMS, according to BS 25999. The clauses of BS 25999 applied in the BCMS should be proven with control points; otherwise, no statement can be made about implementation of the standards. This case of the existence or non-existence of control points per document level ($\lambda_1, ..., \lambda_4$) can be expressed as

$$I_{ex} = \frac{\sum_{i=1}^{n} CP_{\lambda l} - \sum_{j=1}^{m} NoCP_{j(BCP)}}{\sum_{i=1}^{n} CP_{\lambda l}} \tag{10}$$

Thus, the indicator of control points I_{ex} is on the range between 0 and 1:

$$I_{ex} = \begin{cases} 1, & if\ NoCP = 0 \\ 0, & if\ \forall\ CP_{\lambda l} = 0; l = \{1,2,3,4\} \\ otherwise. \end{cases} \tag{11}$$

For ideal implementation of each standard in a business, the indicator should satisfy $I_{ex} \approx 1$ for each standard. This means that there are no deviations ($NoCP \approx 0$) between the control points (clauses) of the standards and the actual existing control points. When $I_{ex} \ll 1$, too few of the standard clauses have been applied and optimization is needed.

The existence of policies says little about whether they are actually present or whether they exist only on paper. Thus, Eq. 11 is a necessary but insufficient condition. This is precisely where the indicator of the degree of enforcement ($I_{op_{(BCP)}}, I_{op_{(DRP)}}$) is applied.

The indicator of the degree of enforcement ($I_{op_{(BCP)}}$) is based on the result of BCP Assessments, practical exercises, and deviations from the planned controls. For a BCP, the nonexistent measures ($NoC_{j(BCP)}$) are related to the necessary measures ($C_{\lambda l(BCP)}$) relative to the pyramid-level documents (see Figure 7). Whether adequate controls for a particular risk scenario are available for the continuation of critical business processes is determined. For each identified risk to critical business processes, there is a BCP and DRP. Here, the risk scenarios could be completely different. For example, a BCP and DRP for the risk of a pandemic scenario looks quite different than a scenario for the risk that a major supplier *(key stakeholder)* fails unexpectedly.

The indicator of the degree of enforcement ($I_{op_{(BCP)}}$) per document level (Eq. 12) checks the extent of discrepancies in the assessments between the action in BCP ($C_{\lambda l(BCP)}$) and the actual sequence ($NoC_{j(BCP)}$) in an exercise,

$$I_{op_{(BCP)}} = \frac{\sum_{i=1}^{n} C_{\lambda l(BCP)} - \sum_{j=1}^{m} NoC_{j(BCP)}}{\sum_{i=1}^{n} C_{\lambda l(BCP)}} \tag{12}$$

Thus, the indicator of the control points $I_{op_{(BCP)}}$ is on the range between 0 and 1 and is analogous to Eq. 11,

$$I_{op_{(BCP)}} = \begin{cases} 1, & if\ NoC_{(BCP)} = 0 \\ 0, & if\ \forall\ C_{\lambda l(BCP)} = 0; l = \{1,2,3,4\} \\ otherwise. \end{cases} \tag{13}$$

The BCP and DRP are closely linked to the standard but must be considered separately to allow for a granular approach. The indicator of the degree of enforcement ($I_{op_{(DRP)}}$) with relation to the DRP is based on the results from the assessments or exercises and the deviations ($NoC_{j(DRP)}$) of the proposed DRP ($C_{\lambda l(DRP)}$) controls,

$$I_{op_{(DRP)}} = \frac{\sum_{i=1}^{n} C_{\lambda l(DRP)} - \sum_{j=1}^{m} NoC_{j(DRP)}}{\sum_{i=1}^{n} C_{\lambda l(DRP)}} \tag{14}$$

Thus, the indicator of the control points $I_{op_{(DRP)}}$ is on the range between 0 and 1 and is analogous to Eq. 11. This indicator assesses the difference between planned activities and actual exercises,

$$I_{op_{(DRP)}} = \begin{cases} 1, & if\ NoC_{(DRP)} = 0 \\ 0, & if\ \forall\ C_{\lambda l(DRP)} = 0; l = \{1,2,3,4\} \\ otherwise. \end{cases} \tag{15}$$

Equation 15 ensures that the value of the practical experience gained during exercises for disaster recovery is recognized.

Key to effectiveness is the question of whether in fact all critical business processes in terms of resources have been considered with a BIA in relation to the scope of the BCMS. This observation is carried out using the indicator to assess coverage. The indicator (I_{co}) of the coverage of a BIA in relation to resources (key products, stakeholders, etc) within the scope leads to

$$I_{co} = \frac{\sum_{i=1}^{n} Res_{i(BIA)} - \sum_{j=1}^{m} Res_{j(NoSP)}}{\sum_{i=1}^{n} Res_{i(BIA)}} \tag{16}$$

Equation 16 places the critical resources (Res) within the BIA that must be treated with non-existing policies ($NoSP$) in relation to resources,

$$I_{co} = \begin{cases} 1, & if\ Res_{(NoSP)} = 0 \\ 0, & if\ \forall\ Res_{(BIA)} = 0 \\ otherwise. \end{cases} \tag{17}$$

Thus, the indicator (I_{co}) is on the range between 0 and 1 and is analogous to Eq. 11. The fewer the number of analyses that are present (BIA) for the critical resources, the smaller the coverage of the $I_{co} \ll 1$ critical processes, and the lower the effectiveness.

Finally, the indicators of effectiveness can be calculated with

$$Efk = I_{ex} \times I_{op_{(BCP)}} \times I_{op_{(DRP)}} \times I_{co} \tag{18}$$

This indicator (Efk) fluctuates between 0 and 1 and represents a point in a specific space spanned by the indicators. This key indicator says something about the effectiveness of the BCMS and the quality of the BCP and DRP. It provides a significant statement about a situation on the basis of the underlying indicators. Furthermore, Efk satisfies Def. 2 and is a key performance indicator for a company.

If the indicator is determined by numerous exercises and at a regular time interval t_0 and t_3 (see Figure 5), a conclusion may be drawn about the likelihood of survival in the event of a disaster. This aspect is discussed in the next section.

V. Estimation of the Survivability of a Business

In this section, the survival probability of a business is discussed. It is assumed that the business has implemented a BCMS in accordance with BS 25999 and that the indicators of effectiveness Efk and economic efficiency Efz have been identified. However, when economic efficiency is considered in advance (preventive or reactive controls) of a balance of controls, the indicator Efz is not used to consider the likelihood of survival.

After a disaster, the likelihood of survival of an enterprise is determined by the ratio of effectiveness. The effectiveness (Efk) can be understood as a random variable X in the interval *(a,b)* (see Figure 8). Figure 8 shows only the part between t_0 and t_3 (cf. Figure 5). Here, *(a)* can be identified as the entry point at the time of a disaster and then *(b)* is the date defined by the MTPD. Figure 8 relates the interval *(a,b)* to time ($a = t_0, b = t_3$). If the two markers *(a=1, b=0)* are set, the result of (x) lies in this interval if the exercises (assessments) of the BCP and DRP are used and an exercise gives a result of (x). If ($x = 1$) in the ideal case, then (t_0) and (t_1) almost coincide and the starting point of the BCP is immediately after the occurrence of the disaster. The reverse is also true: the smaller ($x \ll 1$) is, the longer before time (t_1) occurs, and the later the starting point of the BCP. If ($t_1 \geq t_3 = MTPD$), the business is irretrievable.

If there are enough exercises and assessments of the BCP and DRP, so that the effectiveness (Efk) can be measured and projected onto the interval *(a,b)*, the probability $P(a \leq X \leq b)$ for the interval $a \leq X \leq b$ can be given, where X takes on a value from the interval. Then, the likelihood function of the random variable X is known. Thus, the distribution function $F(x) = P(X \leq x)$ can be determined. A distribution function of something like $F(x) = x^{-1}$ would be ideal for a business, because then the majority of the exercise results are in the interval (a, b) between 1 and 0.5. This is the case represented by the curve Efk_I in Figure 8.

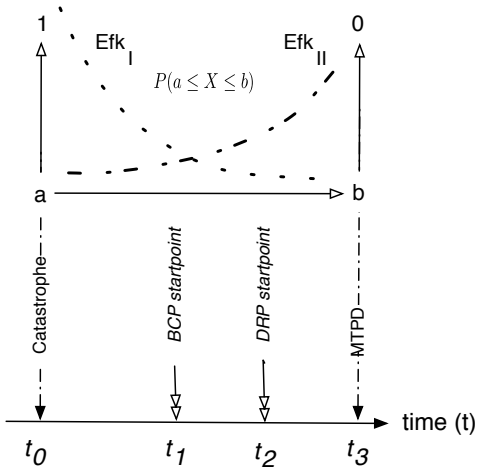


Fig. 8: Efk as a random variable within the interval a,b

In contrast, the curve Efk_{II} in Figure 8 represents an unfavorable curve for the indicator of effectiveness. In this case, the majority of the exercises are near the MTPD, i.e., near time t_3. Businesses that have displayed such an unfavorable course of effectiveness are not adequately equipped for a disaster and can probably survive only because of fortunate circumstances. This conclusion is in agreement with the empirical studies by Knight and Pretty [14].

Therefore, the closer a business's exercise results are to $x = 1$, the higher the probability that this business will survive a catastrophic event. However, these statements are valid only when plans such as BCP and DRP already exist when the disaster occurs and when these plans have been enacted, practiced, etc. Otherwise, measurements of indicators and key indicators–if no BCP or DRP is available–are meaningless. In that case, the cost curve is similar to curve 2 in Figure 5. Thus, an *ex-ante* statement would be possible only if sufficient information is available. Sufficient information is available if enough exercises in the BCP and DRP have been carried out.

The advantage to this method lies in the structured analysis of indicators and key indicators. This can also inform a board of management as to how a company is likely to respond to a disaster.

A. Key performance indicator of economic efficiency

The literature discussing cost considerations with regard to the security of information is controversial. A number of articles classify the calculation of the expenditure for security countermeasures in a Return of Security Investment (ROSI), often involving (perimeter) defensive techniques [23], [24], [25], [26]. A possible profit-loss of the organisation is confronted with the protection of IT assets and the costs of a successful attack are weighed against the security costs (countermeasures).

Other considerations in the literature deal with profit-loss as a loss of productivity; e.g., if a file server becomes unavailable, productivity declines because a number of employees become incapable of working [27]. Analysis of such problems are confounded because suitable material for a benchmark still does not exist [27].

Considerations of the profit-loss are aimed at increases in operating expenses and at influences on business processes. These impact economic efficiency. However, when considered in isolation, security costs represent only part of the economic efficiency of an BCMS.

Elsewhere it is argued that a cost consideration could not be successful with the ROSI model [28]. Further, it has been suggested that companies often apply a fear, uncertainty, and doubt (FUD) strategy for investments in security countermeasures [29]. A good overview of different approaches to the ROSI model is available in [25].

Note that the above approaches include neither indirect costs nor operating expenses in cost evaluations. In addition, direct costs are only partially taken into account. From the point of view of critical processes of an BCMS, the above approaches consider only partial aspects. With the efficiency of an BCMS, the focus is on the economic aspects of the BCPs and DRPs. For each rare risk that would have an huge impact on the value chain there must exist a BCP and DRP as well as all the processes and documents listed in Figure 7. Therefore, economic efficiency is to be thought of, in principle, as a cost/benefit relationship. To successfully plan the budget for the critical processes of an enterprise, the costs of all BCPs and DRPs must be considered.

A Total-Cost-of-Ownership (TCO) model provides an adequate look at the costs [30]. In the TCO model, three cost drivers are identified: direct costs (D_C), indirect costs (I_C), and operating expenses (O_C). At first glance, the TCO model seems to be sufficient for the interests of an BCMS when considering infrastructure costs. The three cost categories can be defined as follows:

- Sum of direct costs ($\sum_{i=1}^{n} D_{C_i}$): Employees, hardware, software, external services, physical environments (buildings, etc.) in which data processing should take place under secure conditions for an organisation. Moreover, in addition to the acquisition costs of the devices (security appliances), their depreciation also has to be calculated.
- Sum of operating expenses ($\sum_{j=1}^{m} O_{C_j}$): Costs that must be considered when calculating the maintenance, servicing, and repair of the components listed as direct costs above.
- Sum of indirect costs ($\sum_{k=1}^{p} I_{C_k}$): These expenses originate as a result of unproductive time from the end user.

The general TCO model would have to be adapted to the scope of an BCMS, that is, to the critical processes. In addition, the TCO model should not be of a static nature; instead, in the interest of increasing efficiency, it should be subject to a Deming cycle in accordance with ISO 9001.

As a modification, the TCO model, referencing a fiscal year, e.g., Fy_0 at t_0, could calculate the costs based on the infrastructure controls of the critical business processes of an BCMS. With this, the infrastructure costs of a BCMS in a fiscal year can be expressed as follows:

$$Fy_0 = \sum_{i=1}^{n} D_{C_i} + \sum_{j=1}^{m} I_{C_j} + \sum_{k=1}^{p} O_{C_k} \tag{19}$$

Then we can calculate a change (Iteration) from one fiscal year (Fy_0) at time t_0 to the next fiscal year (Fy_1) at time t_1. Besides the infrastructure costs, the expenses for the BCPs and DRPs need to be considered. An essential benefit of a BCMS is that it aims to establish a connection between cost and the recognized rare risks. In trying to define the economic efficiency of the risk defence with a BCP and DRP, a series of questions arise: According to Figure 7, the whole costs for all BCPs exercises (BCP_{costs}) and DRPs exercises (DRP_{costs}) for a BCMS can be derived from the pyramid carried out in one fiscal year, e.g. (Fy_0). This management is strictly carried out according to economic conditions. If, in the next fiscal year, a BCP/DRP exercise is again carried out at the time Fy_1, an optimization must have be done in between, because

- The processes for a BCP or a DRP can be optimized.
- The processes and controls, procedures, checklists can be optimized.
- The expenses for transferring the risks have changed (increased, decreased).
- In different (that is, in more then one) BCP and DRP the same controls, procedures, and checklists can be used.

As a result, a possible difference arises for the whole exercise cost of $\sum BCP_{costs}$ and for the whole exercise cost of $\sum DRP_{costs}$, which can be explained by a change in the cost of dealing with the BCP and DRP exercises and the increasing experiences. This means that the cost for a control that is used for one BCP/DRP could differ from that for a control that is used in more than one BCP/DRP.

So, for the KPI of the efficiency (Efz_k), which can be understood as the economic component with reference to an interval (Δt), when we consider the difference ($\Delta F \geq 0 = Fy_0 - Fy_1$) between the total BCP/DRP expenses for two fiscal years, we obtain

$$Efz_i = \frac{\left(\sum_{i=1}^{n} BCP_{iCost} + Fy_0\right) - \left(\sum_{i=1}^{n} BCP_{iCost'} + Fy_1\right)}{\sum_{i=1}^{n} BCP_{iCost} + Fy_0} \tag{20}$$

$$Efz_j = \frac{\left(\sum_{j=1}^{n} DRP_{j_{Cost}} + Fy_0\right) - \left(\sum_{j=1}^{n} DRP_{j_{Cost'}} + Fy_1\right)}{\sum_{j=1}^{n} DRP_{j_{Cost}} + Fy_0} \tag{21}$$

Equations 20 and 21 show that $Efz_{i,j} \in \mathbb{R}$ could be either a positive or a negative indicator. Nevertheless, in Eqs. 20 and 21, it is postulated that in the fiscal year Fy_1, a smaller budget is required for rare risk defence than in fiscal year Fy_0. Therefore, the key indicator is typically positive. Otherwise, if a larger budget is allocated than in the previous year, a negative indicator results.

The second key performance indicator (KPI_2) is related to the efficiency (Efz) of a BCMS. As mentioned above, a BCMS is a reactive model; in contrast, the ISO/IEC 27001 standard requires preventive controls related to the possible risks. Both a BCMS and an Information Security Management System (ISMS) according to ISO 27001 have risk management as a central component.

Bass and Robichaux discuss the different forms of handling preventive, detective, and corrective controls in connection with a baseline assurance [31]. If the ideas of [31] are applied, the question arises as to which of the recognized potential risks require preventive or reactive (corrective) actions. The present paper posits that this is merely a question of cost: it does not involve technical or organizational issues. Risk management corresponds to cost management and we know that a Business Continuity Management System (BCMS) according to BS25999 contains risk management. A similar result is found in [32].

In the case of a BCMS, this means that the reactive controls of each BCP and each DRP are cheaper to use than the value of business processes (value chain), and they are as cost effective as potential preventive (P_{rev}) controls. Thus, a cost inequality arises. Over a fiscal year ($Fy0$), the inequality involves these four costs: the cost of a each BCP (BCP_{cost}) and each DRP (DRP_{cost}), the additional costs (Adv_{cost}), and the cost ($P_{rev-Control_{cost}}$) for preventive controls,

$$Efz = BCP_{cost} + DRP_{cost} + Adv_{cost} \ll P_{rev-Control_{cost}} \ll Rev_{Fy0} \tag{22}$$

Here (Rev) is the business profit. The inequality (22) does not display static behavior. It provides a boundary condition for an ISMS in accordance with ISO 27001 and for a BCMS in accordance with BS 25999; however, the boundary conditions are temporal and must be periodically reviewed. It may well be that a potential risk can be dealt with more cheaply using a preventive action rather than a corrective/reactive one.

As an example, consider a company that is known to be located in a flood zone or an earthquake zone (see Figure 9). According to an ISMS, a preventive action would be to move the company. In contrast, a BCMS (BCP, DRP) would initiate action only after flooding or an earthquake occurred. The costs in light of the probability of risk must be balanced against each other, and this is precisely the inequality that is described by Eq. 22.

The indicators of effectiveness and economic efficiency have been determined in this section. In the next section, using the indicator of effectiveness, the performance will be determined.

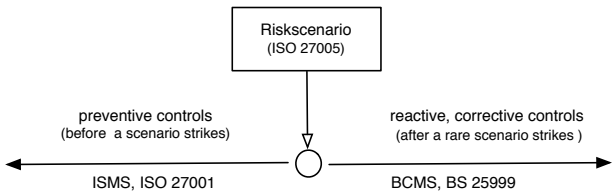


Fig. 9: Risk scenarios and the difference between ISMS and BCMS

B. Key performance matrix of effectiveness and efficiency

To determine the quality of a BCMS, the KPI of the effectiveness of a BCP must be placed in relation to the efficiency of a BCP. This takes into equal consideration both the efficiency (economic) and the effectiveness of a BCMS. These key indicators are two properties that should be kept strictly separated qualitatively and should not be aggregated into a single key indicator. The actual security countermeasures for critical business processes and their efficient realization can be shown in a matrix. Within the matrix, the KPIs of the effectiveness of the BCMS span one axis and the key indicators of efficiency span the other. The key performance indicators of effectiveness and of efficiency are bounded: $0 \leq Efk_k \leq 1$ and $-1 \leq Efz_k \leq 1$. The following can be defined as a first arbitrary linear approximation for effectiveness:

$$Efk_k = \begin{cases} yes & = & 0.5 < 1 \\ no & = & 0 \leq 0.5 \end{cases} \tag{23}$$

If the key indicator is above 0.5, the BCMS lies in the positive area (yes); if it is below 0.5, a (no) is assigned. A similar distinction can be defined for the key indicator of efficiency:

$$Efz_k = \begin{cases} yes & = & 0 < 1 \\ no & = & -1 \leq 0 \end{cases} \tag{24}$$

In principle all four possible combinations of Eq. 23 and Eq. 24 are observable; the four (a, b, c, d) are shown in Figure 10.

Case (a) can be described as an ideal state of a BCMS.

effective / efficient	yes	no
yes	a): BCMS is effective and efficient	b): BCMS is effective but not efficient
no	d): BCMS is not effective but efficient	c): BCMS is not effective nor efficient

Fig. 10: Performance matrix of an BCMS

a: *BCMS is effective and efficient*
This case can be defined as a strategic balance. Safeguarding critical business processes is in a strategic balance such that implementations of security controls are completely efficient. The BCMS supports the IT strategy efficiently with the right security controls, and the security controls are marked by an optimum cost/ benefit relationship.

In addition to the strategic balance, three kinds of imbalance exist for an IMS[4] [33]. Transferred onto a BCMS, the three correspond to the cases b, c, and d that appear in Figure 10.

b: *BCMS is effective but not efficient*
This situation corresponds to a strategic waste. The enterprise situation has high effectiveness due to the operation of an information security management system, but efficiency has not been achieved. In fact, in case (b), the achievement potential of an BCMS is effectively exhausted; however, exhaustion takes place uneconomically.

c: *BCMS is neither effective nor efficient*
This situation corresponds to a strategic dilemma. The operation of a BCMS and its achievement potential are neither effective nor efficient. Although considerable investments are expended in information security, the achievement potential is barely exhausted, and effective security countermeasures for critical business processes are not realized. Dissipation and waste of valuable resources exist.

d: *BCMS is not effective, but it is efficient*
This situation corresponds to a strategical dissipation. The efficiency of the BCMS is high, but its effectiveness is very low. The achievement potential of the BCMS is not properly recognised nor exhausted. Every control in information security is considered unique and, hence, is often misjudged.

If a performance $(Efk_k; Efz_k)$ measurement finds any imbalance (b, c, d), the BMCS must act as in Figure 11. The actuator initiates the check-and-act phases of the PDCA cycle so that corrective and preventive actions are performed. This process should continue until a balance between effectiveness and efficiency is attained, i.e., until case (a) is realized. Figure 11 shows this operation within a control loop according to a deterministic finite state machine.

Moreover, even if a company is already in the range of the strategic balance, further improvements may be possible, leading to minimal turnover. This improvement can be obtained via a combinatorial optimization between the KPI of effectiveness and the KPI of efficiency for each BCP. We present this idea in detail in the next section.

VI. Trade-off between effectiveness and efficiency

To perform a cost benefit analysis of information security, this article proposes two KPIs. For each KPI, suitable measurable indications are defined. The KPI of effectiveness and

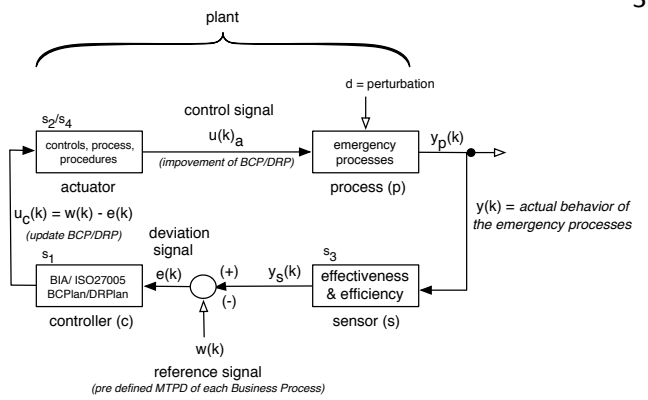


Fig. 11: Control loop for a BCP

the KPI of economic efficiency compete (Fig. 12), so that an alignment in favor of one KPI is necessarily done at the expense of the other. In [13], a key performance matrix with four ranges is presented according to the ISMS and in [34] a similar trade-off approach is presented for an ISMS.

The best range of values for the KPIs is the strategic balance in which the KPI of effectiveness and the KPI of efficiency support the economic strategy and achieve a suitable cost/benefit relationship. One of the main task of an BCMS with its PDCA-Cycle is to reach the strategic balance.

To optimize the BCP/DRP, requirements must be positioned so as to maximize effectiveness in the direction of a strategic balance. This means, for instance, that more exercises must be done for all working instructions, records, and policies structured according to Figure 7 procedures (objective evidence of policy enforcement). This would reduce the risk that the BCP/DRP was not working very well. However, this requirement would exceed the calculated budget. With regards to economic efficiency, one attempts to minimize the cost for each BCP/DRP with respect to investments so as to reduce turnover as little as possible.

Figure 12 shows the two KPIs like contrasting faces. The graphs are based on typical behaviour and we present a first approximation. The introduced budget limit of 30% is taken from the Ph.D. thesis from Soo Hoo on an empirically determined limit of investment [35]. This trade-off can be interpreted as a variation of the knapsack problem (KP). The knapsack problem is an integer combinatorial optimization problem and is $\mathcal{NP}-hard$. This means that a ROSI calculation has a complex solution.

This description of the 0-1 knapsack problem follows Martello[5] and Toth [36]. To use the approach from Martello and Toth for this trade-off, it is necessary to determine an optimum for the cost of each BCP/DRP with some certain controls (x) related to some certain policies (p) within the limited predefined investment [35]. In this 0-1 knapsack approach, we use for the controls $x_j(j = 1, ...n), n \in \mathbb{N}$, which could reduce one or more risks from the SoA through countermeasures

[4]IMS is the abbreviation for an information management system

[5]cf. Martello & Toth, page 1–5

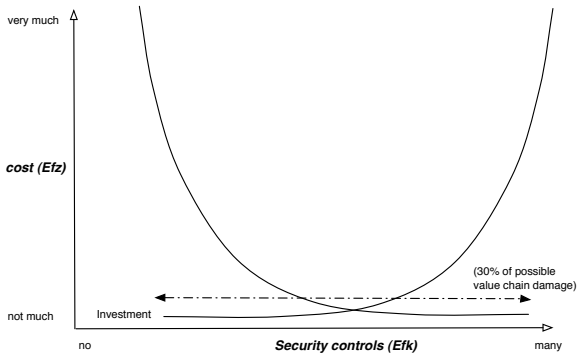


Fig. 12: Trade-off between Efz_k and Efk_k

(controls, x_j),

$$x_j = \begin{cases} 1 & = \text{ if control j is being used;} \\ 0 & = \text{ otherwise} \end{cases} \quad (25)$$

Furthermore, we use p_j for policies (P_{ol}) and w_j for the cost of each control x_j. Hence, a policy that is able to reduce more than one risk is more welcome; otherwise, it is better to mitigate a risk than to avoid it. Like Soo Hoo, we use c to describe the upper investment limit.

The definitions are as follows:

p_j: policy in terms of benefit from each control x_j,
w_j: cost for each control, which considers each BCP/DRP,
c: upper investment limit from Soo Hoo.

We expect that policies reduce more than one risk, in accordance with Eq. 19, so we try to optimize the function z,

$$maximize\ z = \sum_{j=1}^{n} = p_j x_j \quad (26)$$

We interpret p_j as a policy to confront risk; therefore, the value p_j for safeguarding the critical business process (cBP) will increase when p_j mitigates more than one risk under the side condition of w_j. Now, we attempt to figure out for which controls in (x) the following is valid:

$$\sum_{j=1}^{n} = w_j x_j \leq c \quad (27)$$

With Eqs. (25), (26), and (27), we can define a 0-1 knapsack problem. To solve the complexity of this 0-1 knapsack problem, this paper proposes a heuristic procedure. In Martello and Toth [36], different heuristic solution are discussed. In our contribution, the Branch-and-Bound (BB) procedure of the Horowitz-Sahni algorithm (HS) was chosen as a first approximation. The Branch and Bound procedures are essentially based on a problem branching and a limitation by means of lower and upper bounds for the subsets.

1) Branch: The basic principle of the Branch and Bound procedure is based on a minimization. A forward movement consists of inserting the largest possible set of new consecutive items into the current solution. A branching of the problem (P_0) is performed, yielding $k = 3$ subproblems $P_i\{1 = 1...k\}$, so that the following is an allowable solution for the subset (x_j):

$$x(P_0) = \bigcup_{i=1}^{k} x(P_j) \quad (28)$$

The three sub-problems can be thought of as the controls that are used in more than one BCP/DRP. The following subproblems then exist: $P_1(BCP_a)$, $P_2(BCP_b)$, $P_3(BCP_c)$.

2) Bound: Still, for each subset there are limitations, namely a lower bound (LB) and an upper bound (UB). If it is valid that $LB \geq UB$ is a set of a solution, this set will not be investigated further (elimination of uninteresting subsets). The ideal value of the upper boundary for P_0, like an optimal approximation, must be found heuristically. As a first approximation, Soo Hoo's budget limit of 30% can be used. During the process, the UB corresponding to the current solution of P_0 is computed and compared with the current best solution. If $LB_i < UB$ and if the optimal solution is $P_1^{'}$ and is valid for P_i or P_0, then a new best solution has been found for P_0 and we replace $UB := LB_i$.

Finally, an example of a Horowitz-Sahni algorithm is shown in Figure 13. This algorithm has been used with the Fortran program from the book [36]; an example is calculated with the following data. The simple example is given by solving the Horowitz-Sahni algorithm for a given set of policies (P_{ol}) which is a special Indicator on the first level ($I_{\lambda_1} = 7$), a simple given set of $n = 7$ controls (x_j, $j = 1, ...7$), a current solution $\hat{z}$, and a current best solution z. For a given set of policies, we can elaborate on

p= $\{70, 20, 39, 37, 7, 5, 10\}$ which are useful for more than one BCP. The scale is 1,...,100 units. To face each BCP/DRP and control (consult Eq. 28) for a given set of cost of controls, also in a scale of 1,...,100 units, we use
w= $\{31, 10, 20, 19, 4, 3, 6\}$.
c= $\{50\}$ is the size of the capacity of knapsack we use.

We present the results in Figure 13. In this example, u is an upper bound and $\hat{x}_j$ is a current solution. The best solution so far is x_j.

Finally we can draw a short result from this trade-off analysis. If a company is in the range of the strategic balance between the effectiveness and the efficiency of its BCMS according to BS25999 and, if the company needs to have further improvement to reduce turnover to as little as possible, then a combinatorial optimization is very useful. Such an optimization should balance the benefit of a policy in terms of risk, which is considered for each control, and the cost of each control in terms of avoiding, mitigating, or transferring the risk to a determined limit of investment.

VII. Conclusion and future work

The empirical studies by Knight and Pretty [14] suggest that the quality of a BCMS, as well as the related BCP and DRP, should be looked at more intensely: the existence of a BCMS in accordance with BS 25999 does not necessarily say

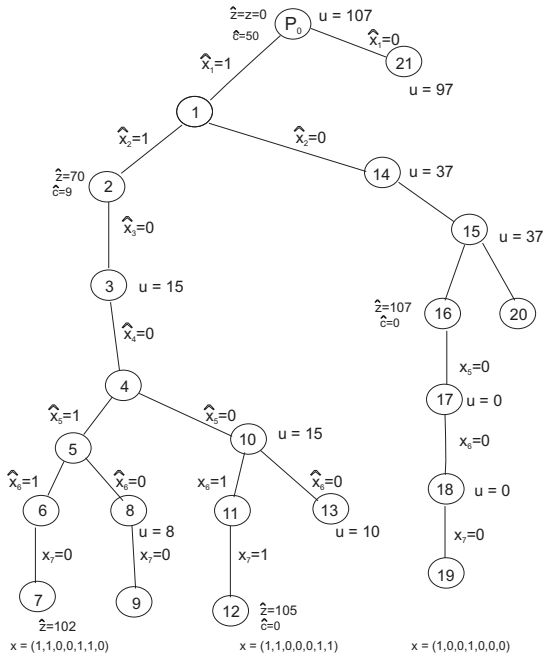


Fig. 13: Example with Horowitz-Sahni Algorithm

anything about survival probability in the event of a disaster. Survival depends on the implementation of the BCMS, and here the BCP and DRP are interpreted as reactive controls of great importance to survival in the event of a disaster.

In this paper the importance of the output and efficiency of a BCP and DRP have been demonstrated using indicators. Furthermore, it has been shown that by using two indicators, the effectiveness and economic efficiency of a BCMS can be measured. These two indicators represent key performance indicators for a company. If there are a number of measurements for effectiveness, a forecast can be made based on a random variable in terms of survival probability, but this can be done only if there is sufficient experience in applying the BCP and DRP. In addition, these key performance indicators can be used by a company to document its performance.

However, this method of using indicators evaluates the processes behind the BCP and DRP only approximately. The disadvantage of the method is that there must be sufficient experience in using the BCP and DRP; therefore, a company is not well prepared for catastrophes that are unknown. Combinations of or additions to the BCP and DRP based on similar catastrophic scenarios are only possible if the processes behind the BCP and DRP are exercised in advance using relevant types of simulations.

Unfortunately, there are still no appropriate methods to pursue these ideas. Currently, processes are typically associated with the layout of an event-driven Process Chain (ePC), which is merely a snapshot of processes, not a simulation in the sense of running a complete process. These considerations may suggest approaches for further investigation.

References

[1] W. Boehmer, "Survivability and Business Continuity Management System According to BS 25999," *Proceedings of the Emerging Security Information, Systems and Technologie, 2009. SECUWARE '09, Third International Conference on, IEEE Computer Society*, pp. 142–147, June, 18-23 2009.

[2] BS25999-1, "Business Continuity Management System – Part 1: Code of practice, BSI (UK)." ISBN 0580496015, 11 2006.

[3] BS25999-2, "Business Continity Management System – Part 2: Specification, BSI (UK)." ISBN 9780580599132, 11 2007.

[4] IBM, "Panic slowley. integrated disaster response and bulit-in business continuity," ibm.com/itsolutions/uk/governance/businesscontinuity, 2006.

[5] SC27, "ISO/IEC 27001:2005, Information technology - Security techniques - information security management systems - Requirements." Beuth-Verlag, Berlin, 10 2005.

[6] ITGI, "Cobit, control objective in information and related technology, 4th. ed.." IT Governance Institute, ISBN 1-933284-37-4, 2006.

[7] C. Brandt, T. Engel, W. Boehmer, and C. Roeltgen, "Diskussionsvorschlag einer Lösungsskizze zur Behandlung von operationellen IT-Sicherheitsrisken nach Basel II auf der Grundlage von Anforderungen der Credit Suisse," in *Multikonferenz Wirtschaftsinformatik, München, MKWI2008*, 2008.

[8] W. Boehmer, C. Brandt, and J. F. Groote, "Evaluation of a business continuity plan using process algebra and modal logic," in *2009 IEEE Toronto International Conference – Science and Technology for Humanity TIC-STH 2009 - SIASP 2*, pp. pp. 147–152, Ryerson University, 245 Church Street, Toronto, Ontario, Canada, 2009.

[9] W. Boehmer, C. Brandt, and J. Groote, "Evaluation of a business continutiy plan using process algebra and modal logic," *Computer Science Report CSR-09-12, Eindhoven University of Technology,*, 2009.

[10] M. Alemanni, G. Alessia, S. Tornincasa, and E. Vezzetti, "Key performance indicators for PLM benefits evaluation: The Alcatel Alenia Space case study," *Comput. Ind.*, vol. 59, no. 8, pp. 833–841, 2008.

[11] R. R. Rodriguez, J. J. A. Saiza, and A. O. Basa, "Quantitative relationships between key performance indicators for supporting decision-making processes," *Computer in Industry*, 2008.

[12] L. Tsinas, B. Trösken, and S. Sowa, "KPI-Framework für Informationssicherheit," 2009.

[13] W. Boehmer, "Appraisal of the Effectiveness and Efficiency of an Information Security Management System Based on ISO 27001," *Emerging Security Information, Systems, and Technologies, The International Conference on (SECUWARE 2008), IEEE Computer Society*, vol. 0, pp. 224–231, 2008.

[14] K. R. and P. D., "The impact of catastrophes on shareholder value," the oxford executive research briefings, Templeton College, University of Oxford, Oxford, England, 1996.

[15] M. Nemzow, "Business continuity planning," *Int. J. Netw. Manag.*, vol. 7, no. 3, pp. 127–136, 1997.

[16] G. Quirchmayr, "Survivability and business continuity management," in *ACSW Frontiers '04*, (Darlinghurst, Australia), pp. 3–6, Australian Computer Society, Inc., 2004.

[17] B. J. L. Landry and M. S. Koger, "Dispelling 10 common disaster recovery myths: Lessons learned from Hurricane Katrina and other disasters," *J. Educ. Resour. Comput.*, vol. 6, no. 4, p. 6, 2006.

[18] K. Saleem, S. Luis, Y. Deng, S.-C. Chen, V. Hristidis, and T. Li, "Towards a business continuity information network for rapid disaster recovery," in *dg.o '08: Proceedings of the 2008 international conference on Digital government research*, pp. 107–116, Digital Government Society of North America, 2008.

[19] I. Shklovski, L. Palen, and J. Sutton, "Finding community through information and communication technology in disaster response," in *CSCW '08: Proceedings of the ACM 2008 conference on Computer supported cooperative work*, (New York, NY, USA), pp. 127–136, ACM, 2008.

[20] S. Tjoa, S. Jakoubi, and G. Quirchmayr, "Enhancing business impact analysis and risk assessment applying a risk-aware business process modeling and simulation methodology," in *ARES '08,*, (Washington, DC, USA), pp. 179–186, IEEE Computer Society, 2008.

[21] J. Lunze, *Ereignisdiskrete Systeme; Modellierung und Analyse dynamischer Systeme mit Automaten, Markovketten und Petrinetzen*. ISBN 3-486-58071-X, Oldenbourg Verlag, 1. auflage ed., 2006.

[22] A. Calder, "PDCA Cycle & Documentation Pyramid." IT Governance: a Manager's Guide to Data Security and ISO27001/27002, ISMS Toolkit, 2007.

[23] J. Eloff and M. Eloff, "Information Security Architecture," *Computer Fraud & Security*, vol. 2005, no. 11, pp. 10–16, 2005.

[24] D. Larochelle and N. Rosasco, "Towards a Model of the Costs of Security," *Technical Report CS-2003-13, University of Virginia, Dept. of Computer Science.*, 06 2003.

[25] T. Tsiakis and G. Stephanides, "The economic approach of information security," *Computers & Security*, vol. 24, pp. 105 –108, March 2005.

[26] L. A. Gordon and M. P. Loeb, "The economics of information security investment," *ACM Trans. Inf. Syst. Secur.*, vol. 5, no. 4, pp. 438–457, 2002.

[27] W. Sonnenreich, J. Albanese, and B. Stout, "Return On Security Investment (ROSI) - A Practical Quantitative Model," *Journal of Research and Practice in Information Technology*, vol. 38, no. 1, pp. p. 45–56, 2008.

[28] *Return on security investment – proving its worth it*, vol. 2005, 2005.

[29] H. Cavusoglu, B. Mishra, and S. Raghunathan, "A model for evaluating IT security investments," *Commun. ACM*, vol. 47, no. 7, pp. 87–92, 2004.

[30] J. S. David, D. Schuff, and R. S. Louis, "Managing your total IT cost of ownership," *Commun. ACM*, vol. 45, no. 1, pp. 101–106, 2002.

[31] T. Bass and R. Robichaux, "Defense-in-depth revisited: Qualitative risk analysis methodology for complex networkcentric operations," *IEEE MILCOM*, vol. 2001, pp. 28–31, 2001.

[32] B. Blakley, E. McDermott, and D. Geer, "Information security is information risk management," in *NSPW '01: Proceedings of the 2001 workshop on New security paradigms*, (New York, NY, USA), pp. 97–104, ACM, 2001.

[33] L. Heinrich and F. Lehner, *Informationmanagement, Planung, Überwachung und Steuerung der Informationsinfrastruktur*. ISBN-13:9783486577723, Oldenbourg Verlag, 8. auflage, seite 84, ff ed., München, 2005.

[34] W. Boehmer, "Cost-benefit trade-off analysis of an ISMS based on ISO 27001," *ARES Conference, The International Dependability Conference, IEEE Computer Society*, pp. 392 –399, March, 16th. – 19th. 2009.

[35] K. S. Hoo, *How Much is Enough? A Risk Management Approach to Computer Security;*. PhD thesis, Stanford University, CRISP, 2000.

[36] S. Martello and P. Toth, *Knapsack Problems, Algorithms and Computer Implementations*. ISBN 0471924201, John Wiley and Sons Ltd., 1990.

Formalization of Security Properties: Enforcement for MAC Operating Systems and Verification of Dynamic MAC Policies

Jérémy Briffaut, Jean-François Lalande, Christian Toinard
Laboratoire d'Informatique Fondamentale d'Orléans
ENSI de Bourges – Université d'Orléans
88 bd Lahitolle, 18020 Bourges cedex, France
{jeremy.briffaut,jean-francois.lalande,christian.toinard}@ensi-bourges.fr

***Abstract*—Enforcement of security properties by Operating Systems is an open problem. To the best of our knowledge, the solution presented in this paper[1] is the first one that enables a wide range of integrity and confidentiality properties to be enforced. A unified formalization is proposed for the major properties of the literature and new ones are defined using a Security Property Language. Complex and precise security properties can be defined easily using the SPL language but the system includes 13 predefined protection templates. Enforcement of the requested properties is supported through a compiler that computes all the illegal activities associated with an existing MAC policy. Thus, we provide a SPLinux kernel that enforces all the requested Security Properties. When the MAC policies are dynamic, it becomes difficult to compute all the possible illegal activities. Our paper proposes a solution to that problem. A meta-policy includes evolution constraints for the MAC policies. A verification tool computes the requested security properties that are defined through the SPL language. That tool provides the illegal activities for all the possible MAC dynamic policies. Thus, the security administrator can verify the MAC policies and can optimize the requested security properties. It helps him to evaluate the memory and processor load associated with the enforcement of the request security properties. Efficiency is presented for protecting high-interaction honeypots.**

***Index Terms*—Security Properties, Protection, Mandatory Access Control, Verification**

I. INTRODUCTION

Security of the Operating Systems usually relies on Discretionary Access Control (DAC): access permissions are set by users on files they own, namely at the user's discretion. This access control model has proven to be fragile [2].

That is why access decisions have been moved from user control to a Mandatory Access Control (MAC) associated with a protection policy that describes how subjects and objects are allowed to interact. For example, the NSA's Security-Enhanced Linux (SELinux) has been developed as part of the Linux kernel. It implements a strong and highly configurable MAC. Nevertheless these MAC systems can only guarantee simple security properties (as a process cannot access a file). Complex security properties cannot be ensured by this approach. For example, existing MAC systems do not control information flows involving multiple processes and resources.

The literature shows there is no approach to easing the formalization of the required security properties. The authors, to date, consider only specific properties but do not provide a generic method to formalize a large range of security properties related to integrity or confidentiality. Enforcement is also limited to a subset of the required properties. To the best of our knowledge, our proposal is the first one that 1) enables a large set of security properties to be defined and 2) provides a generic method for enforcing all the properties that are supported by our Security Property Language.

13 security templates are presented in this paper that formalize some well know security properties of the literature and propose new ones. We consider that these properties are the most important ones and some of them have been tested on a honeypot. But, new ones can be defined very easily, either by the definition of a new property or by the composition of the proposed template. Moreover, the 13 proposed templates are very flexible. A security administrator can easily reuse them to express multiple concrete security properties. He can define a concrete property with specific values for the arguments of the template.

An extended Linux kernel, called SPLinux, is available to control all the concrete security properties that are requested by the security administrator. This is the first solution to enforce such a large and precise set of security properties. A security administrator uses our SPL compiler to compute two kinds of inputs: 1) the MAC policy available at the target Operating System and 2) the requested concrete properties defined using the SPL language. The compiler produces all the illegal activities that are allowed by the target MAC policy. Then, the Linux kernel uses a userland application that controls all the illegal activities. Thus, a system call fails according to 1) the DAC policy, 2) the MAC policy and 3) the required concrete properties. So, enforcement of the requested security properties is proposed for a target MAC system e.g. SELinux.

When MAC policies are dynamic, the MAC policy states can become infinite. In order to be able to control those states, evolution constraints are required to reduce the possible

[1]This paper is an extended work of the paper presented at SECURWARE 2009 [1].

states. A meta-policy approach has been proposed in previous works [3]. However, the security administrator needs a tool to help him decide if the meta-policy is safe and if the related MAC policy states can be computed efficiently by the SPL compiler. Our verification tool takes two kinds of inputs: the meta-policy that controls the MAC policies and the required concrete security properties. Our verification system provides at least one decision showing an illegal activity. It can also provide, for a given MAC meta-policy, all the possible illegal activities associated with the enforcement of each property. Thus, the security administrator can decide either to 1) remove a security property that cannot be violated or 2) add a better security property that reduces the number of illegal activities or 3) modify the meta-policy in order to minimize the overhead.

A real application is proposed for protecting high interaction honeypots. Our high interaction honeypot is a highly distributed system. A single meta-policy protects our distributed honeypot against corruption of the various nodes. Despite more than two years of experimentation and numerous intrusions, the target Operating System has never been compromised. This large scale experiment have been precisely described in [4]. Even if this experiment is not a formal proof that the meta-policy protects the honeypot against all the possible attacks, we think that a system that has never been compromised while hosting malicious users suggests the efficiency of the proposed protection mechanism.

II. Related Work

Under Linux, there are at least four security models available to ensure a Mandatory Access Control policy: SELinux, GRSecurity, SMACK and RSBAC. But none of these solutions can ensure a large set of security properties. At best they can ensure one or two limited properties, such as the Bell and LaPadula confidentiality property or the Biba integrity property. Under the BSD family, solutions such as Trusted BSD (available within the following Operating Systems: FreeBSD, OpenBSD, MacOSX, NetBSD) provide more or less the same kind of Mandatory Access Control as SELinux. But, there again they fail to ensure the great majority of requested security properties.

The major limitation is associated with a transitive closure of allowed system calls that enables a security property to be violated. All the existing approaches fail to manage the causal relationships between the system calls correctly, thus authorizing illegal activities.

Several works address how security properties can be enforced within an Operating System. [5] considers only protection policies and enforcement mechanisms, such as those in MAC systems. Thus, the author does not deal with information flows and more generally does not deal with confidentiality or integrity but rather safety. Solutions such as [6], [7] consider noninterference between privileged and unprivileged entities, which is only one specific property. However, they cannot formalize classes of security properties, such as the prevention of flows between privileged entities.

Many solutions deal with the detection of violations of security properties. Solutions such as [8] detect violation of both confidentiality and integrity between privileged and unprivileged entities. Again, those solutions consider only noninterference and cannot enforce other classes of security properties.

[9], [10] detects the flows violating a DAC policy. This is a limited property. Moreover, its practical usage is very limited for DAC systems since a DAC policy is not safe and the related detections include numerous false positives and false negatives.

[11] counts more than 150 publications related to information-flow security. The majority deal with noninterference. The other part aims at enforcing information-flow policies using program analysis techniques. Information flow analysis in a program language is not suited to enforcing protection between several processes that are using the services of an Operating System.

Finally, many works address the conformance of policies. For example, [12] verifies that XACML access control policies really match the Bell and LaPadula, Biba or Chinese Wall models. But, those solutions cannot manage a large range of security properties and do not deal with dynamic policies. [13] considers the verification of SELinux policies. However, it does not provide any administration language to ease the formalization of the required security properties, nor does it manage dynamic MAC policies.

The HiStar Operating System [14] associates each objet or subject with an information flow level. Four different information levels are proposed. For example, a subject with level 1, cannot read an object with level 3. In practice, those levels are very close to capabilities. In HiStar a capacity is a collection of read and write levels that must be consistent with the requested object level. The problem of HiStar is that it is very close to the Bell and Lapadula model and it suffers from the same limitations.

The Flume Operating System [15] proposes a reduced form of MAC policy. They consider two kinds of processes, unconfined and confined. A confined process cannot access the file systems through dedicated system calls. The control of unconfined processes requires a dedicated MAC policy between security contexts. In contrast with SELinux, it is not a fine grain MAC system. Flume is in essence very close to HiStar since it uses capabilities as security contexts. However, Flume does not control information flows.

Asbestos [16] reuses the idea behind HiStar by considering four different levels of information. The protection rules can only express pairwise relationship patterns. Again, information flows involving multiple interactions and processes cannot be controlled easily.

Works related to the enforcement of dynamic policies, such as [17], [18], generally consider how to detect simple conflicts within dynamic policies. For example, they detect if it is safe to remove or add a role or a context, otherwise the considered access control could become invalid, conflicting or unsupported. So, they address conflicting rules but do not

enforce a large set of security properties.

To the best of our knowledge, even recent works such as [16], [15], [14], [19], [20], [21] do not provide any administration language that enables a large range of security properties to be easily formalised. All the Operating Systems described in the literature fail to guarantee the requested security properties. The security of Operating Systems thus remains an open problem. The literature does not address the enforcement of precise and flexible security properties related to integrity and confidentiality. Finally, work on dynamic policies mainly studies how to deal with conflicting protection rules. However, available tools are not able to compute the illegal activities for dynamic MAC policies in order to enforce a large set of security properties. Finally, there has been, as yet, no real experiment. Our paper addresses all the above points.

III. Abstract Security Language

In order to formalize security properties associated with the activities of the Operating System, let us firstly define the model of the target MAC systems. That model fits the majority of MAC systems. Some of them do not present certain of the modelized functionalities, such as the control of process transitions. In such cases, our formalization is reduced to the functionalities provided by the target MAC system. However, our modelling is adapted for a MAC system providing a fine grain control such as SELinux. For other MAC systems, extensions can be implemented to be able to enforce all the required security properties. Details about how to extend existing Linux systems to enforce the required security properties are available in [22]. In that paper, an abstract Security Property Language is provided to model the required security properties. Hereafter, a concrete Security Property Language uses that abstract language and applications are given for the SELinux systems.

Since the causal dependencies between the system calls and the formalization of the system activities are poorly addressed in the literature, this section defines all the notions of our abstract SPL. In the next section IV, the abstract SPL language enables us to formalize 13 security templates.

A. System representation

A system is modellised by a set of security contexts performing operations on other security contexts. The security contexts identify the various entities of the system. Let us denote as SC the whole set of security contexts existing in a given system. Two sets of security contexts are considered ($SC = SCs \cup SCo$). SCo is the set of security contexts acting as an object: each $sc_o \in SCo$ characterizes a passive entity (file, socket, ...) on which system calls can be performed. SCs is the set of security contexts acting as a subject: each $sc_s \in SCs$ characterizes an active entity, i.e. processes, that can perform actions, i.e. system calls. For example, let us consider the Apache webserver reading an HTML file, the Apache process is identified as a subject ($apache\_t \in SCs$) and the file is considered as an object ($var\_www\_t \in SCo$).

Let us denote as IS (Interaction Set) the set of all the elementary operations, i.e. system calls, existing in the system (read, write, execute, ...). An interaction it represents a subject $sc_s \in SCs$ executing an operation $eo \in IS$ on a given context $sc_t \in SC$. An interaction is a 3-uple defined by:

$$it = (sc_s \in SCs, sc_t \in SC, eo \in IS)$$

noted:

$$sc_s \xrightarrow{eo} sc_t$$

When an operation is performed, there are two consequences from the security point of view. The interaction can produce:

- an information transfer noted $sc_1 > sc_2$;
- a transition $sc_s \xrightarrow{trans} sc_t$.

An *information transfer* conveys information from context sc_1 to sc_2 using a write-like operation or a read-like operation. For example, the read interaction $sc_{apache} \xrightarrow{file:read} sc_{var\_www}$ corresponds to the information transfer $sc_{var\_www} > sc_{apache}$ and the write interaction $sc_{apache} \xrightarrow{file:write} sc_{var\_www}$ corresponds to the information transfer $sc_{apache} > sc_{var\_www}$. A precise definition about read and write-like operations is given later in section III-D.

A *transition* enables a process to move from context sc_s to context sc_t. When the process is running in the context sc_t, it acquires new privileges associated with sc_t. For example, the sc_{init} process launches the Apache web server using a forked process that transits using transition $sc_{init} \xrightarrow{process:transition} sc_{apache}$.

B. Causal dependency

The state of the art related to causality shows that there is no relevant definition of causal dependency to express security properties. The difficulty lies in having a satisfactory estimator of causality between interactions. A new definition of causal dependency is given in this paper between a source and a target context. Two interactions are causally dependant if:

- those interactions share a common security context
- the first interaction occurs before the end of the second interaction
- the first interaction modifies the state of the shared security context
- the second interaction modifies the state of the final security context

The figure 1 gives an example of a causal dependency between two interactions it_1 and it_2. This example clearly shows that:

1) sc_2 is a shared context,
2) it_1 finishes before the end of it_2,
3) an information flow occurs from sc_1 to sc_2,
4) an information flow occurs from sc_2 to sc_3.

Definition 3.1: The causal dependency between two interaction it_1 and it_2, noted $it_1 \nrightarrow it_2$, is defined by

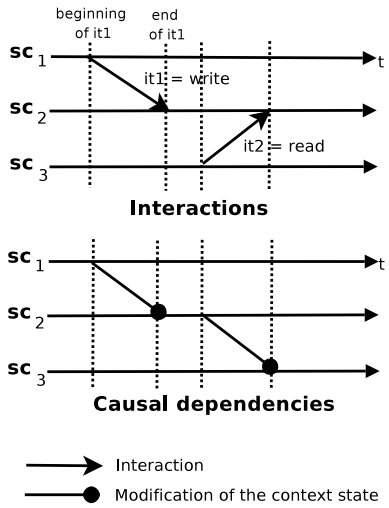


Fig. 1. Read and Write interaction / Causal dependency

$$\begin{cases} sc_2 \in it_1, \\ sc_2 \in it_2, sc_3 \in it_2, \\ date_{sc_2}(begin(it_1)) \leqslant date_{sc_2}(end(it_2)), \\ it_1 \text{ modifies the state of the shared context } sc_2, \\ it_2 \text{ modifies the state of the security context } sc_3. \end{cases}$$

That definition overestimates the probability that the information can flow from the source to the target. However, it is a satisfactory estimator which only takes into account the possible causal relationships. Our abstract SPL efficiently uses that causal dependency to make the formalization of advanced security properties easier.

C. System activities

In this section, an abstract language is proposed to describe the activities of the different processes. That language uses our definition of causal dependency to model the activities presenting indirect flows. Several operators enable those activities to be combined.

A system activity is defined by a set of interactions. We propose to distinguish four activity classes:

- The *interaction class* is the class where each activity includes a single interaction that can permit direct flows.
- The *sequence class* contains activities composed of sequences of interactions. It enables indirect flows involving several interactions to be modelled.
- The *correlation class* is the class where activities are a combination of interactions and/or sequences.
- The last class of interactions are the remaining activities which are not expressed by our language, i.e. activities that cannot be observed by the target Operating System.

1) Sequence class: A sequence of interaction is a transitive closure of causally dependent interactions. It models indirect flows.

Definition 3.2: A sequence of n interactions, noted $sc_{source} \Rightarrow sc_{target}$, from a context sc_{source} to sc_{target} is a transitive closure of $n-1$ causal dependencies

$$s.a. \begin{cases} n \geq 2, \\ sc_{source} \in it_1, \\ sc_{target} \in it_n, \\ \forall k = 1..n-1, it_k \twoheadrightarrow it_{k+1} \end{cases}$$

As performed in section III-A we distinguish two kinds of sequences, first, an *information flow* where each interaction is an information transfer, second a *transition sequence* where each interaction of the sequence is a transition. Formally, we obtain:

Definition 3.3: An information flow between sc_{source} and sc_{target}, noted $sc_{source} \gg sc_{target}$, is a sequence $sc_{source} \Rightarrow sc_{target}$

$$s.a. \begin{cases} sc_{source} \Rightarrow sc_{target} = \{it_1, \ldots, it_n\}, \\ \forall k = 1..n, it_k = sc_k > sc_{k+1} \end{cases}$$

Definition 3.4: A transition sequence between sc_{source} and sc_{target}, noted $sc_{source} \Rightarrow_{trans} sc_{target}$, is a sequence $sc_{source} \Rightarrow sc_{target}$

$$s.a. \begin{cases} sc_{source} \Rightarrow sc_{target} = \{it_1, \ldots, it_n\}, \\ \forall k = 1..n, it_k = sc_k \xrightarrow{trans} sc_{k+1} \end{cases}$$

Using the example in figure 1, the sequence $\{(sc_1, sc_2, \{file : write\}), (sc_3, sc_2, \{file : read\})\}$ is an information flow $sc_1 \gg sc_3$ composed of two transfers: $sc_1 > sc_2$ and $sc_2 > sc_3$. In the case of two transitions, we obtain the transition sequence $sc_1 \Rightarrow_{trans} sc_3$.

2) Correlation class: A correlation is a combination of several sequences and/or interactions. It represents a complex activity of the system. For example, a user can access an Apache information if that user transits to the Apache context and then reads the Apache configuration . This activity is composed of one sequence of transition $sc_{user} \Rightarrow_{trans} sc_{apache}$ and one interaction $sc_{apache\_conf\_t} > sc_{apache}$. To describe the relationships between the elements of the correlation, we define three operators $\circ$, $\wedge$ and $\vee$.

For each sequence $f = sc_1 \xrightarrow{eo_1} \ldots \xrightarrow{eo_{n-1}} sc_n$, noted $sc_1 \Rightarrow sc_n$, let us define a function F which associates the last context sc_n to the first context sc_1, that is: $F(sc_1) = sc_n$.

Definition 3.5: Let $f = sc_{f_1} \Rightarrow sc_{f_n}$ and $g = sc_{g_1} \Rightarrow sc_{g_m}$ two sequences where $sc_{f_n} = sc_{g_1}$. Let F and G be the corresponding functions.
The composition of two sequences f and g corresponds to a global sequence $sc_{f_1} \Rightarrow (sc_{f_n} = sc_{g_1}) \Rightarrow sc_{g_m}$ that respects the equation: $G \circ F(sc_{f_1}) = G[F(sc_{f_1})] = G(sc_{f_n}) = G(sc_{g_1}) = sc_{g_m}$. We note $g \circ f$ this new sequence $sc_{f_1} \Rightarrow sc_{g_m}$.

Definition 3.6: Let a and b two interactions, sequences or correlations, $(a \wedge b)$ is observed if a is observed and b is observed.

Definition 3.7: Let a and b two interactions, sequences or correlations, $(a \vee b)$ is observed if a is observed or b is observed.

$$
\begin{array}{rcl}
activity & ::= & [\ description\ "="\ ]\ correlation \\
correlation & ::= & (correlation \wedge correlation) \\
 & & |(correlation \vee correlation) \\
 & & |composition \\
composition & ::= & (composition \circ composition) \\
 & & |terminal \\
terminal & ::= & sequence|interaction \\
interaction & ::= & sc \xrightarrow{eo} sc|sc > sc|sc \xrightarrow{trans} sc \\
sequence & ::= & sc \Rightarrow sc|sc \gg sc|sc \Rightarrow_{trans} sc \\
sc & ::= & "\ security\ context\ " \\
eo & ::= & "\ elementary\ operation\ " \\
description & ::= & "\ name\ of\ activity\ "
\end{array}
$$

Fig. 2. Grammar of activities

We give two particular definitions for the correlation class using the defined operators. The first definition gives a subject access to a special privilege on a given object.

Definition 3.8: Let $seq = sc_{source} \Rightarrow sc_{inter}$ be an interaction sequence and $it = sc_{inter} \xrightarrow{eo} sc_{target}$, the composition $(it \circ seq)$ overestimates the possibility of the privilege access eo on sc_{target} by sc_{source}. It is noted: $sc_{source} \Rightarrow_{eo} sc_{target}$.

The second definition represents a subject access to the information of a given object:

Definition 3.9: Let $seq = sc_{acces} \Rightarrow sc_{inter}$ be a sequence of interactions and $flow = sc_{info} \gg sc_{inter}$ an information flow, the correlation $(seq \wedge flow)$ overestimates the possibility of accessing the information contained in sc_{info} by sc_{acces}. It is noted $sc_{acces} \Rrightarrow sc_{info}$.

These two correlations overestimate the possibility of accessing a privilege or an information. We call these two cases an *indirect access* to privilege/information. We define two subcases that improve the estimation: if the sequence of interaction $seq = sc_{acces} \Rightarrow sc_{inter}$ is a sequence of transitions $seq = sc_{acces} \Rrightarrow_{trans} sc_{inter}$ this means that the process has a direct access to the privilege/information. We call this case a *direct access* to privilege/information.

These operators are used to define the grammar of the language that describes all the possible activities of the three classes (interactions, sequences, correlation), presented in figure III-C2. With this grammar we can express complex activities, as shown in the following example: the activity $((it \circ seq_{trans_1}) \wedge (seq_{trans_2} \wedge (flow \vee seq_{int}))$ begins with a transition sequence seq_{trans_1} that permits the interaction it; the activity is also composed of a sequence of transitions seq_{trans_2} and a sequence of information transfer $flow$ or a sequence of interactions seq_{int}.

3) Activities overview: Table I gives an overview of all possible activities modelled in this section. These activities are separated into three classes: interactions, sequences, and correlations. The top part of the Table describes these three classes without any assumption about the nature of each operation. In the bottom part, the activities that characterize information flow, transition and privilege access are summed-up with their associated notations.

D. Complementary definitions

Additional functions are needed to model the security properties that will be presented in section IV. These functions mainly characterise the operations of IS.

Definition 3.10: $\texttt{is\_write\_like}:IS \rightarrow \{true, false\}$ is the function that says if an operation modifies an object and can be assimilated to the write operation.

Definition 3.11: $\texttt{is\_read\_like}:IS \rightarrow \{true, false\}$ is the function that says if an operation gets information from an object and can be assimilated to the read operation.

Definition 3.12: $\texttt{is\_execute\_like}:IS \rightarrow \{true, false\}$ is the function that says if an operation can execute an object.

Definition 3.13: $\texttt{is\_add\_like}:IS \rightarrow \{true, false\}$ is the function that says if an operation can add information to an object without being able to read the previous written information in this object.

IV. Security properties

This section specifies the proposed template of security properties using the activity language defined in the previous section III. These 13 templates include the most known security properties of the literature but include also new security properties in order to show how powerful our language is. This set of templates is not limited. Other templates can be easily defined using the concrete SPL that implements the abstract SPL.

The 13 proposed properties are the ones that we setup on our honeypot to experiment them [4]. The obtained results are briefly described in section VIII. The classical properties of the literature will not be new for the reader: integrity, confidentiality and the well known variants like the Biba integrity or the Bell and LaPadula confidentiality property are examples that enters the proposed model. It shows how the model is able to clearly describe these properties. At the end of the section, the Table II gives an overview of the properties with a name and refers to their formal definition. This is crucial to have such a formal definition of the properties and to be able to show, in the section V how the SPL language will implement these properties.

A. Integrity

This section proposes four types of integrity security properties from the literature:

- Object integrity, as defined in [23]
- Biba Integrity, which implements the Biba model [24]
- Integrity of subjects, which implements noninterference between processes [8]
- Domain Integrity, which implements a chroot

TABLE I
ACTIVITIES OVERVIEW

	Interaction	Sequence	Correlation
Neutral	Interaction $sc_1 \xrightarrow{eo} sc_2$	Interactions sequence $sc_{source} \Rightarrow sc_{target}$	Access to information $sc_{source} \Rrightarrow sc_{target}$
			Privilege access $sc_{source} \Rightarrow_{eo} sc_{target}$
Characterisation	Information transfer $sc_1 > sc_2$	Information flow $sc_{source} \gg sc_{target}$	Acces to information by transition $sc_{source} \Rrightarrow_{trans} sc_{target}$
	Transition $sc_1 \xrightarrow{trans} sc_2$	Sequence of transitions $sc_{source} \Rightarrow_{trans} sc_{target}$	
			Privilege access by transition $sc_{source} \Rightarrow_{trans\_eo} sc_{target}$

1) Object integrity: The integrity of a system object can be altered when a write operation is performed on this object. This object integrity property ensures that no process of the system is able to modify a resource, either directly or by a sequence of interactions. The direct protection of an object is easily obtained with a MAC mechanism. Nevertheless, better integrity protection is required against sequences of operations. For example, if a user uses an exploit on Apache that leads Apache to modify a file of /var/www, this is a sequence of operations that violates the integrity of /var/www.

Property 4.1: The integrity of an object $sco_1 \in SC_O$, noted $P_{4.1}(scs_1, sco_1)$, is respected by a subject $scs_1 \in SC_S$ if:

$$\forall eo \in IS \text{ s.a. } scs_1 \xrightarrow{eo} sco_1, \neg is\_write\_like(eo)$$

$$\wedge$$

$$\forall eo \in IS \text{ s.a. } scs_1 \Rightarrow_{eo} sco_1, \neg is\_write\_like(eo)$$

Property 4.2: The integrity of a set of objects $SC_{O1} \subset SC_O$, noted $P_{4.2}(SC_{S1}, SC_{O1})$, is respected by a set of subjects $SC_{S1} \subset SC_S$ if:

$$\forall sco_1 \in SC_{O1},\ \forall scs_1 \in SC_{S1},\ P_{4.1}(scs_1, sco_1)$$

2) Biba Integrity: The BIBA model [24] defines three rules to guarantee the integrity of the system:

1) To allow scs_1 to have read access to an object sco_1, its integrity level must be less than or equal to the object integrity level;
2) To allow scs_1 to have write access to an object sco_1, its integrity level must be greater than or equal to the object integrity level;
3) To allow scs_1 to invoke a subject scs_2, its integrity level must be greater than or equal to the invoked subject integrity level;

A level of integrity is needed for each entity of the system: let $I : SC \rightarrow \mathbb{N}$ be a function that gives the level of integrity of a security context.

Property 4.3: The Biba integrity of an object $sco_1 \in SC_O$, noted $P_{4.3}(scs_1, sco_1)$, is respected by a subject $scs_1 \in SC_S$ if:

$$\forall eo \in IS \text{ s.a.} \begin{cases} scs_1 \xrightarrow{eo} sco_1 \\ is\_read\_like(eo) \end{cases}, I(scs_1) \leqslant I(sco_1)$$

$$\forall eo \in IS \text{ s.a.} \begin{cases} scs_1 \xrightarrow{eo} sco_1 \\ is\_write\_like(eo) \end{cases}, I(scs_1) \geqslant I(sco_1)$$

$$\forall eo \in IS \text{ s.a.} \begin{cases} scs_1 \xrightarrow{eo} sco_1 \\ is\_execute\_like(eo) \end{cases}, I(scs_1) \geqslant I(sco_1)$$

Property 4.4: For a system, composed by a set of contexts SC, the integrity of this system, noted $P_{4.4}(SC)$, is respected if:

$$\forall scs, sco \in SC_S \times SC_O, P_{4.3}(scs, sco)$$

3) Integrity of Subjects: The integrity of subjects corresponds to the non-interference property [25]. The non-interference property uses the notation of commutativity. Two interactions $it_1 = (scs_1, sco_1, eo_1)$ and $it_2 = (scs_2, sco_1, eo_2)$ are commutative if the operation eo_1 does not change what the context scs_2 sees of sco_1. For example, the operations $eo_1 = \{file : write\}$ and $eo_2 = \{file : read\}$ are not commutative, and the operations $eo_1 = \{file : read\}$ and $eo_2 = \{file : read\}$ are commutative. Let $commute : IS \times IS \rightarrow \{true, false\}$ the function that returns the commutativity of two operations using the table defined in [26].

Property 4.5: The integrity of a subject $scs_1 \in SC_S$, noted $P_{4.5}(scs_1, scs_2)$, is respected by a subject $scs_2 \in SC_S$ if:

$$\forall sco_1 \in SC_O,\ \forall eo_1, eo_2 \in IS \text{ s.a.} \begin{cases} scs_1 \xrightarrow{eo_1} sco_1 \\ scs_2 \xrightarrow{eo_2} sco_1 \end{cases}, \quad commute(eo_1, eo_2)$$

$$\forall sco_1 \in SC_O,\ \forall eo_1, eo_2 \in IS \text{ s.a.} \begin{cases} scs_1 \Rightarrow_{eo_1} sco_1 \\ scs_2 \Rightarrow_{eo_2} sco_1 \end{cases}, \quad commute(eo_1, eo_2)$$

Property 4.6: For a system, composed by a set of subject contexts SC_S, the integrity of these subjects, noted $P_{4.6}(SC_S)$, are guaranteed if:

$$\forall scs_1, scs_2 \in SC_S, P_{4.5}(scs_1, scs_2)$$

4) Domain Integrity: The property of domain integrity is linked to the notion of *chroot* available in operating systems or software. A set of contexts in a *chroot* are isolated from other outside contexts. Any interaction with a context outside the chroot (the domain) is a violation of the domain property.

Property 4.7: Let $chroot_{in} \subset SC$ be a set of contexts representing a domain.
The integrity of the domain $chroot_{in}$, noted $P_{4.7}(chroot_{in})$, is respected if:

$$\begin{array}{l} \forall eo \in IS \\ \forall sc_1, sc_2 \in SC \end{array} \text{, s.a. } sc_1 \xrightarrow{eo} sc_2,$$

$$\begin{cases} sc_1 \in chroot_{in} \Longrightarrow sc_2 \in chroot_{in} \\ \vee \\ sc_2 \in chroot_{in} \Longrightarrow sc_1 \in chroot_{in} \end{cases}$$

B. Confidentiality

The confidentiality property prevents unwanted accesses. Thus, information flow cannot be operated in order to steal information from an object. Four confidentiality properties are declined in this section:

- Confidentiality of system contexts
- Bell & LaPadula confidentiality
- Bell & LaPadula restrictive confidentiality
- Data access consistency

1) Confidentiality of system contexts: This property guarantees that a context sc_2 is confidential for the context sc_1 if no information flow is possible from sc_1 to sc_2.

Property 4.8: The confidentiality of $sc_2 \in SC$ vs $sc_1 \in SC$, noted $P_{4.8}(sc_1, sc_2)$, is guaranteed if:

$$\begin{cases} \nexists (sc_2 > sc_1) \\ \nexists (sc_2 \gg sc_1) \\ \nexists (sc_1 \Rrightarrow sc_2) \end{cases}$$

Property 4.9: For a system, composed of a set of contexts SC, the confidentiality of a subset of contexts $SC_1 \subset SC$ vs a subset $SC_2 \subset SC$, noted $P_{4.9}(SC_1, SC_2)$, is guaranteed if:

$$\forall sc_1, sc_2 \in SC_1 \times SC_2,\ P_{4.8}(sc_1, sc_2)$$

2) Bell & LaPadula confidentiality: The Bell & LaPadula model uses a level of classification for the subjects and a sensitivity level for objects [27]. Two rules have to be respected in order to guarantee the Bell & LaPadula confidentiality property:

1) if a subject context performs a read operation on an object, its level of classification has to be greater than the sensitivity level of the object;
2) if an information flows from the object o_2 to o_1, the sensitivity level of o_2 must be greater than that of o_1.

Let $class : SC \rightarrow \mathbb{N}$ the function that associates a subject with its classification level and for an object its sensitivity level.

Property 4.10: The Bell & LaPadula confidentiality property of an object $sco_1 \in SC_O$, noted $P_{4.10}(sco_1, scs)$, is respected by a subject $scs \in SC_S$ if:

$$\exists scs > sco_1,\ class(scs) \geqslant class(sco_1)$$

$$\forall sco_2 \in SC_O \text{ s.a. } sco_2 \gg sco_1,\ class(sco_1) \geqslant class(sco_2)$$

Property 4.11: For a system, composed of a set of contexts SC, the Bell & LaPadula confidentiality of the system, noted $P_{4.11}(SC)$, is guaranteed if:

$$\forall sco, scs \in SC_O \times SC_S,\ P_{4.10}(sco, scs)$$

3) Bell & LaPadula restrictive confidentiality: In contrast with the previous property, the Bell & LaPadula restrictive confidentiality provides a finer grain protection. Three rules are needed to express this property:

1) if a subject context reads an object, its level of classification has to be greater than the sensitivity level of the object;
2) if a subject context adds information to an object (for example, appending text with no possible read and modification of the existing data on a text file), its level of classification has to be less than the sensitivity level of the object;
3) if a subject context modifies an object (for example, read and write classification on a text file), its level of classification has to be equal to the sensitivity level of the object;

Property 4.12: The Bell & LaPadula restrictive confidentiality property of an object $sco \in SC_O$, noted $P_{4.12}(sco, scs)$, is respected by a subject $scs \in SC_S$ if:

$$\exists scs > sco_1,\ class(scs) \geqslant class(sco)$$

$$\forall eo \in IS \text{ s.a. } \begin{cases} scs \xrightarrow{eo} sco \\ is\_add\_like(eo) \end{cases}, class(scs) \leqslant class(sco)$$

$$\exists sco_1 > scs,\ class(scs) = class(sco)$$

Property 4.13: For a system, composed of a set of contexts SC, the Bell & LaPadula restrictive confidentiality of the system, noted $P_{4.13}(SC)$, is guaranteed if:

$$\forall sco, scs \in SC_O \times SC_S,\ P_{4.12}(sco, scs)$$

4) Data access consistency: This property aims to guarantee that the access to the data is consistent between direct access (one interaction) and a sequence of interactions that leads to data access. In contrast with data access consistency for DAC systems [10], our property provides a better approach since a MAC policy can be safe. In case of MAC systems, if a context cannot read a file directly, there is no reason for it to be able to read the information using a transition to another context.

Property 4.14: The data access consistency property of two contexts $sc_1, sc_2 \in SC^2$, noted $P_{4.14}(sc_1, sc_2)$, is respected if:

$$sc_1 \Rrightarrow sc_2 \text{ implies } sc_2 > sc_1$$

Property 4.15: For a system, composed of a set of contexts SC, the data access consistency property, noted $P_{4.15}(SC)$, is guaranteed if:

$$\forall sc_1, sc_2 \in SC, P_{4.14}(sc_1, sc_2)$$

C. Privilege abuse

This class of security property is designed to prevent a malicious use of the privileges available on a system. We distinguish five possible types of privilege abuse:

- The separation of duties
- No context transition
- Trusted path execution
- Policy abuse
- Meta-policy abuse

1) The Separation of Duties: The property of the separation of duties is defined in [28] by the idea that an object created by one person cannot be executed by that same person. A more general definition is given in [29]: several operations performed on a same object must be done so by different users. This property must take into account that:

- a subject can perform operations directly on the subject using an interaction: in this case we call the property the separation of direct duties;
- a subject can use a sequence of operations to perform the operation on the object: in this case we call the property the separation of extended duties.

Let us give a more general definition of the separation of duties associated with two sets of operations: OP_1 and OP_2. Then, this general definition is illustrated with the definition given in [28] that concerns the special write operation.

Property 4.16: Let $OP_1, OP_2 \subset IS^2$ two sets of elementary operations. A system composed of contexts $SC = SC_S \cup SC_O$ respects the separation of duties of OP_1 and OP_2 if:

$$\left.\begin{array}{l}\forall sc_s \in SC_S \\ \forall sc_o \in SC_O \\ \forall eo_1, eo_2 \in IS\end{array}\right. \text{ s.a. } \left\{\begin{array}{l} it_1 = sc_s \xrightarrow{eo_1} sc_o \\ it_2 = sc_s \xrightarrow{eo_2} sc_o \\ eo_1 \in OP_1 \\ (it_2 \circ it_1)\end{array}\right. , eo_2 \notin OP_2$$
$$\wedge$$
$$\left\{\begin{array}{l} it_3 = sc_s \Rightarrow_{eo_1} sc_o \\ it_4 = sc_s \Rightarrow_{eo_2} sc_o \\ eo_1 \in OP_1 \\ (it_4 \circ it_3)\end{array}\right. , eo_2 \notin OP_2$$

The definition of [28], called the separation of execute/write duties property can be implemented with $OP_2 = \{execute\}$ and $OP_1 = \{write\}$:

Property 4.17: The separation of execute/write duties property on an object $sco \in SC_O$ for a subject $scs \in SC_S$, noted $P_{4.17}(scs, sco)$, is respected if:

$$\forall eo_1, eo_2 \in IS \text{ s.a. } \left\{\begin{array}{l} it_1 = scs \xrightarrow{eo_1} sco \\ it_2 = scs \xrightarrow{eo_2} sco \\ is\_write\_like(eo_1) \\ (it_2 \circ it_1)\end{array}\right. ,$$
$$\neg is\_execute\_like(eo_2)$$

Property 4.18: The separation of execute/write duties property for a subject $scs \in SC_S$ and all possible related objects of the system, noted $P_{4.18}(scs)$, is respected if:

$$\forall sc_o \in SCO,\ P_{4.17}(scs, sco)$$

Property 4.19: The separation of extended execute/write duties property on an object $sco \in SC_O$ for a subject $scs \in SC_S$, noted $P_{4.19}(scs, sco)$, is respected if:

$$\forall eo_1, eo_2 \in IS \text{ s.a. } \left\{\begin{array}{l} it_1 = scs \Rightarrow_{eo_1} sco \\ it_2 = scs \Rightarrow_{eo_2} sco \\ is\_write\_like(eo_1) \\ (it_2 \circ it_1)\end{array}\right. ,$$
$$\neg is\_execute\_like(eo_2)$$

Property 4.20: For a system, composed of a set of contexts SC, the separation of execute/write duties property of the system, noted $P_{4.20}(SC)$, is guaranteed if:

$$\forall scs, sco \in SC_S \times SC_O,\ P_{4.17}(scs, sco) \wedge P_{4.19}(scs, sco)$$

2) No Context transition: Let us define a new property called "No context transition" that guarantees that a process will not be able to transit to another context. This can be considered as an integrity property for the process running in this context; it can also be seen as a sort of confidentiality for this process that cannot change its context and bring information into another context.

Property 4.21: The no context transition of a subject scs_1 with scs_2, noted $P_{4.21}(scs_1, scs_2)$, is guaranteed if:

$$\left\{\begin{array}{l} \neg scs_1 \xrightarrow{trans} scs_2 \\ \neg scs_1 \Rightarrow_{trans} scs_2\end{array}\right.$$

A more general version of this property protects one security context that cannot transit to any other context. For example, this property can be applied to the Apache web server that will not be allowed to transit to any other context in case of an attack trying to exploit a vulnerability.

Property 4.22: The no context transition of a subject scs_1, noted $P_{4.22}(scs_1)$, is guaranteed if:

$$\forall scs_2 \in SC_S \text{ s.a. } P_{4.21}(scs_1, scs_2)$$

3) Trusted path execution: This notion of trust refers to the administrators that trust the executables they have installed but does not trust the executables installed by other users. To use this property, a set of contexts noted TPE is built by the administrator to group all the software that he wants to authorize on the system. Any file that can be executed outside of this set breaks the trusted path execution property.

Property 4.23: Let $TPE \subset SC$ the set of the trusted contexts. The system guarantees the trusted path execution property, noted $P_{4.23}(TPE)$, if:

$$\forall sc_1, sc_2 \in SC, \forall eo \in IS \text{ s.a } \left\{\begin{array}{l} sc_1 \xrightarrow{eo} sc_2 \\ is\_execute\_like(eo)\end{array}\right. ,$$
$$sc_2 \in TPE$$

4) Policy abuse: A MAC policy can be deployed on a system. This policy, noted $\mathcal{POL}$, defines all the allowed interactions on the system. Any operation not included in this policy is a violation attempt.

Property 4.24: Let $\mathcal{POL} = \{it_1, \ldots, it_n\}$ be a MAC policy. A system guarantees the respect of the policy abuse property, noted $P_{4.24}(SC, \mathcal{POL})$, if:

$$\forall sc_1, sc_2 \in SC, \forall eo \in IS \text{ s.a. } it = sc_1 \xrightarrow{eo} sc_2,$$

$$it \in \mathcal{POL}$$

5) Meta-policy abuse: This property guarantees that a meta-policy, defined in section VI, is respected. A meta-policy constrains a policy by evolution constraints. The policy of the system is then dynamic but respects the meta-policy. An interaction is a violation of the meta-policy if no policy that respects the meta-policy contains this interaction.

Property 4.25: Let $\mathcal{MP}$ be a meta-policy. A MAC policy $\mathcal{POL}$ that respects $\mathcal{MP}$ is noted $\mathcal{POL} \in \mathcal{MP}$. A system guarantees the respect of the meta-policy abuse property, noted $P_{4.25}(SC, \mathcal{MP})$, if:

$$\begin{matrix}\forall sc_1, sc_2 \in SC \\ \forall eo \in IS\end{matrix} \text{, s.a. } it = sc_1 \xrightarrow{eo} sc_2,$$

$$\exists \mathcal{POL} \in \mathcal{MP} \text{ s.a. } it \in \mathcal{POL}$$

In practice, this property is hard to check as the control of dynamic MAC policies is complex and can require a high overhead. Section VII covers this problem.

D. Security properties sum-up

Table II gives an overview of the described security properties. Each property has a precise name, a reference number in the model, a name in the concrete SPL implementation and a short explanation about the property. With these 13 security properties, 13 protection templates will be presented in the next section that gives a real implementation using the concrete SPL language. Nevertheless, it is important to remember that the SPL language can handle more security properties and that only the 13 most important properties are given in this paper.

V. Concrete Security Property Language

Our abstract SPL language needs a concrete syntax that can be compiled as a protection language. Instead of giving the complete concrete syntax, which is pointless since this implements the previous abstract language, this section describes how each template is expressed using our concrete SPL.

Our concrete SPL enables us to define security functions, i.e. templates. Calling those functions corresponds to the instantiation of the protection template to the target Operating Systems. Each template is very powerful. It enables multiple protection properties to be easily expressed using only specific values for the arguments. Moreover, each concrete security property makes it possible to compute the whole set of illegal activities in contradiction with the requested security properties, as presented in section VII.

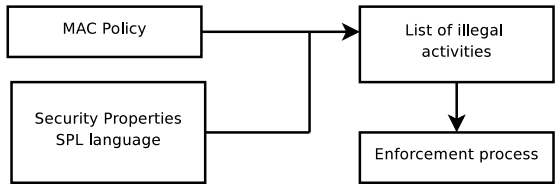


Fig. 3. Enforcement of Security Properties

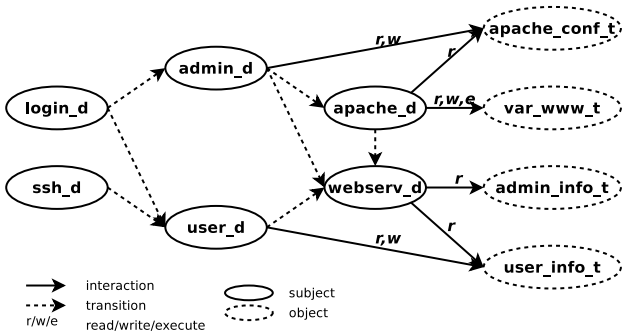


Fig. 4. Example policy for the web server Apache

Those templates are written using a concrete language called SPL, for Security Properties Language, that we have created specifically to be able to quickly write new security properties. A compilation of our concrete SPL language produces all illegal activities i.e. violating one of the security properties [22]. An enforcement process uses those illegal activities and communicates with the SPLinux kernel in order to guarantee the requested properties as represented in figure 3. See [22] to have details about the enforcement architecture that complements the classical DAC and the SELinux protections.

Through the compilation of the function calls, all the violations of the security properties found in the MAC policy are added to the list of illegal activities. These activities can be used to:

- correct the MAC policy if possible;
- to prevent or detect these violations if they occur.

Hereafter, two types of examples are given for each property:

- a simplified example associated with a web server policy, presented on figure 5;
- a complete MAC policy associated with a SELinux host. The whole policy is impossible to represent in this paper as the number of contexts approaches 2,000 and there are more than 100,000 interactions.

A. Apache policy description

The policy presented in figure 4 is a simplified version of the policy that is used for Apache and for the logging of the users in the system. The users can log in the system using a local console under the *login_d* context or via SSH under the *ssh_d* context. Then, a user can transit into the *user_d* context or to the *administrator_d* context in order to run commands from the shell. Note that a SSH user is not allowed to become an administrator in this policy.

TABLE II
SECURITY PROPERTIES SUM-UP

Property	Specificity	Reference	Rule	Goal
Integrity	Objects Integrity	$P_{4.2}(SC_{S1}, SC_{O1})$	`integrity`	A set of contexts must not modify another set of contexts
	Biba Integrity	$P_{4.4}(SC)$	`int_biba`	Biba security model applied to a set of contexts
	Subjects Integrity	$P_{4.6}(SC_S)$	`int_subject`	Non-interference property applied on a set of subjects (process)
	Domain Integrity	$P_{4.7}(chroot_{in})$	`int_domain`	Chroot for a set of contexts (virtual chroot)
Confi-dentiality	Confidentiality	$P_{4.9}(SC_1, SC_2)$	`confidentiality`	A set of contexts must not obtain information from another set
	Bell&Lapadula	$P_{4.11}(SC)$	`conf_blp`	Bell & LaPadulla security model applied to a set of contexts
	Bell&Lapadula restrictive	$P_{4.13}(SC)$	`conf_blpr`	Bell & LaPadulla restrictive security model applied to a set of contexts
	Data Access Consistency	$P_{4.15}(SC)$	`conf_data`	A context must not obtain information that it cannot obtain directly
Privilege Abuse	Duties Separation	$P_{4.20}(SC)$	`duties_separation`	A context must not modify and then execute another context
	No Context Transition	$P_{4.22}(scs_1)$	`no_transition`	A context must not access another context by transition
	Trusted Path Execution	$P_{4.23}(TPE)$	`tpe`	All trusted executables contexts must grouped in a specific set
	Policy Abuse	$P_{4.24}(SC, \mathcal{POL})$	`conformity`	The violation attempts of the local policy are forbidden
	Meta-Policy Abuse	$P_{4.25}(SC, \mathcal{MP})$	`meta-conformity`	The possible violation attempts of the meta-policy are forbidden

The administrator can setup the Apache configuration (read and write permissions on *apache_conf_t*) and can transit to the context *apache_d* in order to launch the service. Then, Apache can read its configuration and use the files under the *var_www_t* context.

A web service context *webserv_d* enables remote users to get information from their account. From that web service, a user authenticates and retrieves personal information (but he cannot modify it) located in the *user_info_t* context. Apache is able to transit in the special context *webserv_d* to obtain the permission to read the information of the user or of the administrator. Of course, a user logged in the system with a shell can also read or write his personal information typed with the *user_info_t* context.

The administrator is also allowed to transit to the *webserv_d* context, because he needs to manage the web service. The administrator is not supposed to run processes in this context that would read the personal information of the users (which represents a violation of confidentiality).

B. Integrity

The integrity property $P_{4.1}(sc1, sc2)$ corresponds to the SPL function of listing 1. This function takes two arguments: a set of subject contexts and a set of object contexts. For each write operation eo between a subject context $sc1$ and an object context $sc2$, the function checks that such an interaction does not exist in the policy. The same check is done considering a write privilege access (c.f. definition 3.8).

Listing 1. Integrity check function of $P_{4.1}(sc1, sc2)$

```
1  define integrity( $sc1 IN SCS, $sc2 IN SCO ) [
2          Foreach $eo IN is_write_like(IS)
3                  SA { $sc1−> { $eo } $sc2 } ,
4                          { not(exist()) };
5          Foreach $eo IN is_write_like(IS)
6                  SA { $sc1 => { $eo } $sc2 } ,
7                          { not(exist()) };
8  ];
```

1) Apache example: Using the policy of figure 4, the administrator can express that he wants to guarantee that any user using SSH will not be able to modify the Apache configuration (as only the local root is supposed to do so). Moreover, the administrator is not supposed to modify the personal information of the user (*user_info_t*).

Listing 2. Integrity security properties for Apache

```
1  integrity( $sc1:="ssh_d", $sc2:="apache_conf_t" );
2  integrity( $sc1:="admin_d", $sc2:="user_info_t" );
```

2) Operating system example: For a whole operating system, the first rule of listing 3 guarantees that no user is able to modify the binaries of the system, the second rule guarantees that no user can modify the configuration files of the system in */etc*. That integrity function prevents any ordinary user from modifying those files. This kind of guarantee cannot be obtained by means of a classical integrity checker like AIDE, TRIPWIRE. Nevertheless, the root user, logged locally on the host is able to modify these files.

Listing 3. Integrity security properties example

```
1  integrity( $sc1:="user_u:user_r:user.∗_t", $sc2:=".∗:.∗:.∗_exec_t" );
2  integrity( $sc1:="user_u:user_r:user.∗_t", $sc2:=".∗:.∗:.∗etc_t" );
```

C. Domain integrity

The integrity property $P_{4.7}(chroot_{in})$ corresponds to the listing function 4. This property allows it to virtually *chroot* a set of contexts.

Listing 4. Domain integrity check function of $P_{4.7}(chroot_{in})$

```
1  define int_domain( $CHROOT IN SC ) [
2          Foreach $eo IN IS, Foreach $sc1 IN $CHROOT, Foreach $sc2
                   IN SC
3                  SA { $sc1 −> { $eo } $sc2 } ,
4                          { $sc2 IN $CHROOT };
5          Foreach $eo IN IS, Foreach $sc1 IN SCS, Foreach $sc2 IN
                   $CHROOT
6                  SA { $sc1 −> { $eo } $sc2 } ,
7                          { $sc1 IN $CHROOT };
8  ];
```

1) Apache example: This property prevents an exploit against Apache enabling a user to gain root privileges, for example.

Listing 5. Domain integrity for apache

```
1  int_domain( $CHROOT:="apache.∗", "webserv_d", ".∗info_t");
```

This property cannot be enforced at start-up. During the boot sequence, the property is not available. It is enforced after the starting of the Apache process. Moreover, it prevents an attacker from restarting the web server. The principle of guaranteeing online properties is described in [22].

2) Operating system example: A user can be *chrooted* in his home directory. Again, this system instantiates this property after the boot sequence and the login of the user (otherwise, the user cannot log into the system). Thus, an exploit in the user domain cannot leave that domain.

Listing 6. Domain integrity security property example

```
1  int_domain( $CHROOT:="user_u:.*:.*" );
```

D. Confidentiality

The confidentiality property $P_{4.8}(sc_1, sc_2)$ corresponds to the function of listing 7. This property checks that no information transfer or flow exists between two contexts.

Listing 7. Confidentiality check function of $P_{4.8}(sc_1, sc_2)$

```
1  define confidentiality( $sc1 IN SCS, $sc2 IN SCO ) [
2          SA { $sc2 > $sc1 },
3                  { not(exist()) };
4          SA { $sc2 >> $sc1 },
5                  { not(exist()) };
6  ];
```

1) Apache example: For Apache, the configuration files of the web server must be confidential for the users entering the system via SSH. Moreover, the administrator must not access the personal information of the users in the context *user_info_t*. In contrast, the users must not access the personal data of the administrator.

Listing 8. Confidentiality for Apache

```
1  confidentiality( $sc1 := "ssh_d", $sc2 := "apache_conf_t" );
2  confidentiality( $sc1 := "admin_d", $sc2 := "user_info_t" );
3  confidentiality( $sc1 := "user_d", $sc2 := "admin_info_t" );
```

2) Operating system example: The goal is to protect the files */etc/shadow* which contain the hashed user passwords, the special file */dev/mem* that gives access to the memory space of the processes. For software such as Firefox, the confidentiality property prevents any leak of information of the cookies and cache files. For example, if the user launches a malware, the malware will be unable to read the Firefox cookies or cache.

Listing 9. Confidentiality security properties example

```
1  confidentiality( $sc1:="user_u:user_r:user.*_t", $sc2:=system_u:
       object_r:shadow_t );
2  confidentiality( $sc1:="user_u:user_r:user.*_t", $sc2:=system_u:
       object_r:memory_device_t );
3
4  confidentiality( $sc1:=user_u:user_r:user_t, $sc2:=user_u:object_r:
       user_mozilla_cookie_t );
5  confidentiality( $sc1:=user_u:user_r:user_t, $sc2:=user_u:object_r:
       user_mozilla_cache_t );
```

E. Data access consistency

The data consistency property $P_{4.14}(sc_1, sc_2)$ corresponds to the function of listing 10. This property allows indirect access only if direct access is allowed.

Listing 10. Data access consistency check function $P_{4.14}(sc_1, sc_2)$

```
1  define conf_data($sc1 IN SC, $sc2 IN SC) [
2          SA { $sc1 >>> $sc2 },
3                  { exist[$sc2 > $sc1] };
4  ];
```

1) Apache example: For Apache, the property of listing 11 controls the access to personal information files for all the users. The user will be able to access to his personal information (*user_info_t*) using the web service, because he is allowed to read it directly. In contrast, he cannot access the personal information of the administrator using the web service (or using a vulnerability of the web service) because the user has no direct permission on the *admin_info_t* context.

Listing 11. Data access for apache

```
1  conf_data( $sc1 := "user_d", $sc2 := ".*_info_t" );
```

2) Operating system example: The property of listing 12 protects the system from the users. For example, if a user becomes root, he will not be able to read or write the */etc/shadow* file, because he has no direct permission on it.

Listing 12. Data access consistency security property example

```
1  conf_data($sc1:="user_u:user_r:user.*_t" , $sc2 := "system_u:.*:.*" );
```

F. Duties separation

The separation of duties property $P_{4.18}(scs)$ corresponds to the function of listing 13. This function controls the modification of a file and prevents its execution. A special security property has been written for bash scripts and is explained in the example of section V-F2.

Listing 13. Duties separation check function of $P_{4.18}(scs)$ and a special security property for interpreted scripts

```
1  define duties_separation( $sc1 IN SC ) [
2          Foreach $eo1 IN is_write_like(IS), Foreach $eo2 IN
               is_execute_like(IS), Foreach $sc2 IN SCO
3                  SA { ($sc1 −> { $eo2 } $sc2 o $sc1 −> { $eo1 }
                       $sc2) } ,
4                    { not( exist() ) };
5  ];
6  define dutiesseparationbash( $sc1 IN SC ) [
7          Foreach $eo1 IN is_write_like(IS), Foreach $eo2 IN
               is_execute_like(IS), Foreach $eo3 IN is_read_like(IS),
8          Foreach $sc2 IN SCO, Foreach $sc3 IN SC,
9          Foreach $a1 IN ACT, Foreach $a2 IN ACT
10                 SA { ( [ $a2 := $sc1 −> { $eo3 } $sc2 ] o ( [ $a1 :=
                       $sc1 −> { $eo2 } $sc3 ] o $sc1 −> { $eo1 }
                       $sc2) ) } ,
11                   { INHERIT($a2 , $a1) };
12 ];
```

1) Apache example: The property of listing 14 prevents the execution of a file written by Apache in */var/www*. It avoids the exploit of a vulnerability against Apache that force Apache to write a script and then to execute it.

Listing 14. Duties separation for Apache

```
1  duties_separation( $sc1 := "apache_d" );
```

2) Operating system example: For a system, the property of listing 15 restricts the rights of a user: if the user downloads a program, he will not be able to execute it. The second rule of listing 15 prevents the writing of a script and its execution. This is a special rule because the binary file that is executed is not the script but the bash shell itself that

reads the modified script. The function *dutiesseparationbash* of listing 4.18 checks that the created bash process is inherited from the user process (to identify the user that launched the script) and prevents the reading of the script by the bash process.

Listing 15. Duties separation security property example

```
1  duties_separation( $sc1:="user_u:user_r:user.*_t" );
2  dutiesseparationbash( $sc1:="user_u:user_r:user.*_t" );
```

G. Trusted path execution

The trusted path execution property $P_{4.23}(TPE)$ corresponds to the function of listing 16. The first function defines the set of binaries that are executables. The second function adds a set of source context that are allowed to execute the binaries and to read the libraries of the set TPE (because an executed binary reads libraries).

Listing 16. Trusted path execution check function of $P_{4.23}(TPE)$

```
1  define tpe( $TPE IN SC ) [
2          Foreach $sc1 IN SCS, Foreach $sc2 IN SC, Foreach $eo IN
               is_execute_like(IS)
3            SA { $sc1 –> { $eo } $sc2 } ,
4                  { ( $sc2 IN $TPE ) };
5  ];
6
7  define tpeuser( $sc1 IN SCS, $TPE IN SC ) [
8          Foreach $sc2 IN SC, Foreach $eo1 IN is_execute_like(IS),
               Foreach $eo2 IN is_read_like(IS)
9            SA { ( $sc1 –> { $eo1 } $sc2 OR $sc1 –> { $eo2 }
                   $sc2 ) } ,
10                 { ( $sc2 IN $TPE ) };
11 ];
```

1) Apache example: The property of listing 17 allows only the execution of the authentication programs, the web server and web service. Even if the MAC policy or the UNIX rights (uog+x) authorize the execution permission for programs in */var/www*, this property will prevent the execution of these programs. If an attacker exploits a vulnerability, he can force Apache to execute malware (downloaded file). The trusted path property prevents the execution of that malware.

Listing 17. Trusted path execution for Apache

```
1  tpe( $sc1 := {"login_d", "ssh_d", "apache_d", "webserv_d"} );
```

2) Operating system example: The first tpe rule of listing 18 defines the classical executables of the system that all users and daemons can use. The second rule $tpeuser$ is more restrictive than the first one: it allows only the users to execute libraries, Firefox, Claws Mail and Open Office. The last rule is a special rule for Open Office that needs the execution of a shell to be able to launch itself.

Listing 18. Trusted path execution security property example

```
1  tpe( $TPE:= { ".*:.*:.*bin_t", ".*:.*:.*exec_t", ".*:.*:.*lib_t", "system_u:
       object_r:ld_so_t" } );
2  tpeuser( $sc1:=user_u:user_r:user_t, $TPE:= { ".*:.*:.*lib_t", "
       system_u:object_r:mozilla_exec_t", "system_u:object_r:
       clawsmail_exec_t", "system_u:object_r:ooffice_exec_t", "
       system_u:object_r:ld_so_t" } );
3  tpeuser( $sc1:=user_u:user_r:user_ooffice_t, $TPE:= { "user_u:user_r:
       user_ooffice_t", "system_u:object_r:bin_t","system_u:object_r:
       ld_so_t","system_u:object_r:lib_t","system_u:object_r:
       ooffice_exec_t","system_u:object_r:ooffice_lib_t","system_u:
       object_r:shell_exec_t" } );
```

H. No context transition

The no context transition property $P_{4.22}(scs_1)$ corresponds to the function of listing 19. This function checks that no transition is possible for the context given in the parameters.

Listing 19. No context transition check function

```
1  define no_transition( $sc1 IN $SCNoTransition ) [
2          Foreach $sc2 IN SCS
3                  SA { $sc1 __> $sc2 },
4                          { not(exist()) };
5  ];
```

1) Apache example: The property of listing 20 guarantees that the web service cannot transit to another context. For example, an attacker that exploits a vulnerability can try to force the web service to transit to the *admin_d* or *user_d* contexts to obtain the privileges associated with these contexts.

Listing 20. No context transition for Apache

```
1  no_transition( $sc1 := "webserv_d" );
```

2) Operating system example: The first rule of listing 21 prevents an attacker from exploiting a vulnerability on Apache, Php or Mysql services to get more privileges on the system (for example, the root privileges). The second rule prevents the user from transitting to another context, such as the root or Apache domain.

Listing 21. Confidentiality property example

```
1  no_transition( $SCNoTransition:= { "system_u:system_r:httpd_t","
       system_u:system_r:httpd_php_t","system_u:system_r:mysql_t" }
       );
2  no_transition( $SCNoTransition:= { "user_u:user_r:user_t" } );
```

VI. Summary of our meta-policy approach

Before moving on to the verification section of dynamic policies in section VII, this section recalls the basics of our meta-policy [3] approach. The meta-policy provides a means to reduce the space of possible MAC policies, adding constraints on the allowed policies. This way, it becomes possible for the security properties for dynamic policies to be verified, as presented in section VII. This section gives a short overview of our meta-policy model and an example of use.

A. Meta-policy model

This section gives a short overview of the concept of meta-policy. Further details are given in [3]. It allows dynamic policies i.e. the evolution of multiple local MAC policies whose states are constrained by modification rules.

1) Initial Policy: Our meta-policy contains an initial policy i.e. a set of Interaction Vectors. The generic rule for enabling a set of Interaction Vectors is as follows:

```
1  enableIV(pattern_S, pattern_O, pattern_oper)
```

$pattern_S$ is a regular expression that designates the subject security contexts that are considered in this rule. $pattern_O$ designates the object security contexts and $pattern_{oper}$ defines the different operations to be allowed between these contexts.

2) Modification rules: Each local policy can evolve according to modification rules. A modification rule includes different elements: the security context allowed to request the modification and the Interaction Vectors. The considered modifications are: addition, removal or change.

1 enableAddIV($sc_{requester}$, ($pattern_S$, $pattern_O$,$pattern_{oper}$))
2 enableModIV($sc_{requester}$, ($pattern_S$, $pattern_O$,$pattern_{oper}$))
3 enableDelIV($sc_{requester}$, ($pattern_S$, $pattern_O$,$pattern_{oper}$))

The security context $sc_{requester} \in SCs$ is the one that can request the modification for $iv = (pattern_S, pattern_O, pattern_{oper})$. For example, $enableAddIV(sc_{admin}, (apache\_(.*), var\_www\_(.*), \{file : (.*)\}))$ can be used to authorize a local administrator sc_{admin} to add the interactions permitting a HTTP process to access any file located in the directory */var/www* with the required privileges. A second set of rules is used to control security context modifications:

1 enableAddSC($sc_{requester}$, $pattern_{SC}$)
2 enableDelSC($sc_{requester}$, $pattern_{SC}$)

For example, $enableAddSC(sc_{admin}, apache\_(.*))$ enables a local administrator to add required Apache server security contexts. Note that similar rules are defined in order to label an "object" with a security context or to modify this labelling.

B. Example

Listing 22. Meta-policy example

```
1  enableIV( login_d, admin_d, transition )
2  enableIV( login_d, user_d, transition )
3  enableIV( ssh_d, user_d, transition )
4  enableIV( user_d, webserv_d, transition )
5  enableIV( admin_d, webserv_d, transition )
6  enableIV( admin_d, apache_d, transition )
7  enableIV( apache_d, webserv_d, transition )
8  enableIV( admin_d, apache_conf_t, { read, write } )
9  enableIV( apache_d, apache_conf_t, { read } )
10 enableIV( apache_d, var_www_t, { read, write, execute } )
11 enableIV( webserv_d, .*_info_t, { read } )
12 enableIV( user_d, user_info_t, { read, write } )
13
14 enableAddSC(webserv_d, .*_info_t)
15 enableAddIV( webserv_d, (webserv_d, .*_info_t, { read } ))
16 enableDelIV( webserv_d, (webserv_d, .*_info_t, { read } ))
```

The listing 22 contains an example of meta-policy. In this example, a user can open a session with *ssh* or local login, but the administrator can only authenticate locally. Both of them can modify objects containing personal information and execute a web-service allowed to access this information. Moreover, the administrator can execute an Apache web-server and modify its configuration. Apache can only read this configuration, but it is allowed to read, write or execute temporary objects. Finally, Apache can also execute the web-service.

C. Flow graphs between the security contexts

Starting from a MAC policy e.g. the initial policy of our meta-policy, we have built two different graphs that will be used to check the security properties (as described in the next section). These graphs are the following:

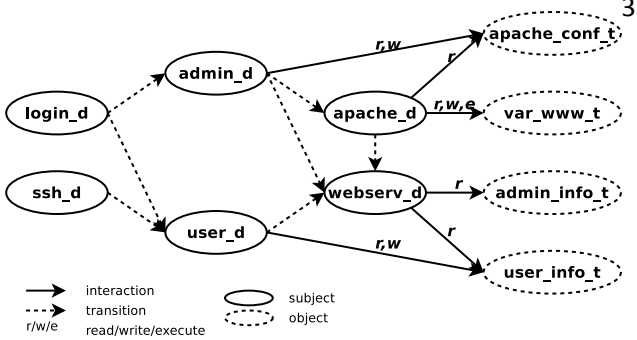


Fig. 5. Interaction graph

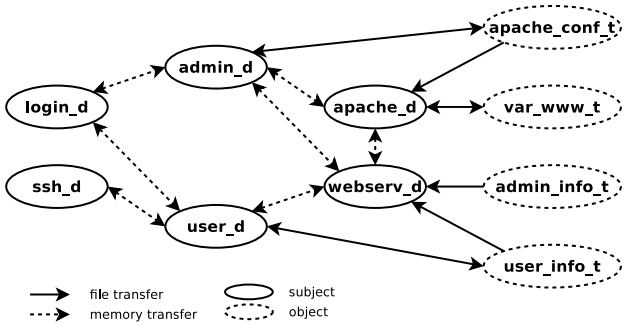


Fig. 6. Information flow graph

1) Interaction Graph: The interaction graph is a representation of all legal interactions allowed by the local policy. An interaction between two security contexts sc_1 and sc_2 is represented in G by an arc , noted $sc_1 \rightarrow_{eo} sc_2$, valuated by the set of authorized operations eo. Figure 5 presents an interaction graph for the previous meta-policy of listing 22. This graph is composed of six subjects and four objects.

2) Flow Graphs: Several flow graphs are computed starting from the previous interaction graph. The direct flows are included within the transitive flows i.e. an indirect path between two contexts. In order to simplify the explanations and focus on the enforcement of dynamic policies, limited details are given about those different graphs. Let us give an explanation for the information flow graph. This graph represents all the allowed transfer of information.

A transfer of information between two security contexts sc_1 and sc_2 is in fact a *read* (*write*) operation from sc_2 to sc_1 (from sc_1 to sc_2). Note that a *transition* from sc_1 to sc_2 allows a transfer of information in both directions. The information graph is a conversion of the interaction graph: the read arcs are flipped, an arc $sc_{s2} \rightarrow sc_{s1}$ is added for each transition arc $sc_{s1} \rightarrow sc_{s2}$.

Figure 6 represents the information flow graph corresponding to the previous example. Another flow graph is computed that includes all the allowed flows of control. A flow of control is a causal sequence including several reading or writing-like operations that provides a path for executing or writing an object.

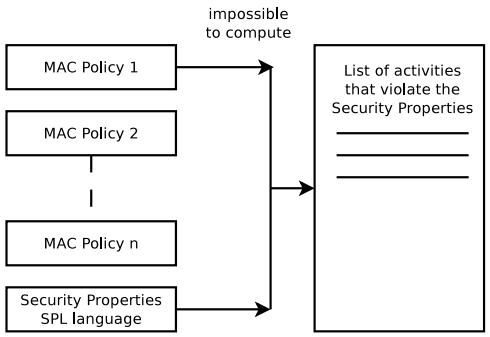


Fig. 7. Impossibility of computing the illegal activities for dynamic policies without a tool based on the meta-policy approach

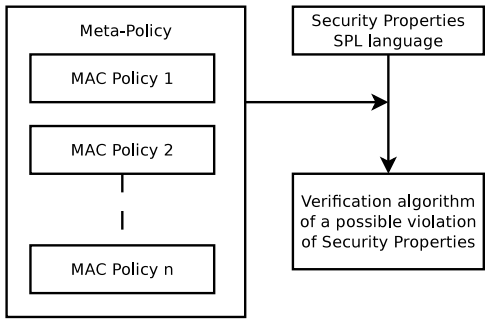


Fig. 8. Strategy of verification for dynamic policies

D. Motivation for a verification algorithm

After presenting the meta-policy model, one can wonder if it is possible for the security properties to be enforced whereas the policies are evolving. If the policy is dynamic, that is if the administrator changes the MAC policy frequently, the generated list of activities that violate the security properties will change. In figure 7, we show that the number of MAC policies is infinite and that the strategy of enforcement of the security properties is no longer possible.

In figure 8, a solution is proposed using the meta-policy that controls all the possible policies, the computation of the infinite number of possible policies is no longer needed. We will directly use the meta-policy itself and the security properties in order to verify if a possible policy could violate one of the security properties. The goal is to get a positive or negative answer.

VII. Verification of MAC Policies

This section presents a method to check the required security properties for dynamic MAC policies.

First, the verification of a security property can be made on a static policy (with no evolution). The algorithm is presented in section VII-A and is straightforward: it consists in a computation of paths on a graph that corresponds to the search for possible information flows, forbidden transitions, etc... The precise algorithm for each verification depends on the security property to be checked.

Second, the aim of this section is to present how to verify security properties for dynamic policies. Using the meta-policy model presented in section VI, we show how to verify a security property in order to 1) have a meta-policy that satisfies the required security properties or 2) compute all the illegal activities.

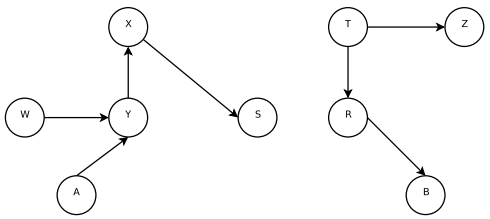


Fig. 9. Policy Graph example

For each algorithm, a concrete example for Apache is given that guarantees the integrity property of the Apache configuration files.

A. Verification of Static Policies

As explained in section VI-C, enforcement of security properties corresponds to the computation of all the illegal paths in the flow graphs. In those graphs, our method produces all the illegal activities associated with a given security property. When deploying the MAC policy on a system, the SPLinux kernel enforces the required security properties for the local Policy [22].

B. Verification of Dynamic Policies

Remember that the meta-policy allows the local policies to evolve. The consequences on the policy graph are detailed below, for each rule of the meta-policy:

- enable[Add|Del]SC($sc_{requester}$, $pattern_{SC}$): creates / deletes nodes in the graph labelled by $pattern_{SC}$
- enable[Add|Del]IV($sc_{requester}$, ($pattern_S$, $pattern_O$,$pattern_{oper}$)): adds / removes operations between all nodes matching $pattern_S$ and nodes matching $pattern_O$, if the operation matches $pattern_{oper}$. If no operation remains, arcs are deleted.
- enableModIV($sc_{requester}$, ($pattern_S$, $pattern_O$,$pattern_{oper}$)): modifies the operation label of arcs which joins nodes matching $pattern_S \rightarrow pattern_O$.

Consequently, enforcement is not limited to the computation of the previous static graphs because those graphs can change in the next version of the policy.

1) Principles of the proposed method: Let us consider the general case to check if there is an information flow between A and B in the graph shown in figure 9 that represents a policy. If we only consider this static graph, obviously, there is no information flow. Now let us consider the following general meta-policy rules:

Listing 23. General meta-policy rules

```
1 enableAddSC(sc_admin, expr_1,)
2 enableAddSC(sc_admin, expr_2,)
3 enableAddIV(sc_admin, expr_1, expr_2, {.*}))
```

$expr_1$ and $expr_2$ are regular expressions matching two sets of nodes. These expressions allow the administrator to

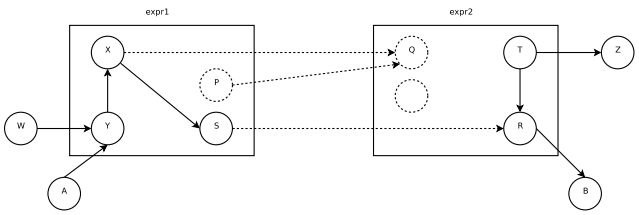


Fig. 10. Graph for listing 23

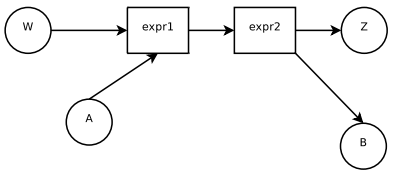


Fig. 11. Meta-nodes computation for graph of listing 23

create nodes, and modify arcs between these created nodes. Eventually, some already existing nodes can be matched by one of the regular expressions. The problem of verification of the possible existence of an information flow between A and B is now more complex because the administrator may add nodes and arcs and create a path between A and B as shown in figure 10.

We propose to solve the problem by using some extra nodes or arcs associated with the meta-policy rules. To take into account these rules, the two sets are grouped in a "meta-node" named $expr_1$ and $expr_2$. We obtain a new graph as shown in figure 11. It shows that a path can occur between A and B: $A \rightarrow expr_1 \rightarrow expr_2 \rightarrow B$.

2) Example: An example for the preceding principle is given below. It is a based on the graph of the policy given in figure 5. The goal is to guarantee that there is no possible access to *apache_conf_t* for the *user_d* context when connecting via SSH. It prevents, for example, an attack that allows a user that have permission to put web pages on the web server and that exploits a vulnerability to modify or to read the Apache configuration. With the model presented in section IV, this can be written as follows:

$$\begin{cases} P_{4.1}(ssh\_d, apache\_conf\_t) & \text{(integrity)} \\ P_{4.8}(ssh\_d, apache\_conf\_t) & \text{(confidentiality)} \end{cases}$$

that is, using the SPL language:

Listing 24. Integrity and Confidentiality for the configuration of Apache
```
1  integrity( $sc1:="ssh_d", $sc2:="apache_conf_t" );
2  confidentiality( $sc1:="ssh_d", $sc2:="apache_conf_t" );
```

Obviously, with the policy of figure 5, such an attack is impossible. Nevertheless, consider the listing 25 that presents a possible meta-policy installed by the administrator. This allows the security contexts to be added for php and to add interactions between the web server and php. Special rules allow php contexts to write new information in the configuration files of Apache and in /var/www.

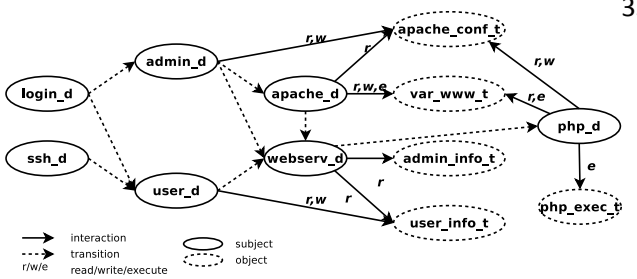


Fig. 12. Updated policy graph with rules of listing 26

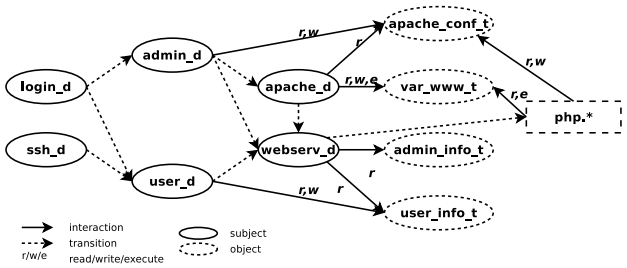


Fig. 13. Updated policy graph with rules of listing 26

Listing 25. Meta-policy for php installation
```
1  enableAddSC(admin_d, php.∗)
2  enableAddIV(admin_d, (php.∗,php.∗, {.∗}))
3  enableAddIV(admin_d, (webserv.∗,php.∗, {.∗}))
4  enableAddIV(admin_d, (php.∗,apache_conf.∗, {r,w}))
5  enableAddIV(admin_d, (php.∗,var_www.∗, {r,w,e}))
```

The listing 26 is the modification performed by the administrator that updates the policy according to the meta-policy. The figure 12 represents the new policy graph resulting from the listing 26 for the policy of figure 5. With this new policy, there is a possible attack from the *ssh_d* context against the *apache_conf_t* configuration files that was not possible before with the policy of figure 5.

Listing 26. Policy modification by admin_d user when installing php
```
1  enableSC(php_d)
2  enableSC(php_exec_t)
3  enableIV(webserv_d,php_d, {transition})
4  enableIV(php_d,php_exec_t, {e}))
5  enableIV(php_d,apache_conf_t, {r,w}))
6  enableIV(php_d,var_www_t, {r,e}))
```

The proposed methodology creates a meta-node that corresponds to the regular expression *php.**. It adds arcs 1) between this meta-node and the nodes *apache_conf_t* and *var_www_t* and 2) between *webserv_d* and the meta-node. The new graph including the meta-node is given in figure 13. In this new graph, it is easy to check if a path exists between *ssh_d* and *apache_conf_t*. The computation of this path will be reported to the administrator as a possible vector of attack.

C. Rules with intersecting regular expressions

If we consider a meta-policy with several rules containing regular expressions, and if some of them intersect each other, then the solution presented before is not correct. This section presents a modified version of the presented methodology to take this case into account.

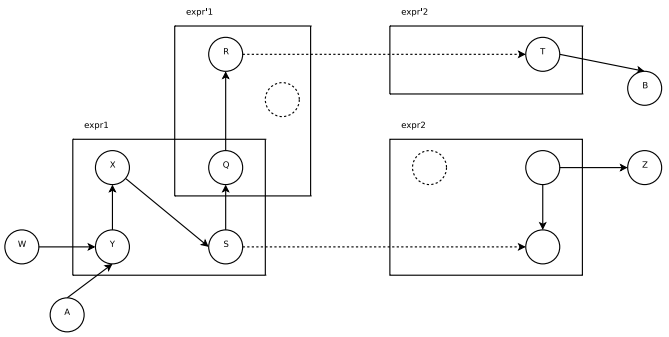


Fig. 14. Example of a meta-policy with two intersecting regular expressions

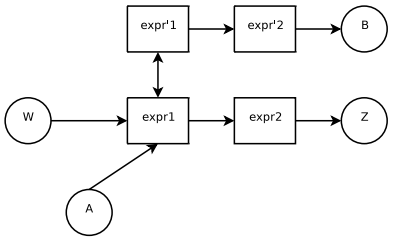


Fig. 15. Aggregated meta-graph

1) Principles of the proposed method: Let us suppose a general meta-policy, presented in listing 27 and represented in figure 14 where the regular expressions $expr_1$ and $expr'_1$ interleave.

Listing 27. Meta-policy rules where $expr_1$ intersects $expr_2$
```
1  enableAddSC(sc_admin, expr_1)
2  enableAddSC(sc_admin, expr_2)
3  enableAddIV(sc_admin, ( expr_1, expr_2, {.*}))
4  enableAddSC(sc_admin, expr'_1)
5  enableAddSC(sc_admin, expr'_2)
6  enableAddIV(sc_admin, ( expr'_1, expr'_2, {.*}))
```

With the preceding methodology, four meta-nodes will be built: $expr_1$, $expr'_1$, $expr_2$, $expr'_2$. Then, because of line 3 (resp. line 6) of the meta-policy of listing 27, an arc is created between $expr_1$ and $expr_2$ (resp between $expr'_1$ and $expr'_2$). But, as $expr_1$ and $expr'_1$ interleave, a possible security context Q can be created inside $expr_1$ and $expr'_1$ at the same time. Thus, a new arc has to be created between $expr_1$ and $expr'_1$ as shown in figure 15.

The last remaining difficulty is to compute the possible intersections between the regular expressions. If the length of the regular expression is limited, then computing the intersection is polynomial with the number of meta-policy rules [30]. If we consider that the length of the regular expression is not limited, the algorithm cannot be computed in polynomial time [31].

2) Example: We now consider a more complex example of meta-policy for the installation of PHP version 5. To improve the security for php5 and to guarantee the properties $P_{4.1}(ssh\_d, apache\_conf\_t)$ and $P_{4.8}(ssh\_d, apache\_conf\_t)$ initially presented in section VII-B2, we deleted some rules, mainly to avoid the php5 process to be able to write the Apache configuration. The listing 28 shows the new meta-policy. Only three rules have been kept as we now consider that php5, in this new version, has no reason to be able to write the Apache configuration. Nevertheless, the corresponding rules have been kept in the policy file for backward compatibility for the php4 process. At this point, it is not easy to see that there is a possibility of attack from the SSH context against the Apache configuration.

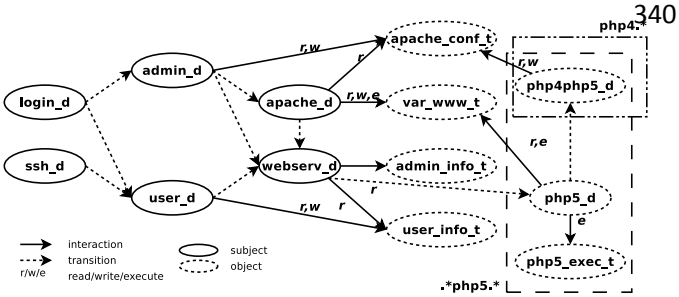


Fig. 16. Updated policy graph with rules of listing 29

Listing 28. Meta-policy for php5 installation
```
1  // Php5 meta−policy rules
2  enableAddSC(admin_d, .∗php5.∗)
3  enableAddIV(admin_d, (webserv.∗,.∗php5.∗, {.∗}))
4  enableAddIV(admin_d, (.∗php5.∗,var_www.∗, {r,w,e}))
5  // Backward compatibility rules for php4
6  enableAddSC(admin_d, php4.∗)
7  enableAddIV(admin_d, (php4.∗,apache_conf.∗, {r,w}))
```

The php5 process seems isolated from the *apache_conf_t* context. But, the administrator can create the rules of listing 29, that respect the meta-policy of listing 28. In this new policy, the *php5_d* process remains isolated from *apache_conf_t*, but a transition is possible to the context *php4php5_d*. This context matches two regular expressions: *.*php5.** and *php4.**. As the meta-policy allows *php4.** to write to the Apache configuration, the last rule of the listing allows *php4php5_d* to do it. The figure 16 shows the resulting contexts and the intersection of regular expressions appears clearly.

Listing 29. Policy modification by admin_d user when installing php
```
1  enableSC(php5_d)
2  enableSC(php5_exec_t)
3  enableIV(webserv_d,php5_d, {transition})
4  enableIV(php5_d,php5_exec_t, {e}))
5  enableIV(php5_d,var_www_t, {r,e}))
6  enableSC(php4php5_d)
7  enableIV(php5_d, php4php5_d {transition})
8  enableIV(php4php5_d,apache_conf_t, {r,w}))
```

Remember that there is a strong hypothesis in this example: the administrator creates the context *php4php5_d* for backward compatibility. It enables a malicious user to write the Apache configuration. The problem is that the administrator could not anticipate the problem when reading the listing 28, because the problem is not trivial. That is why the administrator needs a tool to analyze the meta-policy in order to check the potential violation of the security properties.

D. Algorithm

Our algorithm computes all the possible intersections of the regular expressions. This algorithm have been implemented

in the PIGA tool and technical implementation details can be found in [22]. A set of arcs corresponding to these intersections is added in each graph (between the corresponding meta-nodes). After this first phase, each security property is processed by looking for the security pattern in the corresponding graphs. The result of this second phase is a set of illegal activities that violate the requested security property. Each illegal activity can be static, i.e. without any regular expression, or dynamic, i.e. with at least one regular expression. This second class of illegal activities is what we call a "meta-activity". The PIGA tool is able to compute any illegal activity either static or dynamic i.e a meta-activity expressed using meta-nodes that are designated with regular expressions.

Require: G: the interaction graph i.e. the initial policy
Require: MPR_{sc}: meta-policy rules of type EnableAddSC
Require: MPR_{iv}: meta-policy rules of type EnableAddIV
Ensure: G': the computed meta-graph

1: G'=G.clone()
2: **for** $sc \in MPR_{sc}$ **do**
3: G'.addNode(new Node(sc))
4: **end for**
5: **for** each $rule \in MPR_{iv}, rule = enableAddIV(s,(expr_1, expr_2, perms))$ **do**
6: G'.addNode(v_{expr_1} = new Node($expr_1$))
7: G'.addNode(v_{expr_2} = new Node($expr_2$))
8: G'.addArc(v_{expr_1}, v_{expr_2})
9: G'.addArc(v_{expr_2}, v_{expr_1})
10: **for** each $v \in G'$ **do**
11: **if** v matched by $expr_1$ **then**
12: **for** each $v' \in G'.neighbors(v)$ **do**
13: G'.addArc(v', v_{expr_1}, G'.dir(v',v))
14: **end for**
15: **end if**
16: **if** v matched by $expr_2$ **then**
17: **for** each $v' \in G'.neighbors(v)$ **do**
18: G'.addArc(v', v_{expr_2}, G'.dir(v',v))
19: **end for**
20: **end if**
21: **end for**
22: **end for**
23: **for** each $rule \in MPR_{iv}, rule = enableAddIV(s,(expr_1, expr_2, perms))$ **do**
24: **for** each $rule' \in MPR_{iv}, rule = enableAddIV(s,(expr_3, expr_4, perms))$ **do**
25: **if** $expr_1 \cap expr_3 \neq \emptyset$ **then**
26: **if** $\exists v \in G'$ / v matches $expr_1$ and $expr_3$ **then**
27: G'.addArc(v_{expr_1}, v_{expr_3})
28: G'.addArc(v_{expr_3}, v_{expr_1})
29: **end if**
30: **end if**
31: // Same treatment for ($expr_1$, $expr_4$), ($expr_2$, $expr_3$), ($expr_3$, $expr_4$)
32: **end for**
33: **end for**

This algorithm computes the new meta-graph G' that is used to search paths between two security contexts. The algorithm first creates the meta-nodes associated to the regular expressions into G' (lines 1-9). Then, if the nodes of G' can be included in one of the regular expressions, the arcs are updated to link the meta-nodes (i.e. the regular expressions) to the corresponding neighbours (lines 10-22). In other words, all the matching nodes are grouped within the corresponding meta-nodes. The second part of the algorithms checks if two meta-nodes can intersect each other in which case the two meta-nodes are linked together.

VIII. Experimentation on a Honeypot

In order to secure a high-interaction Honeypot, we defined the security properties of listing 30 in order to give high protection to the system resources and prevent the system from being corrupted. The precise experiment, the technical details, and the discussion about the results are given in [4]. Some of the results are recalled in this section to give to the reader an illustration of the use of the security properties modelled using the SPL language.

Several standard Linux hosts without special protection have been setup to see if attackers can compromise them. Those kind of hosts are compromised in less than a week. Windows hosts are surviving only one day. The compromised hosts have been removed from the experiment after some weeks because it is too much work to maintain and monitor them. In contrast with these standard Linux and Windows hosts, the SPLinux protected honeypot systems have never been compromised during two years of experiment.

The protected hosts that are considered in the presented results are the one that have been analyzed by the PIGA tool [22] which formal algorithm has been described in section VII. The protection is then enforced using our SPLinux kernel which monitors in real time the activities of the system processes. If a process violates a security property, the SPLinux kernel prevents the execution of an interaction of the incriminated activity.

The first property, *integrity*, expresses the fact that no subject context can modify the binary files. The second rule, *confidentiality*, prevents the user from reading any file of the system. The third property, *int_domain*, expresses the fact that no chrooted context is able to interact with any context outside the chrooted domain. The fourth property, *no_transition*, expresses the fact that a system process cannot transit to a bad domain (a blacklist of dangerous contexts). At last, the third property, *duties_separation*, expresses the fact that a context is not able to be executed by another one if it has only just been modified by this same another one.

Listing 30. Security properties for the honeypot

```
1 integrity($sc1:=".*", $sc2=".*:.*:.*_exec_t");
2 confidentiality($sc1:=user_u:user_r:user_t, $sc2:="system_u:object_r
      :.*" );
3 int_domain( $CHR:=".*:.*:.*user.*");
4 no_transition($sc:=user_u:user_r:user_t);
5 duties_separation($sc1:=".*");
```

TABLE III
ILLEGAL META-ACTIVITIES.

		Gateway	User	Initial
Graph	SC	577	3 017	595
	IV	17 684	314 582	18 215
Security Property Rules	integrity	137	9 461	140
	int_domain	16 283	510 215	16 546
	confidentiality	29 510	726 842	29 510
	duties_separation	243	16 405	270
	no_transition	3 555	126 228	3 941
Results	Size of the database	1,1MB	3,6MB	1,1MB
	Number of audit	13 664	44 503	14051
	Computation time	47s	10min31s	52s

Let us consider those properties using several complete meta-policies for two kinds of host. The first kind of host is a gateway and the second kind is a user host.

Each host contains the same initial policy as a part of the meta-policy. That initial policy is processed as a static policy. The results are given in column $Initial$. The PIGA tool finds 140 activities that can violate the $integrity$ property, 16,546 activities violating the int_domain property, 29,510 activities violating the $confidentiality$ property, 270 activities violating the $duties\_separation$ property and 3.941 activities violating the $no\_transition$ property.

The meta-policy of the gateway host includes few modification rules. The PIGA tool processes the corresponding meta-activities, including 137 activities that can violate the $integrity$ property, 16,283 activities violating the int_domain property, 29,510 activities violating the $confidentiality$ property, 243 activities violating the $duties\_separation$ property and 3,555 activities violating the $no\_transition$ property. The resulting meta-graphs are smaller than the initial graph since the update rules of the gateway permit some security contexts and interactions to be removed. Thus, it is consistent to have fewer meta-activities than initial activities.

The meta-policy of the user host includes much larger modification rules. PIGA computes 9,461 activities that can violate the $integrity$ property, 510,215 activities violating the int_domain property, 726,842 activities violating the $confidentiality$ property, 16,283 activities violating the $duties\_separation$ property and 126,228 activities violating the $no\_transition$ property. The user host authorizes many updates of the initial policy in order to support all the classical applications such as the Gnome desktop, Firefox, OpenOffice, etc. The majority of the updates enable new security contexts and interactions to be added for those applications. Thus, PIGA finds many meta-nodes authorizing the user contexts to interact with the system. Since all the properties aim at limiting the flows between the user and the system, it is consistent to have many more illegal meta-activities between the system and the user.

In addition, Table III shows the size of the database of the corresponding illegal activities plus the time needed for computation. At present, computation is not optimal since the solution computes each meta-policy as an independent set of rules. Optimizations could be provided by reusing the results already obtained for previous meta-policies. Finally, the resulting illegal activities associated with the permitted updates of the initial policy can be extracted from the database. Thus, the administrator can isolate the illegal activities that are permitted by the modification constraints. He can either modify the modification constraints or decide that it is consistent. In the later case, each time a new local policy is set up, the corresponding illegal activities will be prevented by our SPLinux kernel. Thus, a satisfactory solution is provided for dynamic MAC policies.

IX. CONCLUSION

This paper introduces a precise formalization of a wide range of integrity and confidentiality properties. A dedicated language has been developed to describe these properties. The abstract Security Property Language (SPL) deals with the activities of the operating system and models complex correlations between these activities that can lead to illegal activities. Even if it easy to define abstract properties, the paper gives 13 pre-defined security templates. Some of them are properties well known in the literature, but others are new security properties.

Moreover, a concrete SPL language is proposed that gives a concrete syntax for enforcing the required properties. Thus, concrete security properties are supported for the predefined template. New templates can be easily defined using the concrete SPL language. A compiler is available for processing mandatory access control policies such as SELinux policies. That compiler computes all the illegal activities that could violate the requested properties. The paper demonstrates how to write concrete properties for protecting systems.

This paper also shows how the security properties are linked to mandatory access control mechanisms. The security properties and the MAC policy of the target host allow the compiler to generate the list of illegal activities. This list is the input of our SPLinux kernel that will enforce the security properties. Thus, a system call fails if illegal activities are recognized.

In the case of dynamic policies, the list of illegal activities is not computable. The paper shows that, with the use of a meta-policy, a policy that constraints the MAC policies, it becomes possible to verify if the meta-policy violates one of the security properties. The verification algorithm is based on a special graph where meta-nodes and meta-arcs are added in accordance with the meta-policy. The result of the algorithm is that the administrator knows if the meta-policy authorizes a policy that violates one of the defined security properties. It is a strong result as it is difficult to be able to give such a guarantee for dynamic policies. Moreover, with a simple meta-policy, all the possible illegal activities can be computed in order to improve the meta-policy.

Finally, the paper gives results about the computation of the illegal activities that violate security properties for two kind of hosts of our honeypot. A large variety of combinations of interactions allow the security properties to be violated. Nevertheless, these properties are enforced by our SPLinux kernel. SPLinux prevents the honeypot hosts from being compromised. During two years of experiments, our honeypot systems have never been compromised whereas a standard linux host have been always compromised in less than one week. It is this experiment that shows the efficiency of our approach. Thus, a global solution is provided. Verification is not mandatory since the administrator can trust the SPLinux kernel to enforce all the required security properties. However, verification is a powerful tool to adjust both the meta-policy or the required security properties.

Future works deal with optimization of our compiler to reuse the verification methods. Indeed, the illegal activities can be pre-computed for all the permitted meta-nodes. Thus, a complete database can be provided for enforcing the host. That database will include both static and dynamic activities to prevent the compilation phase when a local policy is updated. Moreover, ongoing works address DAC systems. Despite the fact that DAC systems are more complicated to protect, we are developing a new enforcement mechanism for those kinds of host. Thus, the security properties will be enforced for both DAC and MAC systems. Finally, extensions will be proposed to enable the SPL language to deal with availability and distributed properties.

REFERENCES

[1] J. Briffaut, J.-F. Lalande, C. Toinard, and M. Blanc, "Enforcement of security properties for dynamic mac policies," in *Third International Conference on Emerging Security Information, Systems and Technologies*, IARIA, Ed. Athens/Glyfada, Greece: IEEE Computer Society Press, June 2009, pp. 114–120.

[2] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman, "Protection in operating systems," *Communications of the ACM*, vol. 19, no. 8, pp. 461–471, 1976.

[3] M. Blanc, J. Briffaut, J.-F. Lalande, and C. Toinard, "Distributed control enabling consistent MAC policies and IDS based on a meta-policy approach," in *Proceedings of the Seventh IEEE International Workshop on Policies for Distributed Systems and Networks*. London, Canada: IEEE Computer Society, Jun. 2006, pp. 153–156.

[4] J. Briffaut, J.-F. Lalande, and C. Toinard, "Security and results of a large-scale high-interaction honeypot," *Journal of Computers, Special Issue on Security and High Performance Computer Systems*, vol. 4, no. 5, pp. 395–404, may 2009.

[5] F. Schneider, "Enforceable security policies," *Information and System Security*, vol. 3, no. 1, pp. 30–50, 2000.

[6] R. Focardi and S. Rossi, "Information flow security in dynamic contexts," in *Proceedings of the IEEE Computer Security Foundations Workshop*. IEEE Computer Society Press, 2002, pp. 307–319.

[7] M. Giunti, "Preventing intrusions through non-interference," in *Proceeding of the IEEE Mexican Conference on Informatics Security*. IEEE Computer Society Press, 2006.

[8] C. Ko and T. Redmond, "Noninterference and intrusion detection," in *IEEE Symposium on Security and Privacy*, 2002, pp. 177–187.

[9] G. Hiet, V. Viet Triem Tong, B. Morin, and L. Me, "Monitoring both os and program level information flows to detect intrusions against network servers," in *Proceedings of the 2nd workshop on MONitoring, Attack detection and Mitigation*, Nov. 2007.

[10] G. Hiet, V. V. T. Tong, and L. Mé, "Policy-based intrusion detection in web applications by monitoring java information flows," in *CRISIS '08: 3nd International Conference on Risks and Security of Internet and Systems*, Oct. 2008.

[11] S. Zdancewic, "Challenges for information-flow security," in *Proceedings of the 1st International Workshop on the Programming Language Interference and Dependence*, 2004.

[12] V. C. Hu, E. Martin, J. Hwang, and T. Xie, "Conformance checking of access control policies specified in xacml," in *Proceedings of the 31st Annual International Computer Software and Applications Conference*, vol. 2. Washington, DC, USA: IEEE Computer Society, 2007, pp. 275–280.

[13] G.-J. Ahn, W. Xu, and X. Zhang, "Systematic policy analysis for high-assurance services in selinux," in *Proceedings of the 2008 IEEE Workshop on Policies for Distributed Systems and Networks*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 3–10.

[14] N. Zeldovich, S. Boyd-Wickizer, E. Kohler, and D. Mazières, "Making information flow explicit in histar," in *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2006, pp. 19–19.

[15] M. Krohn, A. Yip, M. Brodsky, N. Cliffer, M. F. Kaashoek, E. Kohler, and R. Morris, "Information flow control for standard os abstractions," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 6, pp. 321–334, 2007.

[16] P. Efstathopoulos and E. Kohler, "Manageable fine-grained information flow," *ACM SIGOPS Operating Systems Review*, vol. 42, no. 4, pp. 301–313, 2008.

[17] M. L. Damiani, C. Silvestri, and E. Bertino, "Hierarchical domains for decentralized administration of spatially-aware rbac systems," in *ARES '08: Proceedings of the 2008 Third International Conference on Availability, Reliability and Security*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 153–160.

[18] L. Seitz, E. Rissanen, T. Sandholm, B. S. Firozabadi, and O. Mulmo, "Policy administration control and delegation using xacml and delegent," in *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 49–54.

[19] N. Li, Z. Mao, and H. Chen, "Usable mandatory integrity protection for operating systems," in *IEEE Symposium on Security and Privacy*, Berkeley, California, May 2007, pp. 164–178.

[20] Z. Mao, N. Li, H. Chen, and X. Jiang, "Trojan horse resistant discretionary access control," in *Proceedings of the 14th ACM symposium on Access control models and technologies*. New York, NY, USA: ACM, 2009, pp. 237–246.

[21] X. Cai, Y. Gui, and R. Johnson, "Exploiting unix file-system races via algorithmic complexity attacks," in *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 27–41.

[22] J. Briffaut, J. Rouzaud-Cornabas, C. Toinard, and Y. Zemali, "A new approach to enforce the security properties of a clustered high-interaction honeypot," in *Workshop on Security and High Performance Computing Systems*, R. K. Guha and L. Spalazzi, Eds. Leipzig, Germany: IEEE Computer Society, June 2009, pp. 184–192.

[23] ITSEC, "Information Technology Security Evaluation Criteria (ITSEC) v1.2," Technical Report, Jun. 1991.

[24] K. J. Biba, "Integrity considerations for secure computer systems," The MITRE Corporation, Technical Report MTR-3153, Jun. 1975.

[25] R. Focardi and R. Gorrieri, "Classification of security properties (part I: Information flow)," in *Foundations of Security Analysis and Design*. Springer Berlin / Heidelberg, 2001, pp. 331–396.

[26] C. Ko and T. Redmond, "Noninterference and intrusion detection," in *IEEE Symposium on Security and Privacy*. Berkeley CA, United-States: IEEE Computer Society, May 2002, pp. 177–187.

[27] D. E. Bell and L. J. La Padula, "Secure computer systems: Mathematical foundations and model," The MITRE Corporation, Bedford, MA, Technical Report M74-244, May 1973.

[28] D. D. Clark and D. R. Wilson, "A comparison of commercial and military computer security policies," in *The Symposium on Security and Privacy*. IEEE Press, 1987, pp. 184–193.

[29] R. Sandhu, "Separation of duties in computerized information systems," in *Database Security, IV: Status and Prospects*, Halifax, U.K., September 1990, pp. 179–190.

[30] K. Dexter, "Lower bounds for natural proof systems," in *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 1977, pp. 254–266.

[31] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation (2nd Edition)*. Addison Wesley, November 2000.

Analysing security requirements formally and flexibly based on suspicion

Nuno Amálio

Abstract—**Increasingly, engineers need to approach security and software engineering in a unified way. This paper presents an approach to the formal analysis of security requirements that is based on model-checking and uses the concept of suspicion to guide the search for threats and security vulnerabilities in requirements. It proposes an approach to security analysis that favours exploration of a system's state space based on what is abnormal or suspicious to find threats and vulnerabilities, instead of ironclad security proofs that try to demonstrate that a system is secure; as this paper shows, such security proofs can often be misleading. The approach is tested and illustrated by conducting two experiments: one focussing on a system with a confidentiality security property, and another with an integrity security property enforced through the separation of duty principle. One of the advantages of the approach presented here is that threats are derived directly from a model of requirements and no prior knowledge about possible attacks is necessary to perform the analysis. The paper shows that suspicion is an effective search criteria for finding vulnerabilities and security threats in requirements, and that the feedback generated by the analysis helps in elaborating security requirements.**

Index Terms—**Security, requirements, formal analysis, Event-Calculus, planning, confidentiality, separation of duty.**

I. Introduction

Traditional approaches to software engineering and current practice tend to treat security concerns as an after-thought [1]. Security requirements are handled as non-functional requirements and are kept separate from their functional counter-parts until design or implementation-time. This raises problems for the whole software development process because (as demonstrated in this paper) functionality has an impact on security. If security aspects are not treated properly at the requirements phase, then the resolution of the problem will inevitably be deferred but at a much higher cost, which is a well known software engineering problem [2]. This issue can be resolved by integrating security into the requirements engineering phase of the software life-cycle [1], [3]. However, the best way to capture, model and analyse security and system requirements in a unified way is still an open problem.

Security requirements have proved tricky to formulate and reason about [4]. There are several methods for reasoning about properties that a system must satisfy. Traditionally, properties are classified as either *safety* or *liveness* [5], [6]. Safety properties say that something bad must not happen, and liveness properties say that something good must eventually happen. We check safety to ensure that bad states are not reachable, and liveness to ensure that good states are eventually reachable. This is used to check that invariants are preserved, that operations are applicable when certain conditions are met (pre-conditions) and that operations have the desired effect taking the system into a valid state. However, important classes of security requirements are either difficult to express using traditional safety or liveness properties, or they are just not possible to express at all [7], [4], [8].

This paper claims that, from a practical point of view, not only it is important to verify *safety* (that some unsecure state is not reached) and liveness (that the security measures do what is expected from them), but also in finding security vulnerabilities and possible security threats that give attack opportunities to malicious users: we need to look for *what can happen under certain suspicious conditions*. Traditionally, such *possibilistic* properties [7] are hard to formulate and reason about. This paper proposes a practical approach for dealing with such properties.

This paper presents a practical approach to the formal analysis of security requirements based on model-checking, where the search for threats and vulnerabilities in requirements is based on what is *suspicious* from a security point of view. This is inspired by anomaly-based approaches to intrusion detection [9], where the search for intrusions at run-time is driven by *abnormal* (or suspicious) behaviour of system use. The approach presented here takes a formal model of requirements and an analysis goal (a description of suspicious states from the analysis point of view), which are used by the model checker to generate traces of events describing how the analysis goals is reached. Each trace gives a scenario illustrating a possible threat or security vulnerability. The generation of plans is done automatically with tool support. The approach is illustrated with the Event-Calculus temporal logic [10] and the analysis is conducted with tool support using the discrete event calculus reasoner[1] (*decreasoner*).

The remainder of this paper starts by giving a brief introduction to event calculus (section II). Then it presents the approach to the formal analysis of security requirements that is proposed here (section III). After this, the analysis approach is used to conduct two experiments: analysis of a simple health-care system with a confidentiality security requirement (section IV), and analysis of a business system with an integrity requirement enforcing separation of duty (section V). Then, the paper summarises the experimental results that we obtained (section VI), discusses the paper's results (section VII), presents some related work (section VIII), and takes the conclusions.

N. Amálio is with the University of Luxembourg.

[1] http://decreasoner.sourceforge.net/

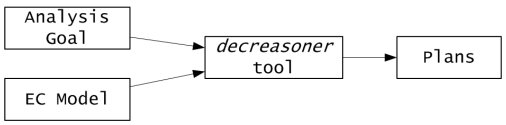


Fig. 1. Planning-based formal analysis. EC model of requirements, and EC description of analysis goal are fed into *decreasoner* tool to obtain a set of plans (scenarios) that achieve the goal.

II. The Event Calculus

Event Calculus (EC) [10], [11], a temporal logic based on first-order predicate calculus designed for common-sense reasoning, enables representation and reasoning about action and change. Its basic ontology comprises *events*, *fluents* and *timepoints*. An *event* is an action that may occur in the world. A *fluent* is a time varying property of the world. A *timepoint* is an instance of time. EC includes a set of basic predicates to describe happening of events, their effects and state of fluents. An EC model is built by describing two types of facts: the fact that an event occurs at a timepoint, and the fact that a property holds at a timepoint [11]. These facts are either true or false.

The basic predicates of EC are as follows:

- $HoldsAt\ (f,t)$ says that fluent f is true at timepoint t.
- $Happens\ (e,t)$ says that event e may occur at timepoint t.
- $Initiates\ (e,f,t)$ says that if event e occurs at timepoint t, then fluent f is true after t.
- $Terminates\ (e,f,t)$ says that if event e occurs at timepoint t, then fluent f is false after t.
- $Initially\ (f)$ says that fluent f holds at timepoint 0.

III. Formal threat analysis by studying reachability

The analysis proposed here is essentially a study of *reachability*: it checks whether certain states are reachable from a model of the requirements. The actual generation of threat scenarios is based on AI planning [12] and uses the *decresoner* tool, which is based on a SAT approach to EC reasoning [13][2]. This is related to what is known in software engineering as *model-checking* [14]: the exploration of all possible states and transitions of a model to determine if a certain property holds or not

Figure 1 depicts the analysis approach followed here. The EC model and EC description of analysis goals are given as inputs to *decreasoner*, which generates a set of plans (or traces) that satisfy the goal. Each plan describes a scenario comprising a sequence of events (a *trace*) that takes the system from the initial state to one of the states described by the goal.

The goal (a predicate) describes a set of states that are interesting from the analysis point of view. Planning generates plans that reach such a state. If there are plans, then the goal is satisfiable: a state as described by the goal can be reached in the model. If no plans can be found, then no state as described by the goal is reachable. If the goal state describes something that should not happen, then the resulting plans (scenarios) describe a sequence of events that reach such a state; thus exposing a way to reach something undesired.

Two strategies are used to formulate the goal. There is a more traditional strategy that does a *safety analysis* by formulating a goal describing states where security is violated and that must not happen; these goals are called *security violation* goals. The other strategy applies suspicion by defining a goal describing *suspicious states deserving investigation* that may expose possible vulnerabilities and threats. These are called *suspicious goals*.

[2]Appendix A shows sample outputs given by decreasoner for the suspicion-based analysis conducted in this paper.

R1	Doctors must be able to access their patient's medical data to provide effective medical care.
R2	A doctor may nominate a substitute who may be able to access the patient's medical data only when main doctor is on leave.

TABLE I
The requirements of Simple Medical Information System.

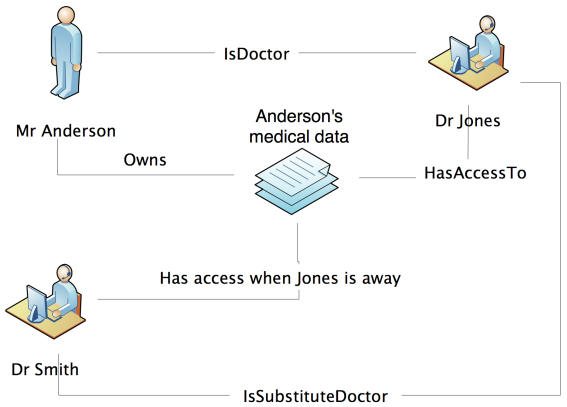


Fig. 2. The SMIS with one patient, Anderson, his doctor Jones, and another doctor, Smith, who is able to replace Jones while he is on leave.

IV. Confidentiality

Confidentiality is about protecting information. It tries to ensure that sensitive information is accessible only to those authorised to access it. Analysis of confidentiality involves checking ways in which confidential information may be accessed by those who are not authorised. Here, confidentiality is studied using a case study of a domain where it is a professional ethical principle: health-care [15].

The case study is a simple medical information system (SMIS) that manages patient information. Due to its sensitive nature, patient information is subject to confidentiality constraints to protect patient's privacy. The requirements of SMIS are summarised in table I. Figure 2 depicts a concrete system scenario of SMIS.

A. EC Model

To satisfy the requirements of SMIS, EC model presented here introduces a protection mechanism based on *credentials*: doctors need to request a credential prior to accessing the data; credentials have a validity period.

The building blocks of an EC model are *sorts*, *events* and *fluents*. EC model of SMIS comprises sort $Delay$, representing

time delays used in credential mechanism, and domain sorts *User*, representing a user of SMIS, *Doctor*, a sub-sort of $User$ that represents doctors using SMIS, and $Patient$, which represents the patients recorded in the system.

EC model of SMIS comprises the following events:

- $AuthoriseAccess\ (d,\ p)$: occurs when doctors (d) request a credential for accessing some patient's data (p).
- $GetMD\ (d,\ p)$: occurs when doctors (d) actually access patient's medical data (p).
- $SetSubstituteDoctor\ (u,\ d_1,\ d_2)$: occurs when a user ($u$) sets some doctor ($d_1$) as substitute of another (d_2).
- $SetDoctorOnLeave\ (u,\ d)$: occurs when a user ($u$) informs system that some doctor (d) is on leave.
- $DoctorNoLongerOnLeave\ (u,\ d)$: occurs when a user ($u$) informs system that a doctor (d) is no longer on leave.

EC model defines the following fluents to hold state:

- $IsDoctorOf\ (d,\ p)$ says who is doctor (d) of some patient (p).
- $CredentialMD\ (d,\ p,\ t)$ says that a doctor (d) has been issued a credential to access data of some patient (p) at time-point t.
- $ExposedToAt\ (d,\ p,\ t)$ says that a doctor (d) has seen the medical data of some patient (p) at time-point t.
- $IsSubstituteDoctor(d_1, d_2)$ says that d_1 is substitute doctor of d_2.
- $OnLeave\ (d)$ says that doctor d is on leave.

EC model of SMIS starts by constraining $Duration$ predicate, which gives actual time associated with delays (of $Delay$ sort). Next EC equation says that $Duration$ relation is functional; each delay has at most one duration (a time point):

$$\forall\ d\ :\ Delay;\ t_1, t_2\ :\ Time\ |\\ Duration\ (d,\ t_1) \wedge Duration\ (d,\ t_2) \Rightarrow t_1 = t_2 \quad (1)$$

Next EC equation says that $Duration$ is total; all delays of the model must have a duration associated:

$$\forall\ d\ :\ Delay\ |\ (\exists\ t\ :\ Time)\ Duration\ (d,\ t) \quad (2)$$

Next EC equation says that $IsDoctorOf$ is a surjective relation; each patient must have a doctor:

$$\forall\ p\ :\ Patient;\ t\ :\ Time\ |\\ (\exists\ d\ :\ Doctor)\ HoldsAt\ (IsDoctorOf\ (d,\ p),\ t) \quad (3)$$

Next EC equation defines predicate $CanAccessMD$, describing conditions ruling doctors' access to patient's data:

$$\forall\ d : Doctor; p : Patient; t : Time\ |\\ HoldsAt\ (CanAccessMD\ (d, p), t)\\ \Leftrightarrow HoldsAt\ (IsDoctorOf\ (d, p), t)\\ \vee\ ((\exists\ d' : Doctor)\ HoldsAt\ (IsDoctorOf\ (d', p), t)\\ \wedge\ HoldsAt\ (IsSubstituteDoctor\ (d, d'), t)\\ \wedge\ HoldsAt\ (OnLeave\ (d'), t)) \quad (4)$$

This says that a doctor can access some patient's medical data at some time point provided doctor is either (a) patient's doctor (fluent $IsDoctorOf$) or (b) substitute doctor of patient's doctor (fluent $IsSubstituteDoctor$) who is on leave at that time (fluent $OnLeave$). This formalises requirements $R1$ and $R2$.

Next EC equation describes how doctors get credentials to access patient's data by describing effect of event $AuthoriseAccess$:

$$\forall\ d\ :\ Doctor;\ p\ :\ Patient;\ t\ :\ Time\ |\\ HoldsAt\ (CanAccessMD\ (d, p), t)\\ \Rightarrow Initiates\ (AuthoriseAccess\ (d, p),\\ CredentialMD\ (d,\ p,\ t),\ t) \quad (5)$$

This says that some doctor gets a credential to access a medical file (fluent $CredentialMD$ is initiated) upon event $AuthoriseAccess$, provided doctor can access patient's data (predicate $CanAccessMD$ defined above). This formalises the scheme of credential-based protection.

Model objects of $Delay$ sort have a duration, as defined by $Duration$ predicate defined above. Next EC equation defines delay $credentialValidity$ representing validity period of a credential:

$$credentialValidity\ :\ Delay \quad (6)$$

Next EC equation defines validity conditions of credentials, which are captured by predicate $HasValidCredential$:

$$\forall\ d\ :\ Doctor;\ p\ :\ Patient;\ t\ :\ Time\ |\\ HoldsAt\ (HasValidCredential\ (d,\ p),\ t)\\ \Leftrightarrow (\exists\ t_2,\ t_3\ :\ Time)\\ HoldsAt\ (CredentialMD\ (d,\ p,\ t_2),\ t)\\ \wedge\ Duration\ (credentialValidity,\ t_3)\\ \wedge\ (t_2\ +\ t_3)\ \geq\ t \quad (7)$$

This says that a credential is valid for the duration period defined by $credentialValidity$.

Next EC equation defines precondition of event $GetMD$, which may happen provided requesting doctor has a valid credential to access requested patient's data:

$$\forall\ d\ :\ Doctor;\ p\ :\ Patient;\ t\ :\ Time\ |\\ Happens\ (GetMD\ (d,\ p),\ t)\\ \Rightarrow\ HoldsAt\ (HasValidCredential\ (d,\ p),\ t) \quad (8)$$

Next EC equation describes how event $GetMD$ initiates (sets to true) fluent $ExposedToAt$ that records exposure to patient's data has been accessed:

$$\forall\ d\ :\ Doctor;\ p\ :\ Patient;\ t\ :\ Time\ |\\ Initiates\ (GetMD\ (d,\ p),\ ExposedToAt\ (d,\ p,\ t),\ t) \quad (9)$$

Next equation constrains fluent $IsSubstituteDoctor$ to be non-reflexive; that is, a doctor may not be set as a substitute of himself:

$$\forall\ d\ :\ Doctor;\ t\ :\ Time\ |\\ \neg\ HoldsAt\ (IsSubstituteDoctor\ (d,\ d),\ t) \quad (10)$$

Next equation describes effect how event $SetSubstituteDoctor$ initiates fluent $IsSubstituteDoctor$ to record that some doctor is substitute of another:

$$\forall\ u\ :\ User;\ d_1,\ d_2\ :\ Doctor;\ t\ :\ Time\ |\\ Initiates\ (SetSubstituteDoctor\ (u, d_1, d_2),\\ IsSubstituteDoctor\ (d_2, d_1), t) \quad (11)$$

Next equation describes how event $SetDoctorOnLeave$ initiates (sets to true) fluent $OnLeave$; some user (u) informs system that some doctor (d) is on-leave:

$$\forall\ u : User; d : Doctor; t : Time\ |\\ Initiates\ (SetDoctorOnLeave\ (u, d), OnLeave\ (d), t) \quad (12)$$

Next equation describes how event $DoctorNoLongerOnLeave$ terminates (sets to false) fluent $DoctorNoLongerOnLeave$; this used so that users u inform system that a doctor (d) is no longer on leave:

$$\forall\, u : User, d : Doctor; t : Time \mid Terminates\ (DoctorNoLongerOnLeave\ (u, d), OnLeave\ (d), t) \quad (13)$$

Next equations describe the initial condition of SMIS:

$$\forall\, d : Doctor; p : Patient; t : Time \mid Initially\ (\neg\ ExposedToAt\ (d, p, t)) \quad (14)$$

$$\forall\, d_1, d_2 : Doctor \mid Initially\ (\neg\ IsSubstituteDoctor\ (d_1, d_2)) \quad (15)$$

$$\forall\, d : Doctor \mid Initially\ (\neg\ OnLeave\ (d)) \quad (16)$$

Above, initially no one has been exposed to medical data (fluent $ExposedToAt$), no doctors are set as substitutes, and there are no doctors on leave.

This completes EC model of SMIS's requirements (table I). Next section, formally analyses this model.

B. Model Analysis

Analysis uses configuration depicted in Fig. 2. There are two doctors, Jones and Smith, and a patient of Jones, Anderson. This is formulated in EC as:

$$jones, smith : Doctor \quad (17)$$

$$anderson : Patient \quad (18)$$

$$Initially\ (IsDoctorOf\ (jones, anderson)) \quad (19)$$

Configuration for model analysis also needs to define duration of $credentialValidity$ delay, which is defined as taking three time-points:

$$Duration\ (credentialValidity, 3) \quad (20)$$

Analysis starts by formulating a security violation goal: it asks whether it is possible to reach a state where patient confidentiality is compromised. In the context of SMIS, this happens when some doctor accesses some patient's data without a valid security credential; the goal expressing this is formulated as:

$$\exists\, d : Doctor; p : Patient; t_1, t_2 : Time \mid HoldsAt\ (ExposedToAt\ (d, p, t_2), t_1) \wedge\neg\ HoldsAt\ (HasValidCredential\ (d, p), t_2) \quad (AG1)$$

For this goal, *decreasoner* does not find any traces (a fragment of decreasoner's output for this goal is given appendix A1). This means that the modelled system cannot reach one of the goal's states. At this point, one could argue that the modelled system is secure because it is not possible to reach an unsecure state, but it isn't so.

Analysis proceeds by applying suspicion. The substitute doctor rule (Requirement $R2$) enables access to patient's data by doctors other than the main patient's doctor. This should occur, but not very often; the situations under which this occurs are suspicious and deserve investigation. Analysis investigates states where those other than the main doctor access the patient's data. This is formulated as the goal:

$$\exists\, d : Doctor;\ p : Patient;\ t_1,\ t_2 : Time \mid HoldsAt\ (ExposedToAt\ (d,\ p,\ t_2),\ t_1) \wedge\neg\ HoldsAt\ (IsDoctorOf\ (d,\ p),\ t_2) \quad (AG2)$$

R3	Only the doctors themselves or one of their administrators are allowed to nominate a substitute, and inform the system of their absence (on-leave) or return to duty.

TABLE II
REQUIREMENTS EMERGING AFTER ANALYSIS OF SMIS INITIAL REQUIREMENTS.

For this goal, *decreasoner* generates traces exposing a security vulnerability, which enables doctors to access patient's data in non-legal ways (see appendix A2 for output generated by decreasoner). In some scenarios, the system behaves as intended: $Jones$ sets $Smith$ as his substitute, at a later time $Jones$ informs system that he is on leave, and so $Smith$ is able to access the medical data (model 5 in appendix A2). Other scenarios are more unusual. In some of them, it is possible that $Smith$ himself requests to be the substitute of $Jones$ (model 1 in appendix A2), and that it is $Smith$ who informs the system that $Jones$ is on-leave (model 1 in appendix A2). Obviously this is strange and could be explored by a malicious doctor determined to get some patient's medical data: (a) he sets himself as substitute of another doctor, then (b) he informs the system that main doctor is on-leave and (c) finally he is able to access the patient's data.

Such scenarios are possible because events $IsSubstituteDoctor$ and $SetDoctorOnLeave$ (equations 10 and 11) are unconstrained: any user may execute them, which introduces a loophole providing an opportunity for access to patient's data in ways that are not intended.

We can confirm the vulnerability by posing an analysis question. We want to know if it is possible that some user can nominate himself as some other doctor's substitute and then to be able to access the data of a patient that is not his own. This is formulated as:

$$\exists\, d_1,\ d_2\ :\ Doctor;\ p\ :\ Patient;\ t_1,\ t_2,\ t_3\ :\ Time \mid Happens\ (SetSubstituteDoctor\ (d_1, d_2, d_1), t_2) \; HoldsAt\ (ExposedToAt\ (d_1, p, t_3), t_1) \; \wedge\neg\ HoldsAt\ (IsDoctorOf\ (d_1, p), t_3)\ \wedge\ t_2 \leq t_3 \quad (AG3)$$

For this goal, descreasoner is able to identify many scenarios, thus confirming the identified vulnerability.

C. Fixing the Model

The analysis' findings are used to elaborate the requirements. We try to remove the vulnerability that has been identified. The requirements that emerge as a result of this elaboration are given in table II; here, $R1$ and $R2$ of table I still hold, and there is new requirement $R3$, which introduces users of type administrators that execute administration tasks on behalf of doctors.

This new requirement needs to be reflected in the EC model. New version of EC model introduces sort $Admin$, subsort of $User$ sort, and which is disjoint from $Doctor$ sort. It also introduces a new fluent:

- $HasAdmin\ (d,\ a)$ indicates administrator user (a) doing administrative tasks on behalf of some doctor d.

This new sort and fluent are used to describe the new requirement. Next EC equation constrains $HasAdmin$ to be a total relation; each doctor must have at least one administrator:

$$\forall\ d\ :\ Doctor;\ t\ :\ Time\ |\\ (\exists\ a\ :\ Admin)\ HoldsAt\ (HasAdmin\ (d,\ a),\ t) \quad (21)$$

Next EC equation constrains $HasAdmin$ to be a surjective relation; each administrator must be associated with a doctor:

$$\forall\ a : Admin;\ t\ :\ Time\ |\\ (\exists\ d\ :\ Doctor)\ HoldsAt\ (HasAdmin\ (d,\ a),\ t) \quad (22)$$

Next EC equation defines predicate $CanDoAdmin$, which indicates users (u) that can do administrative tasks on behalf of some doctor (d):

$$\forall\ u : User;\ d\ :\ Doctor;\ t\ :\ Time\ |\\ HoldsAt\ (CanDoAdmin\ (u,\ d),\ t)\\ \Leftrightarrow\ u\ =\ d\ \vee\ ((\exists\ ad\ :\ Admin\ )\ u = ad\\ \wedge\ HoldsAt\ (HasAdmin\ (d,\ ad), t)) \quad (23)$$

Next EC equations use predicate $CanDoAdmin$ to define a pre-condition for events $SetSubstituteDoctor$, $DoctorOnLeave$ and $DoctorNoLongerNoLeave$. These events may occur provided $CanDoAdmin$ is true; that is, user executing them can do administrative tasks on behalf of affected doctor:

$$\forall\ u\ :\ User;\ d_1, d_2\ :\ Doctor;\ t\ :\ Time\ |\\ Happens\ (SetSubstituteDoctor\ (u,\ d_1,\ d_2), t)\\ \Rightarrow HoldsAt\ (CanDoAdmin\ (u,\ d_1),\ t) \quad (24)$$

$$\forall\ u\ :\ User;\ d\ :\ Doctor;\ t\ :\ Time\ |\\ Happens\ (SetDoctorOnLeave\ (u,\ d),\ t)\\ \Rightarrow HoldsAt\ (CanDoAdmin\ (u,\ d),\ t) \quad (25)$$

$$\forall\ u\ :\ User;\ d\ :\ Doctor;\ t\ :\ Time\ |\\ Happens\ (DoctorNoLongerOnLeave\ (u,\ d),\ t)\\ \Rightarrow\ HoldsAt\ (CanDoAdmin\ (u,\ d),\ t) \quad (26)$$

D. Re-analysing the Model

The analysis configuration is refined by introducing two administrators; one for each doctor:

$$alice,\ sue\ :\ Admin \quad (27)$$
$$Initially\ (HasAdmin\ (jones,\ alice)) \quad (28)$$
$$Initially\ (HasAdmin\ (smith,\ sue)) \quad (29)$$

Under the revised EC model, we re-submit the model to the security violation and suspicion goals:

- For the security violation goal (equation $AG1$ above), we still get no plans (not possible to break confidentiality in an obvious way).
- For the refined suspicious goal (equation $AG3$ above), we no longer get any plans. Meaning that the loophole has been eliminated.
- For the more abstract suspicious goal (equation $AG2$ above) we no longer get obvious security threats, but the results still prompt interesting requirements questions, such as: the system allows doctors to operate the system while they are recorded as being on leave, should this be allowed?

R1	There are two types of users clerks and managers. Managers can performs the tasks that usually the clerks do, but clerks should not usually perform manager's tasks (exception is delegation, below).
R2	Clerks are responsible for starting the refund procedure, and for issuing or cancelling the refund.
R3	The refund shall be issued by a clerk if approved by the managers, or cancelled otherwise.
R4	A refund must by approved by two different managers.
R5	A clerk shall not both prepare and issue or cancel a refund.
R6	Managers can delegate the authority on approval of refunds to one of their administrators.

TABLE III
REQUIREMENTS OF PAYMENT PROCESSING WORKFLOW.

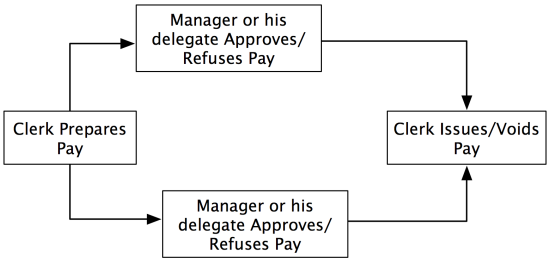


Fig. 3. The payment processing workflow.

V. SEPARATION OF DUTIES

Separation of Duties (SoD) [16], [17] is a security mechanism used to prevent fraud and errors. It aims to prevent a single individual from executing business-critical tasks of some transactions or business processes. SoD requires such tasks to be performed by different users acting in cooperation (e.g by requiring two persons to sign a cheque). Here, SoD is studied using a classical case study: a workflow of payment processing; SoD is used to enable payment authorisations to be performed by different users.

The requirements of this workflow system are given in table III; Fig. 3 depicts underlying workflow. The two tasks involving approval of payments to be carried out by managers, and the tasks *prepare payment* and *issue/void payment*, to be carried out by clerks, are subject to SoD.

A. EC Model

The EC model presented here models a *workflow* as a set of activities. Each activity is made of several alternative tasks; one of the tasks must be carried out to complete the activity. In workflow of Fig. 3, activity *Approve/Refuse Pay* comprises tasks *approve pay* and *refuse pay*, and activity *Issue/Void Pay* comprises tasks *issue pay* and *void pay*. There is always some active activity in some running workflow session.

Workflow tasks are executed by users who have different task execution permissions. Task permissions are defined at the level or rôles; a user is assigned one or more user rôles. In Fig. 3, users that have rôle *clerk* may execute tasks Prepare Pay, Issue Pay and void Pay; users of rôle *manager* may execute tasks Approve Pay and Refuse Pay and all other tasks that clerks do. The model also enables delegation; administrators of managers may also execute tasks on managers' behalf.

The following describes the following elements of EC model: sorts; activities; tasks, rôles and delgation; task execution; and payment processing workflow.

1) Sorts: EC model introduces sorts $Activ$, $Task$, $Session$ and $User$. $Activ$ represents activities of a workflow; $Task$ represents a workflow task that is executed by users; $Session$ represents a running session of a workflow; and $User$ represents users that can execute tasks. Rôles are modelled as sub-sorts of sort $User$; rôles of payment processing workflow are defined below (section V-A5).

2) Activities: Fluent $CurrActiv\ (a,\ s)$ of EC model records current activity (a) of some workflow session (s). Next EC equations constrain fluent $CurrActiv$ to say that there is at most one current activity per workflow session:

$$\begin{aligned}&\forall\ a_1,\ a_2\ :\ Activ;\ s\ :\ Session;\ t\ :\ Time\ |\\ &\quad HoldsAt\ (CurrActiv\ (a_1,\ s),\ t)\\ &\quad HoldsAt\ (CurrActiv\ (a_2,\ s),\ t)\\ &\quad \Rightarrow\ a_1\ =\ a_2 \end{aligned} \tag{30}$$

Next EC equation gives initial state of fluent $CurrActiv$, saying that initially there are no current activities:

$$\begin{aligned}&\forall\ a\ :\ Activ;\ s\ :\ Session\ |\\ &\quad Initially\ (\neg\ CurrActiv\ (a, s)) \end{aligned} \tag{31}$$

To represent workflow configurations in terms of its constituent activities, EC model uses predicates $OccursBefore$. This indicates the ordering of activities in a workflow. Predicate $OccursBefore$ is defined below for payment processing workflow (section V-A5, equation 50).

Predicate $IsStartActiv$ indicates start activity of a workflow; it is defined from predicate $OccursBefore$ as:

$$\begin{aligned}&\forall\ a_1\ :\ Activ\ |\\ &\quad IsStartActiv\ (a_1)\\ &\quad \Leftrightarrow\ \neg\ ((\exists\ a_2\ :\ Activ)\ OccursBefore\ (a_2,\ a_1)) \end{aligned} \tag{32}$$

In model proposed here, separation of duties is enforced on activities. In workflow of Fig. 3, we have a SoD constraint between activities *Approve/Refuse Pay 1* and *Approve/Refuse Pay 2* and *Prepare Pay* and *Issue/Void Pay*. In EC model, such constraints are represented using predicate SoD, which is defined for each workflow that is to be described and analysed (described below for payment processing workflow in equation 51).

To define a precondition for event $StartWrkf$, EC model introduces predicate $Started$, which indicates whether some session has been started or not:

$$\begin{aligned}&\forall\ s\ :\ Session;\ t\ :\ Time\ |\\ &\quad HoldsAt\ (Started\ (s),\ t)\\ &\quad \Leftrightarrow\ (\exists\ a\ :\ Activ;\ t_2\ :\ Time)\ t_2 \geq t\\ &\qquad \wedge\ HoldsAt\ (CurrActiv\ (a,\ s),\ t_2) \end{aligned} \tag{33}$$

Event $StartWrkf$ starts a workflow session. Next EC equation defines pre-condition of event $StartWrkf$; a workflow session may start if it has not already been started:

$$\begin{aligned}&\forall\ s\ :\ Session;\ t\ :\ Time\ |\\ &\quad Happens\ (StartWrkf\ (s),\ t)\\ &\quad \Rightarrow\ \neg\ HoldsAt\ (Started\ (s),\ t) \end{aligned} \tag{34}$$

Next EC equation says how event $StartWrkf$ initiates fluent $CurrActiv$; when a workflow starts, current activity becomes workflow's start activity (predicate $IsStartActiv$):

$$\begin{aligned}&\forall\ s\ :\ Sesstion;\ a\ :\ Activ;\ t\ :\ Time\ |\\ &\quad IsStartActiv\ (a)\\ &\quad \Rightarrow\ Initiates\ (StartWrkf\ (s), CurrActiv\ (a,\ s),\ t) \end{aligned} \tag{35}$$

3) Tasks, rôles and delegation: As said above, tasks are associated with activities. This association is defined through predicate $IsTaskOfActiv$; this predicate is defined for each workflow being described and analysed (it is defined in equation 49, below, for payment processing workflow).

To know whether some task can be executed in a workflow, EC model introduces predicate $IsTaskOfCurrActiv$, which indicates whether some task belongs to the current activity of some workflow session. Next EC equation defines this predicate; it says that some task belongs to current activity of some session if it is a task of session's current activity:

$$\begin{aligned}&\forall\ ta\ :\ Task; s\ :\ Sesstion;\ t\ :\ Time\ |\\ &\quad HoldsAt\ (IsTaskOfCurrActiv\ (ta,\ s),\ t)\\ &\quad \Leftrightarrow\ (\exists\ a\ :\ Activ)\ HoldsAt\ (CurrActiv\ (a,\ s),\ t)\\ &\qquad \wedge\ IsTaskOfActiv\ (ta,\ a) \end{aligned} \tag{36}$$

Predicate $CanDo\ (u,\ t)$ indicates the rôles (r) that are allowed to execute workflow tasks (t). Certain users may delegate their rôles; predicate $MayDelegTo\ (u_1,\ u_2)$ says that some user ($u1$) is allowed to delegate his rôles to another user (u_2). Both $CanDo$ and $MayDelegTo$ are defined for each workflow being described and analysed; it is described below for payment processing workflow in equations 52 and 53.

Event $DelegsTo$ occurs whenever a rôle delegation takes place. Next EC equation defines this event's pre-condition; users delegate to others provided they are allowed to do so:

$$\begin{aligned}&\forall\ u_1, u_2\ :\ User;\ t\ :\ Time\ |\\ &\quad Happens\ (DelegsTo\ (u_1,\ u_2),\ t)\\ &\quad \Rightarrow\ MayDelegTo\ (u_1, u_2) \end{aligned} \tag{37}$$

Fluent $Delegated\ (u_1,\ u_2)$ says that some user (u_1) has delegated to another. Next EC equation says that event $DelegsTo$ initiates (sets to true) fluent $Delegated$ to enable system to keep track of rôle delegations:

$$\begin{aligned}&\forall\ u_1, u_2\ :\ User;\ t\ :\ Time\ |\\ &\quad Initiates\ (DelegsTo\ (u_1,\ u_2), Delegated\ (u_1,\ u_2),\ t) \end{aligned} \tag{38}$$

Next EC equation defines initial condition of fluent $Delegated$; initially, no rôle delegations have taken place:

$$\begin{aligned}&\forall\ u_1,\ u_2\ :\ User\ |\\ &\quad Initially\ (\neg\ Delegated\ (u_1,\ u_2)) \end{aligned} \tag{39}$$

To capture permissions related with delegations, next EC equation introduces predicate $CanExecAsDelegate$, which indicates whether some user can execute a task as delegate:

$$\begin{aligned}&\forall\ u\ :\ User;\ ta\ :\ Task;\ t\ :\ Time\ |\\ &\quad HoldsAt\ (CanExecAsDelegate\ (u,\ ta),\ t)\\ &\quad \Leftrightarrow\ \neg\ CanDo\ (u,\ ta)\\ &\qquad \wedge\ ((\exists\ u_2\ :\ User)\ HoldsAt\ (Delegated\ (u_2,\ u),\ t)\\ &\qquad \wedge\ CanDo\ (u_2,\ ta)) \end{aligned} \tag{40}$$

This predicate is used to define predicate $HasPerm$, which indicates whether some user has the required permissions to

execute some workflow task; this is true if user has required permissions to execute task with his rôle or if he has been delegated the rôle of someone else with required permissions:

$$\begin{aligned}&\forall\, u\, :\, User;\, ta\, :\, Task;\, t\, :\, Time \mid \\ &\quad HoldsAt\ (HasPerm\ (u,\, ta),\, t) \\ &\quad \Leftrightarrow\ CanDo\ (u,\, ta) \\ &\qquad \vee\ HoldsAt\ (CanExecAsDelegate\ (u,\, ta),\, t) \end{aligned} \quad (41)$$

4) Task execution: Event $ExecTask$ happens whenever some workflow task is executed. Next EC equation defines predicate $ExecutedTaskOfActiv$, which indicates whether some user executed a task of some activity in some workflow session:

$$\begin{aligned}&\forall\, u\, :\, User;\, a\, :\, Activ;\, s\, :\, Session;\, t : Time \mid \\ &\quad HoldsAt\ (ExecutedTaskOfActiv\ (u,\, a,\, s),\, t) \\ &\quad \Leftrightarrow\ (\exists\, ta\, :\, Task;\, t_2\, :\, Time)\ t_2\, <\, t \\ &\qquad \wedge\ Happens\ (ExecTask\ (ta,\, u,\, s),\, t_2) \\ &\qquad \wedge\ HoldsAt\ (CurrActiv\ (a,\, s),\, t_2)) \end{aligned} \quad (42)$$

Predicate $ExecutedTaskOfActiv$ is used to define predicate $BreachesSoD$, which indicates whether some user can breach SoD in some workflow session. $BreachesSoD$ is true whenever some user executed some task of some activity for which there is a SoD constraint with current activity in some workflow session; it defined as:

$$\begin{aligned}&\forall\, u\, :\, User;\, s\, :\, Session;\, t : Time \mid \\ &\quad HoldsAt\ (BreachesSoD\ (u,\, s),\, t) \\ &\quad \Leftrightarrow\ (\exists\, a_1,\, a_2\, :\, Activ)\ HoldsAt\ (CurrActiv\ (a_1, s), t) \\ &\qquad \wedge\ (SoD\ (a_1, a_2) \vee\ SoD\ (a_2, a_1)) \\ &\qquad \wedge\ HoldsAt\ (ExecutedTaskOfActiv\ (u, a_2, s), t)) \end{aligned} \quad (43)$$

When event $ExecTask$ happens, current activity changes to be next activity in the workflow; this goes on until the workflow session finishes. There are several restrictions associated with execution of tasks in a workflow; these are modelled as event pre-conditions of event $ExecTask$: (a) the task belongs to the current activity (predicate $IsTaskOfCurrActiv$, equation 36), that (b) the user has the required permissions to execute the task (predicate $HasPerm$, equation 41), and that (c) the execution of the task by the user does not break separation of duties (predicate $BreachesSoD$, equation 43). This is defined in EC by the equation:

$$\begin{aligned}&\forall\, u\, :\, User;\, ta\, :\, Task;\, s\, :\, Session;\, t : Time \mid \\ &\quad Happens\ (ExecTask\ (ta,\, u,\, s),\, t) \\ &\quad \Rightarrow\ HoldsAt\ (IsTaskOfCurrActiv\ (ta,\, s), t) \\ &\qquad \wedge\ HoldsAt\ (HasPerm\ (u,\, ta),\, t) \\ &\qquad \wedge\ \neg\ HoldsAt\ (BreachesSoD\ (u,\, s),\, t) \end{aligned} \quad (44)$$

As said above, when a task is executed the current activity must change. This requires an EC equation defining a terminates predicate to set the current activity to false if there is a current activity. It also requires an initiates predicate to set the current activity to the next activity of the workflow. These are defined as:

$$\begin{aligned}&\forall\, u\, :\, User;\, a\, :\, Activ;\, ta\, :\, Task; \\ &\quad s\, :\, Session;\, t : Time \mid \\ &\quad HoldsAt\ (CurrActiv\ (a,\, s),\, t) \\ &\quad \Rightarrow\ Terminates\ (ExecTask\ (ta,\, u,\, s), \\ &\qquad CurrActiv(a,\, s),\, t) \end{aligned} \quad (45)$$

$$\begin{aligned}&\forall\, u\, :\, User;\, a_1,\, a_2\, :\, Activ;\, ta\, :\, Task; \\ &\quad s\, :\, Session;\, t : Time \mid \\ &\quad HoldsAt\ (CurrActiv\ (a_1,\, s),\, t) \\ &\quad \wedge\ OccursBefore\ (a_1,\, a_2)) \\ &\quad \Rightarrow\ Initiates\ (ExecTask\ (ta,\, u,\, s), \\ &\qquad CurrActiv\ (a_2,\, s),\, t) \end{aligned} \quad (46)$$

5) Payment processing workflow: EC equations above define infrastructure necessary to describe workflows with SoD constraints. The following EC equations actually define the payment processing system workflow of Fig. 3.

We start by defining the rôles of the workflow. Rôles are modelled as sub-sorts of sort $User$; and so we have $User$ subsorts $Clerk$ and $Manager$. Rôle administrator is modelled as the predicate $IsAdminOf$.

Next EC equation defines the tasks and activities of the workflow:

$$\begin{aligned}&prepPay,\ approvePay1,\ approvePay2, \\ &\quad FinPay\ :\ Activ \end{aligned} \quad (47)$$

$$\begin{aligned}&tPrepPay, tApprovePay, tRefusePay, tIssuePay, \\ &\quad tVoidPay : Task \end{aligned} \quad (48)$$

This says that the workflow activities are those identified in Fig. 3, prepare payment ($prepPay$), approve payment ($approvePay1$ and $approvePay2$), and finalise payment ($FinPay$), and that the tasks are also those of Fig. 3, prepare pay ($tPrepPay$), approve payment ($tApprovePay$), refuse payment ($tRefusePay$), issue payment ($tIssuePay$) and void payment ($tVoidPay$).

Next equation defines the relation that exists between tasks and activities by defining predicate $IsTaskOfActiv$:

$$\begin{aligned}&\forall\, ta\, :\, Task; a\, :\, Activ \mid IsTaskOfActiv\ (ta,\, a) \\ &\quad \Leftrightarrow\ (a\, =\, PrepPay\, \wedge\, ta\, =\, tPrepPay) \\ &\qquad \vee\ ((a\, =\, ApprovePay1\, \vee\, a\, =\, ApprovePay2) \\ &\qquad\quad \wedge\ (ta\, =\, tApprovePay\, \vee\, ta\, =\, tRefusePay)) \\ &\qquad \vee\ (a\, =\, IssueOrVoidPay \\ &\qquad\quad \wedge\ (task = tIssuePay\, \vee\, task = tVoidPay)) \end{aligned} \quad (49)$$

This says that prepare payment activity is made of prepare payment task, approve payment activities ($ApprovePay1$ and $ApprovePay2$) are composed of tasks *approve pay* and *refuse pay*, and that $FinPay$ activity is composed of tasks *issue pay* and *void pay*.

Next equation defines the $OccursBefore$ predicate. It Says that $Prepay$ must occur before $ApprovePay1$, which must occur before $approvePay2$, and that $approvePay2$ must occur before $issueOrVoidPay$:

$$\begin{aligned}&\forall\, a_1,\, a_2\, :\, Activ \mid OccursBefore\ (a_1, a_2) \\ &\quad \Leftrightarrow\ (a_1\, =\, PrepPay\, \wedge\, a_2\, =\, ApprovePay1) \\ &\qquad \vee\ (a_1\, =\, approvePay1\, \wedge\, a_2\, =\, approvePay2) \\ &\qquad \vee (a_1\, =\, approvePay2\, \wedge\, a_2\, =\, issueOrVoidPay) \end{aligned} \quad (50)$$

Next equation defines the SoD constraints of the payment processing workflow by defining predicate SoD. It says that there is a SoD constraint between activities $PrepPay$ and

$IssueOrVoidPay$, and between activities $ApprovePay1$ and $ApprovePay2$:

$$\forall\ a_1,\ a_2\ :\ Activ \mid SoD\ (a_1,\ a_2) \\ \Leftrightarrow\ (a_1\ =\ PrepPay\ \wedge\ a_2\ =\ IssueOrVoidPay) \\ \vee\ (a_1\ =\ ApprovePay1\ \wedge\ a_2\ =\ ApprovePay2) \quad (51)$$

Next equation defines the permissions of workflow tasks by defining predicate $CanDo$. Equation says that mnagers can execute any task and that clerks can execute tasks prepare, issue and void payments:

$$\forall\ u\ :\ User;\ ta\ :\ Task \mid CanDo\ (u,\ ta) \\ \Leftrightarrow\ ((\exists\ ma\ :\ Manager)\ u\ =\ ma) \\ \vee\ ((\exists\ cl\ :\ Clerk)\ (ta\ =\ tPrepPay \\ \vee\ ta\ =\ tIssuePay \vee ta\ =\ tVoidPay)) \quad (52)$$

Next EC equation defines the delegation rule of the payment processing workflow. It says that managers may delegate tasks to their administrators:

$$\forall\ u_1,\ u_2\ :\ User \mid MayDelegTo\ (u_1,\ u_2) \\ \Leftrightarrow\ ((\exists\ ma\ :\ Manager)\ u_1\ =\ ma \\ \wedge\ IsAdminOf\ (u_2,\ u_1)) \quad (53)$$

A payment is issued provided both managers approve it. Next EC equation defines predicate $PayApproved$, which defines what it means for a payment to be approved:

$$\forall\ s\ :\ Session;\ t\ :\ Time \mid \\ HoldsAt\ (PayApproved\ (s),\ t) \\ \Leftrightarrow\ ((\exists\ u_1,\ u_2\ :\ User;\ t_2,\ t_3\ :\ Time;\ ta\ :\ Task) \\ u_1 \neq u_2 \wedge\ t_2\ <\ t \wedge t_3 < t \\ \wedge\ ta\ =\ tApprovePay \\ \wedge\ Happens\ (ExecTask\ (ta,\ u_1,\ s),\ t_2) \\ \wedge\ Happens\ (ExecTask\ (ta,\ u_2,\ s), t_3)) \quad (54)$$

This says that a payment is approved provided task $tApprovePay$ has been executed at two different time-points by two different users in context of a workflow session.

Next two EC equations define the constraints associated with tasks $tIssuePay$ and $tVoidPay$. They say that task $tIssuePay$ may be executed provided the payment has been approved, and that task $tVoidPay$ may be executed provided it has not been approved:

$$\forall\ u\ :\ User;\ s\ :\ Session;\ t\ :\ Time \mid \\ Happens\ (ExecTask\ (tIssuePay, u, s),\ t) \\ \Rightarrow\ HoldsAt\ (PayApproved\ (s),\ t) \quad (55)$$

$$\forall\ u\ :\ User;\ s\ :\ Session;\ t\ :\ Time \mid \\ Happens\ (ExecTask\ (tVoidPay,\ u,\ s),\ t) \\ \Rightarrow\ \neg\ HoldsAt\ (PayApproved\ (s), t) \quad (56)$$

This completes EC model of payment processing workflow requirements (table III). Next section analyses this model.

B. Model Analysis

Analysis is conducted in a configuration made of three managers, Bob, John and Martin, and three clerks Sam, Alice and Sue; Sue also works as an administrator for John. This is defined in EC as:

$$bob, john, martin : Manager \quad (57)$$

$$alice, sam, sue : Clerk \quad (58)$$

$$\forall\ u_1,\ u_2\ :\ User \mid IsAdminOf\ (u_1,\ u_2) \\ \Leftrightarrow u_1\ =\ sue \wedge u_2\ =\ john \quad (59)$$

Analysis starts with a security violation goal to know if it is possible to reach a state where SoD is breached. Next EC equation defines what it means to breach SoD; that is, a user executed tasks belonging to activities constrained under SoD:

$$\forall\ u\ :\ User;\ s\ :\ Session;\ t\ :\ Time \mid \\ HoldsAt\ (BreachedSoD\ (u,\ s),\ t) \\ \Leftrightarrow\ ((\exists\ a_1,\ a_2)\ a_1\ \neq\ a_2 \\ \wedge\ HoldsAt\ (ExecutedTaskOfActiv\ (u,\ a_1,\ s),\ t) \\ \wedge\ HoldsAt\ (ExecutedTaskOfActiv\ (u,\ a_2,\ s),\ t) \\ \wedge\ (SoD\ (a_1,\ a_2)\ \vee\ SoD\ (a_2,\ a_1))) \quad (60)$$

This predicate is used to formulate the goal:

$$\exists\ u\ :\ User;\ s\ :\ Session;\ t\ :\ Time \mid \\ HoldsAt\ (BreachedSoD\ (u,\ s),\ t) \quad (AG4)$$

For this goal, *decreasoner* does not find any plans. This means that it is not possible to reach a state where SoD is breached. Again, one could argue that SoD is preserved and the system is secure, but it isn't so.

Analysis proceeds by investigating the suspicious space. Although a user is allowed to execute more than one task in some workflow session, this should not happen very often and is suspicious. The idea is to explore this somehow suspicious or abnormal situation in order to find clues that help in finding security vulnerabilities. First, we define a predicate describing the suspicious system condition of having a user executing two tasks in a workflow session:

$$\forall\ u : User; s : Session; t : Time \mid \\ HoldsAt\ (ExecutedTwoTasks\ (u, s), t) \\ \Leftrightarrow \exists\ ta_1, ta_2 : Task; t_2, t_3 : Time \mid \\ \wedge\ Happens\ (ExecTask\ (ta_1, u, s), t_2) \\ \wedge\ Happens\ (ExecTask\ (ta_2, u, s), t_3) \\ \wedge\ ta_1 \neq ta_2 \wedge t_2 \leq t \wedge t_3 \leq t \quad (61)$$

Since we are interested in scenarios involving complete workflow runs, next EC equation defines predicate $IsWrkfComplete$, which says whether some workflow session is complete or not:

$$\forall\ s : Session; t : Time \mid \\ HoldsAt\ (IsWrkfComplete\ (s),\ t) \\ \Leftrightarrow\ HoldsAt\ (Started\ (s),\ t) \\ \wedge\neg\ ((\exists\ a\ :\ Activ)\ HoldsAt\ (CurrActiv(a,\ s),\ t)) \quad (62)$$

From these two predicates, we define the suspicious goal by describing states where some user executes two different tasks in some workflow run:

$$\exists\ u : User; s : Session; t : Time \mid \\ HoldsAt\ (IsWrkfComplete\ (s), t) \\ \wedge\ HoldsAt\ (ExecutedTwoTasks\ (u, s), t) \quad (AG5)$$

For this goal, *decreasoner* generates interesting plans. We have scenarios where a manager prepares the payment and then approves it, or that he approves and then issues the payment. This happens because managers may act as clerks and there is no SoD constraint between tasks that managers do and clerks do. As this may give a fraud opportunity, it is important to clarify the requirements regarding this issue.

R6'	Managers can delegate authority on approval of refunds to one of their administrators, but when administrators executes such tasks system should consider that they have been executed on behalf of manager and are is manager had executed them.
R7	The same person may perform tasks as either manager or clerk, but not both, in any workflow session.

TABLE IV
REQUIREMENTS RULING THE PROCESSING OF TAX REFUNDS THAT EMERGED AFTER ANALYSIS.

C. Clarifying the requirements

Clarification of the issue exposed by the analysis results in new requirement $R7$ (table IV), which says that a person can execute tasks under at most one rôle in any workflow session. To take this new requirement into account, EC model introduces predicate $RolesRequiredDiffer$, which says which workflow tasks require different rôles to execute them. This predicate is defined for workflow of payment processing as:

$$\begin{aligned}&\forall\, ta_1,\ ta_2 \,:\, Task \mid \\ &\quad RolesRequiredDiffer\ (ta_1,\ ta_2) \\ &\quad \Leftrightarrow\ (ta_1 = tPrepPay \vee\ ta_1 = tIssuePay \\ &\qquad \vee\, ta_1 = tVoidPay) \\ &\qquad \wedge\, (ta_2 = tApprovePay \ \vee\ ta_2 = tRefusePay) \end{aligned} \quad (63)$$

This predicate says that the roles required for tasks $tPrepPay$, $tIssuePay$ and $tVoidPay$ is different for those of tasks $tApprovePay$ and $tRefusePay$.

Predicate $RolesRequiredDiffer$ is used to state the required requirement by constraining event $ExecTask$. Next EC equation describes this constraint by saying that if some user executes two different tasks then roles required to execute them must not differ:

$$\begin{aligned}&\forall\, u \,:\, User;\ s \,:\, Session;\ ta_1,\ ta_2 \,:\, Task; \\ &\qquad t_1,\ t_2 \,:\, Time \mid \\ &\quad Happens\ (ExecTask\ (ta_1,\ u,\ s),\ t_1) \\ &\quad \wedge\, Happens\ (ExecTask\ (ta_2,\ u,\ s),\ t_2) \\ &\quad \wedge\, ta_1 \neq ta_2 \\ &\quad \Rightarrow\ \neg\, RolesRequiredDiffer\ (ta_1,\ ta_2) \end{aligned} \quad (64)$$

D. Re-Analysing the model

After the fix, the vulnerability identified above that that could give a fraud opportunity is no longer allowed. We resubmit the analysis goal above and decreasoner no longer generates scenarios with those possible fraudulent behaviours.

Analysis turns to delegation, which is known to generate security vulnerabilities. Someone executing a task on behalf of someone is legal but suspicious and deserves investigation. Again, the idea is too look in behaviours involving delegation for clues on possible system vulnerabilities. We introduce a predicate to say whether some user executed some task as delegate; next two EC equations define this predicate:

$$\begin{aligned}&\forall\, u : User; ta : Task; s : Session; t : Time \mid \\ &\quad HoldsAt\ (DelegExecutedFor\ (u, ta, s), t) \\ &\quad \Leftrightarrow\ ((\exists\, t_2 \,:\, Time)\ t_2 < t \\ &\qquad \wedge\, DelegExecutedForAt\ (u,\ ta,\ s,\ t_2)) \end{aligned} \quad (65)$$

$$\begin{aligned}&\forall\, u \,:\, User;\ ta : Task;\ s \,:\, Session;\ t \,:\, Time \mid \\ &\quad DelegExecutedForAt\ (u,\ ta,\ s,\ t) \\ &\quad \Leftrightarrow\ ((\exists\, u_2 \,:\, User) \\ &\qquad Happens\ (ExecTask\ (ta,\ u_2, s), t) \\ &\qquad \wedge\, HoldsAt\ (Delegated\ (u,\ u_2),\ t) \\ &\qquad \wedge\, HoldsAt\ (CanExecAsDelegate\ (u_2,\ ta),\ t)) \end{aligned} \quad (66)$$

Above, predicate $DelegExecutedFor$ says whether some task was executed by delegate for some user. This is defined from predicate $DelegExecutedForAt$, which says whether some user executed the task as delegate.

Goal is defined from $DelegateExecutedFor$ by describing states of complete workflow runs where someone executes a task on behalf of someone else. This results in the goal:

$$\begin{aligned}&\exists\, u : User; ta, : Task; s : Session; t : Time \mid \\ &\quad HoldsAt\ (IsWrkfComplete\ (s),\ t) \\ &\quad \wedge\, HoldsAt\ (DelegExecutedFor\ (u,\ ta,\ s),\ t) \end{aligned} \quad (AG6)$$

Plans generated by *decreasoner* result in what is normally expected under delegation (someone executes a task on behalf of someone else), but they also result in plans that may be possible frauds: a delegate approves a payment on behalf of the manager and the same manager also approves the same payment.

E. Clarifying and elaborating the requirements

From this, we elaborate the requirements, and we get $R6'$ (table IV) an elaboration of requirement $R6$. This says that system must consider tasks executed by administrators acting as delegates as if they had been executed by the managers themselves.

To accommodate this new requirement, we introduce the predicate $ExecutedTask$, which indicates whether some user executed some task, either directly or indirectly through a delegate. This is defined as:

$$\begin{aligned}&\forall\, u \,:\, User;\ ta \,:\, Task;\ s \,:\, Session;\ t \,:\, Time \mid \\ &\quad HoldsAt\ (ExecutedTask\ (u,\ ta,\ s),\ t) \\ &\quad \Leftrightarrow\ (\exists\, t_2 \,:\, Time)\ t_2 < t \\ &\qquad \wedge\, ExecutedTaskAt\ (u,\ ta,\ s,\ t_2)) \end{aligned} \quad (67)$$

$$\begin{aligned}&\forall\, u \,:\, User;\ ta \,:\, Task;\ s \,:\, Session;\ t \,:\, Time \mid \\ &\quad ExecutedTaskAt\ (u,\ ta,\ s,\ t) \\ &\quad \Leftrightarrow\ Happens\ (ExecTask\ (ta,\ u,\ s),\ t) \\ &\qquad \vee\, DelegExecutedForAt\ (u, ta, s, t) \end{aligned} \quad (68)$$

Predicate $ExecutedTask$ defined above is used to redefine predicate 'ExecutedTaskOfActiv' equation 42). New formulation of this predicate is defined by EC equation:

$$\begin{aligned}&\forall\, u \,:\, User;\ a \,:\, Activ;\ s \,:\, Session;\ t : Time \mid \\ &\quad HoldsAt\ (ExecutedTaskOfActiv\ (u,\ a,\ s),\ t) \\ &\quad \Leftrightarrow\ (\exists\, ta \,:\, Task;\ t_2 \,:\, Time)\ t_2 < t \\ &\qquad \wedge\, HoldsAt\ (ExecutedTask\ (u, ta, s), t_2) \\ &\qquad \wedge\, HoldsAt\ (CurrActiv\ (a,\ s),\ t_2)) \end{aligned} \quad (42')$$

In this revised EC model, the possible fraudulent behaviour identified above is no longer allowed.

VI. EXPERIMENTAL RESULTS

In both experiments presented above, formal analysis verified a straightforward safety security property, which could mislead analysts in concluding that an unsecure state would not be reached in the modelled system. However, suspicion-based analysis demonstrated that the modelled systems were in fact not secure.

Section IV analyses a simple medical information system that includes a confidentiality requirement. Following the

Case Study	Anaysis Goal	Time
SMIS	Security Violation (AG1)	3.6s
SMIS	Suspicion goal 1 (AG2)	4.9s
SMIS	Suspicion goal 2 (AG3)	12.6s
SMIS	Security Violation (AG1), after fix	14.s
SMIS	Suspicion goal 1 (AG2), after fix	15.8s
SMIS	Suspicion goal 2 (AG3), after fix	21.1s
Workflow	Security violation (AG4)	235.9s (3.9m)
Workflow	Suspicion goal 1 (AG5)	600.7s (10.0m)
Workflow	Suspicion goal 1, after fix (AG5)	476.9s (7.9m)
Workflow	Suspicion goal 2 (AG6)	619.2s (10.3m)
Workflow	Suspicion goal 2 (AG6), after fix	611.11s (10.2m)

TABLE V
RUNNING TIMES FOR ANALYSIS OF EC MODELS WITH *decreasoner*. TABLE INDICATES CASE STUDY, ANALYSIS GOAL AND TIME TAKEN TO GENERATE PLANS.

traditional route of safety analysis, it was not possible to find ways in which confidentiality would be compromised: without a valid credential it would not be possible to obtain the patient's medical data. Analysis based on suspicion then uncovered a security vulnerability (or loophole) that would enable a malicious user to obtain the required credentials in a non-legal way.

Section V analyses a business process whose security requirements included two integrity requirements enforced through SoD. Again, SoD could be breached, but not in an obvious way. Following the traditional safety analysis route, we checked that it was not possible that the same user would be able to execute two different tasks protected by SoD. Analysis-based on suspicion, however, uncovered several problems: the same user could execute different tasks in a workflow session under different roles, and delegation introduced a *loophole* that would enable users to indirectly breach SoD.

Table V presents the running times of the formal analysis based on planning with *decreasoner*[3]. For each case study, it shows how much time it took to carry out the analysis for each analysis goal. We can see that the analysis of the workflow model of section V takes substantially longer than SMIS model because it is more complex.

VII. DISCUSSION

This paper proposes *suspicion* as a concept driving the analysis of security requirements. Through experiments, it argues that, from a practical point of view, in security the interesting question is not only to verify the in-existence of a state compromising some security property (safety), but also to look for what is suspicious in order to find security vulnerabilities and threats. The experiments conducted in the context of the EC temporal logic, planning and the *decreasoner* tool. They demonstrate the usefulness of suspicion. The traditional safety analysis route, which checks whether some security property is violated, would not expose any security issues; this can mislead analysts in concluding that the system being analysed is secure. Analysis based on suspicion uncovered security vulnerabilities and threats; such findings drive elaboration of the requirements.

One of the advantages of the approach presented here is that security threats can be derived directly from a model of requirements. The analysis that does not need prior knowledge about possible attacks to the modelled system, and so no need to enrich the model with attacker or intruder models. Instead, using the suspicion-based approach proposed here, it is possible to derive threats from a requirements model by posing the model questions based on what is suspicious.

All the vulnerabilities exposed by suspicion-based analysis are related with delegation or passing of capabilities, which are known in security as non-interference properties [18]. The formulation and verification of such properties have proved to be far from trivial [4]. The analysis conducted in this paper confirms that delegation can be trick and hard to get right. Suspicion-based analysis helped in identifying security problems with delegation, and in elaborating the security requirements in order to eliminate such problems. The paper also shows that proof of a straightforward safety property related with security does not deem a system secure. Often, as shown in this paper, the *secure* question is more involved and requires more in-depth knowledge of the requirements. As this paper shows, it can be more revealing to analyse the system in order to explore the consequences of the requirements, which leads to a better understanding of the security needs and issues of the modelled system, rather than trying to prove that a system is secure. For the delegation-related issues explored in these two experiments, such proofs are far from trivial.

The approach presented here generates automatically possible scenarios of misuse (threats) from a statement describing some security violation or suspicious condition (the goal). This provides a flexible and illuminating scheme to the analysis of security requirements. Rather then finding themselves possible threats, analysts describe instead what would constitute a violation of security or a suspicious system condition. Analysis goals require an understanding of the requirements domain, and should be described with some security asset in mind.

The security vulnerabilities exposed by the analysis illustrate the sort of vulnerabilities that attackers exploit to intrude into today's software systems. The vulnerabilities identified in the health-care system give insiders the opportunity to perpetrates attacks on the system; the *insider threat* has been identified as one of the main sources of attacks in the medical domain [15]. Once a source of threats is identified in the requirements, two decisions can be made: (a) introduce further constraints by elaborating the requirements so that the source of threats is eliminated, or (b) do nothing in terms of requirements, but take the problem into account in terms of run-time intrusion and threat detection which then has to judge whether some uses of the system are malicious or not. The latter must be considered because it is not possible to eliminate all possible security threats; doing so could result in a system design that is rigid and over-constrained. The approach presented here enables the detection of threats or vulnerabilities in the system, which also constitutes valuable information for run-time intrusion and threat detection.

The experiments conducted here confirm the importance

[3]Model analysis carried out on an Apple *iMac*, with a 2.93 Ghz Intel Core 2 Duo processor and 4GB memory RAM.

of modelling and analysing security together with system requirements. Both case studies show how a functionality of the system, delegation, have a serious impact on security and how it was necessary to further elicit and elaborate the requirements in order to eliminate threats.

Formal security analysis with tool support is capable of exposing many unexpected situations, providing a level of assurance not guaranteed by semi-formal approaches. The drawback of the analysis with *decreasoner* lies in the efficiency of the tool: as models get more complex, the solution to analysis problems take more time to the point that the analysis becomes unpractical. The workflow model of section V is a simplification of an earlier model to enable practical analysis. As usual, the secret is in getting the right abstraction in order to analyse the property of interest.

It is interesting to comment on the usability of the approach presented here. The process of defining analysis goals may require domain knowledge and skill in building and analysing models. However, the process can be partially or fully automated by following the pattern-based approach proposed of [19], which uses the *Formal Template Language* [20], [21] to represent patterns of EC models together with their associated security monitoring goals. [19] defines templates security violation goals, but patterns of suspicious goals can also be defined if we know in advance what can arouse suspicion. In our experiments, *delegation* was the focus of our suspicious goals; this is something that can be known in advance and captured using patterns. Following [19], we can have goals that capture what is known to arouse suspicion; actual suspicious goals would then be automatically generated from templates. [19] also uses UML models to enable intuitive requirements modelling; the same approach can also be used to enhance the usability of the approach proposed here.

VIII. Related Work

This paper is a revised and extended version of the work presented in [22]. It shows in detail the EC models that are used to illustrate the analysis based on suspicion with EC, and provides a more in-depth discussion on the verification of security properties, such as the one explored in the paper.

The results of this paper argue against ironclad proofs of security and how one needs to be careful in interpreting formal demonstrations of security properties. This theme is not new; in [23], McLean refutes what used to be a widely held belief: that the security model of Bell and LaPadula [24] and its *basic security theorem* would capture the essence of security and that implementations following it would be secure[4]. This refutation was done by stating a similar theorem for a model that is clearly not secure. Both experiments of this paper demonstrated that the modelled system would not breach a straightforward safety property of security (confidentiality and separation of duty), and how that could mislead analysts in concluding that the system was secure. However, more flexible means of analysis exposed vulnerabilities showing that the systems being analysed were in fact not secure; the paper suggests analysis lead by what is suspicious in order to find security vulnerabilities in models of requirements.

The case studies used in this paper illustrate behaviours that are usually tricky to be verified using *safety* or *liveness* arguments. Most vulnerabilities identified in sections IV and V are related to delegation, which has traditionally proved to be tricky; [18] introduces *non-interferance*, a confidentiality policy that deals with delegation-based functionality. The verification of non-interference is far from trivial, and requires a simplified model of a system that is difficult to obtain when modelling requirements. Such behaviours or properties have also been termed *possibilistic* [7] and it is known that certain security policies cannot be expressed using safety or liveness properties represented as sets of traces [7], [8]. It is also known that, in general, non-interferance policies cannot be expressed as safety or liveness properties [8]. In [8], the authors propose *hyperproperties*, which are defined as sets of properties (sets of sets of traces), to represent what is not normally captured with traditional properties. This paper provides a pragmatic approach to formally find security vulnerabilities involving delegation in models of security requirements.

There has been substantial interest on security requirements threat analysis [25], [3], [26], [27]. In [26], [27], specifiers need to explicitly identify scenarios or use-cases of abuse and misuse; here, such scenarios are generated automatically from a description of suspicious states (the goal).

[25] proposes a method based on the more flexible abuse frames, specifying undesirable phenomena that the system should prevent from happening. The approach presented here enables the specification of such undesirable phenomena as goals (here called security violation goals). However, it does not only consider what should not happen, but also considers specification of flexible suspicious conditions that (as shown here) have the potential of exposing unnown threats.

[3] proposes a goal-based method that is similar to the approach presented here. Security goals, such as confidentiality, integrity and availability, are negated to obtain goals that specify what should not happen (our security violation goals). Then, these negations are refined to obtain more flexible goals. The approach presented here is more flexible in that it does not only allow specification of goals that come from negation and refinement security goals, but also leaves the specifier the flexibility of defining what constitutes a suspicious condition. Since it is based on tool support, the specifier can use the feedback coming from the tool to either refine existing analysis goals or specify entirely new ones. Essentially, the work presented here is complementary to the body of work on security requirements threat analysis. It explores automated formal analysis (missing in the works above) and provides experimental evidence to the usefulness and effectiveness of security threat analysis.

Suspicion is ubiquitous in *intrusion detection* [9]. Anomaly-based approaches to intrusion detection [9] are so called because the search for intrusions is driven by abnormal (or suspicious) behaviour patterns of system use. [28] proposes the inclusion of suspicion as a concept driving the models of *misuse-based* intrusion detection; it proposes models based on suspicious activities that may lead to an attack, as opposed

[4]This was not claimed by the authors of [24], but others that interpreted their work believed that that was the case.

to models based on actual attacks. Instead, the approach presented here detects abuse by identifying suspicious states in a model of normal system behaviour.

The approach presented here emerges from its preceding work on threat detection [19]. [19] uses an EC model of requirements and planning to find threats at run-time. In [19], however, the goals used to detect threats are more rigid than the ones used here; they say with absolute certainty whether there is an attack or not when the goal is satisfied. In the approach presented here, there is no certainty of attack if a suspicious goal is satisfied, the plans that reach the goal just give us threats (possible attacks). It would be possible to incorporate our approach based on suspicion as a strategy in looking for threats at run-time. Then, the probabilistic component of such a system (like the one of [19]) would try to assign a probability to the computed threats. Here we use suspicion to look for threats in requirements to avoid systems with security vulnerabilities.

The approach presented here analyses requirements automatically using a tool based on SAT-solving. The advantage of this method with respect to other approaches based on theorem-proving [29], [21] is that the reasoning is automatic, avoiding the need for user-intervation as it is usually the case with theorem proving. The disadvantages are that only a portion of the state space is analysed, and that the models that can be handled need to be small; this problem can be mitigated by using abstraction to produce smaller models enabling analysis of property of interest. Another disadvantage to the work in [29], [21] is that there is no visual description of requirements and properties to check; the user needs to be an expert in the formal language (here EC).

IX. Conclusions

This paper proposes a practical approach to the formal analysis of security requirements based on planning guided by the concept of suspicion. One of the advantages of the approach presented here is that threats can be detected directly from a requirements model, where no prior knowledge about possible attacks is needed to perform the analysis. Instead, the analysis derives threats automatically by posing the model questions based on what is suspicious. The approach was illustrated using the EC and the *decreasoner tool* by performing two experiments: one involving a simple health-care system with a confidentiality requirement and another a business system with an integrity requirement enforced through SoD. It showed that the more obvious way of analysing security, by doing the traditional safety verification would not give any useful results: following this path analysts could be mislead in concluding that an unsecure state could not be reached. However, it was through more flexible analysis based on suspicion that we could obtain useful results exposing subtle security vulnerabilities.

The main contributions of this paper are: (a) the proposal of suspicion as a driving concept in the analysis of security requirements, and (b) the experimental confirmation that, from a practical point of view, it is important to use flexible criteria for the security analysis in order to find vulnerabilities in system requirements (suspicion was proposed as basis for such criteria). The paper also provides experimental evidence to certain claims made in the security requirements literature: (a) it confirmed that it is important to model security requirements together with other functional requirements because functionality impacts on security; (c) it confirmed the existence of security relevant phenomena that is hard or impossible to capture as safety or liveness properties; and (d) demonstrated the importance of formality and tool support and usefulness of automated reachability analysis of requirements.

References

[1] P. T. Devanbu and S. Stubblebine, "Software engineering for security: A roadmap," in *The Future of Software Engineering*. ACM, 2000, pp. 227–239.

[2] B. W. Boehm, "Software engineering," *IEEE Transactions on Computers*, pp. 1266–1241, 1976.

[3] A. van Lamsweerde, "Elaborating security requirements by construction of intentional anti-models," in *Proc. ICSE'04*, 2004, pp. 148–157.

[4] J. Rushby, "Security requireemnts specifications: How and what? (extended abstract)," in *Symp. on Requirements Engineering for information security*, 2001.

[5] L. Lamport, "Proving the correctness of multiprocess programs," *IEEE Trans. on Software Engineering*, vol. 3, no. 2, pp. 125–143, 1977.

[6] B. Alpern and F. B. Schneider, "Recognizing safety and liveness," *Distributed Computing*, vol. 2, pp. 117–126, 1987.

[7] J. McLean, "A general theory of composition for a class of "possibilistic" properties," *IEEE Trans. on Software Engineering*, vol. 22, no. 1, pp. 53–66, 1996.

[8] M. R. Clarkson and F. B. Schneider, "Hyperproperties," in *Computer Security Foundations Symposium*. IEEE, 2008.

[9] D. Denning, "An intrusion detection model," *IEEE Trans. on Software Engineering*, vol. 13, no. 2, pp. 222–232, 1987.

[10] M. Shanahan, "The event calculus explained," in *Artificial Intelligence Today*, ser. LNCS. Springer, 1999, vol. 1600, pp. 409–430.

[11] E. T. Mueller, "Automating commonsense reasoning using the event calculus," *Communications of the ACM*, vol. 52, no. 1, pp. 113–117, 2009.

[12] J. Allen, J. Hendler, and A. Tate, Eds., *Readings in planning*. Morgan Kaufmann, 1990.

[13] E. T. Muller, "Event calculus reasoning through satisfiability," *Journal of Logic and Computation*, vol. 14, no. 5, pp. 703–730, 2004.

[14] E. M. Clarke, O. Grumberg, and D. Peled, *Model checking*. MIT Press, 1999.

[15] R. J. Anderson, "A security policy model for clinical information systems," in *Proc. of SP '96*. IEEE, 1996.

[16] D. D. Clark and D. R. Wilson, "A comparison of commercial and military computer security policies," in *Proc. IEEE Symp. Research in Security and Privacy*, 1987, pp. 184–194.

[17] M. J. Nash and K. R. Poland, "Some conundrums concerning separation of duty," in *Proc. IEEE Symp. Research in Security and Privacy*, 1990, pp. 201–207.

[18] J. A. Goguen and J. Mesenguer, "Security policies and security models," in *IEEE Symposium on Security and Privacy*, 1982, pp. 11–20.

[19] N. Amálio and G. Spanoudakis, "From monitoring templates to security monitoring and threat detection," in *Proc. of SECURWARE '08*. IEEE, 2008, pp. 185–192.

[20] N. Amálio, S. Stepney, and F. Polack, "A formal template language enabling meta-proof," in *FM 2006*, ser. LNCS, vol. 4085. Springer, 2006, pp. 252–267.

[21] N. Amálio, "Generative frameworks for rigorous model-driven development," Ph.D. dissertation, Dept. Computer Science, Univ. of York, 2007.

[22] ——, "Suspicion-driven formal analysis of security requirements," in *SECURWARE'2009*. IEEE, 2009, pp. 217–223.

[23] J. McLean, "A comment on the "basic security theorem" of bell and lapadula," *Information Processing Letters*, vol. 20, pp. 67–70, 1985.

[24] D. E. Bell and L. J. Padula, "Secure computer systems: a mathematical model," Mitre Corporation, Bedford, MA, Tech. Rep. MTR-2547 Vol. II, 1996.

[25] L. Lin, B. Nuseibeh, D. Ince, and M. Jackson, "Using abuse frames to bound the scope of security problems," in *Proc. RE '04*. IEEE, 2004, pp. 354–355.

[26] I. Alexander, "Misuse cases: Use cases with hostile intent," *IEEE Software*, vol. 20, no. 1, pp. 58–66, 2003.
[27] J. McDermott and C. Fox, "Using abuse case models for security requirements analysis," in *Annual computer security applications conference*. IEEE, 1999.
[28] T. Hollebeek and R. Waltzman, "The role of suspicion in model-based intrusion detection," in *Proc. of NSPW '04*. ACM, 2004, pp. 87–94.
[29] N. Amálio, S. Stepney, and F. Polack, "Formal proof from UML models," in *Proc. ICFEM 2004*, ser. LNCS, vol. 3308. Springer, 2004, pp. 418–433.

APPENDIX

A. Sample outputs of decreasoner

This appendix presents sample outputs generated by the *decreasoner* tool, while carrying out the model analysis presented in this paper.

1) Security Violation Goal (AG1): The output generated by decreasoner for the security violation goal (AG1) of section IV-B is:

```
no models found
```

This means that decreasoner could not find any solutions for the analysis goal.

2) Suspicion Goal 1 (AG2): The following output of decreasoner shows one sample solution for the security violation goal (AG1) of section IV-B is:

```
---
model 1:
0
CanAccessMD(Jones, Anderson).
IsDoctorOf(Jones, Anderson).
Happens(AuthoriseAccess(Jones, Anderson), 0).
1
+CredentialMD(Jones, Anderson, 0).
+HasValidCredential(Jones, Anderson).
Happens(SetDoctorOnLeave(Smith, Jones), 1).
2
+OnLeave(Jones).
Happens(SetSubstituteDoctor(Smith, Jones,
Smith), 2).
3
+CanAccessMD(Smith, Anderson).
+IsSubstituteDoctor(Smith, Jones).
Happens(AuthoriseAccess(Smith, Anderson), 3).
4
-HasValidCredential(Jones, Anderson).
+CredentialMD(Smith, Anderson, 3).
+HasValidCredential(Smith, Anderson).
Happens(AuthoriseAccess(Jones, Anderson), 4).
5
+CredentialMD(Jones, Anderson, 4).
+HasValidCredential(Jones, Anderson).
Happens(AuthoriseAccess(Jones, Anderson), 5).
6
+CredentialMD(Jones, Anderson, 5).
Happens(GetMD(Smith, Anderson), 6).
7
-HasValidCredential(Smith, Anderson).
+ExposedToAt(Smith, Anderson, 6).
+GoalSatisfied().
```

The sample above says that event $AuthoriseAccess\ (Jones,\ Anderson)$ happens at timepoint 0, and that $SetDoctorOnLeave\ (Smith,\ Jones)$ happens at timepoint 1. The analysis goal is satisfied by event $GetMD(Smith, Anderson)$ that happens at timepoint 6.

The remaining 4 sample solutions generated by decreasoner for analysis goal AG2 are as follows:

```
---
model 2:
0
CanAccessMD(Jones, Anderson).
IsDoctorOf(Jones, Anderson).
Happens(AuthoriseAccess(Jones, Anderson), 0).
1
+CredentialMD(Jones, Anderson, 0).
+HasValidCredential(Jones, Anderson).
Happens(SetDoctorOnLeave(Smith, Jones), 1).
2
+OnLeave(Jones).
Happens(SetSubstituteDoctor(Jones, Jones,
Smith), 2).
3
+CanAccessMD(Smith, Anderson).
+IsSubstituteDoctor(Smith, Jones).
Happens(AuthoriseAccess(Smith, Anderson), 3).
4
-HasValidCredential(Jones, Anderson).
+CredentialMD(Smith, Anderson, 3).
+HasValidCredential(Smith, Anderson).
Happens(AuthoriseAccess(Jones, Anderson), 4).
5
+CredentialMD(Jones, Anderson, 4).
+HasValidCredential(Jones, Anderson).
Happens(AuthoriseAccess(Jones, Anderson), 5).
6
+CredentialMD(Jones, Anderson, 5).
Happens(GetMD(Smith, Anderson), 6).
7
+ExposedToAt(Smith, Anderson, 6).
+GoalSatisfied().
---
model 3:
0
CanAccessMD(Jones, Anderson).
IsDoctorOf(Jones, Anderson).
Happens(AuthoriseAccess(Jones, Anderson), 0).
1
+CredentialMD(Jones, Anderson, 0).
+HasValidCredential(Jones, Anderson).
Happens(SetDoctorOnLeave(Jones, Jones), 1).
2
+OnLeave(Jones).
Happens(SetSubstituteDoctor(Jones, Jones,
Smith), 2).
3
+CanAccessMD(Smith, Anderson).
+IsSubstituteDoctor(Smith, Jones).
Happens(AuthoriseAccess(Smith, Anderson), 3).
4
-HasValidCredential(Jones, Anderson).
+CredentialMD(Smith, Anderson, 3).
+HasValidCredential(Smith, Anderson).
Happens(AuthoriseAccess(Jones, Anderson), 4).
5
+CredentialMD(Jones, Anderson, 4).
+HasValidCredential(Jones, Anderson).
Happens(AuthoriseAccess(Jones, Anderson),
5).
6
+CredentialMD(Jones, Anderson, 5).
Happens(GetMD(Smith, Anderson), 6).
7
```

```
−HasValidCredential(Smith, Anderson).
+ExposedToAt(Smith, Anderson, 6).
+GoalSatisfied().
___
model 4:
0
CanAccessMD(Jones, Anderson).
IsDoctorOf(Jones, Anderson).
Happens(AuthoriseAccess(Jones, Anderson), 0).
1
+CredentialMD(Jones, Anderson, 0).
+HasValidCredential(Jones, Anderson).
Happens(SetDoctorOnLeave(Jones, Jones), 1).
2
+OnLeave(Jones).
Happens(SetSubstituteDoctor(Smith, Jones,
Smith), 2).
3
+CanAccessMD(Smith, Anderson).
+IsSubstituteDoctor(Smith, Jones).
Happens(AuthoriseAccess(Smith, Anderson), 3).
4
−HasValidCredential(Jones, Anderson).
+CredentialMD(Smith, Anderson, 3).
+HasValidCredential(Smith, Anderson).
Happens(AuthoriseAccess(Jones, Anderson), 4).
5
+CredentialMD(Jones, Anderson, 4).
+HasValidCredential(Jones, Anderson).
Happens(AuthoriseAccess(Jones, Anderson), 5).
6
+CredentialMD(Jones, Anderson, 5).
Happens(GetMD(Smith, Anderson), 6).
7
−HasValidCredential(Smith, Anderson).
+ExposedToAt(Smith, Anderson, 6).
+GoalSatisfied().
___
model 5:
0
CanAccessMD(Jones, Anderson).
IsDoctorOf(Jones, Anderson).
Happens(AuthoriseAccess(Jones, Anderson), 0).
1
+CredentialMD(Jones, Anderson, 0).
+HasValidCredential(Jones, Anderson).
Happens(SetSubstituteDoctor(Jones, Jones,
Smith), 1).
2
+IsSubstituteDoctor(Smith, Jones).
Happens(SetDoctorOnLeave(Jones, Jones), 2).
3
+CanAccessMD(Smith, Anderson).
+OnLeave(Jones).
Happens(AuthoriseAccess(Smith, Anderson), 3).
4
−HasValidCredential(Jones, Anderson).
+CredentialMD(Smith, Anderson, 3).
+HasValidCredential(Smith, Anderson).
Happens(AuthoriseAccess(Jones, Anderson), 4).
5
+CredentialMD(Jones, Anderson, 4).
+HasValidCredential(Jones, Anderson).
Happens(AuthoriseAccess(Jones, Anderson), 5).
6
+CredentialMD(Jones, Anderson, 5).
Happens(GetMD(Smith, Anderson), 6).
7
−HasValidCredential(Smith, Anderson).
+ExposedToAt(Smith, Anderson, 6).
+GoalSatisfied().
```

Development of Measurable Security for a Distributed Messaging System

Reijo M. Savola
VTT Technical Research Centre of Finland
Oulu, Finland
E-mail: Reijo.Savola@vtt.fi

Habtamu Abie
Norwegian Computing Center
Oslo, Norway
E-mail: Habtamu.Abie@nr.no

Abstract—**Systematically developed security metrics make it possible to gather sufficient and credible security evidence for runtime adaptive security management and off-line security engineering and management. This study introduces and analyzes security metrics and parameter dependencies for one particular distributed messaging system. The focus is on the effectiveness and correctness of security-enforcing mechanisms. The security metrics development approach that the study utilizes is risk-driven, requirement-centric, and integrated with the development of Quality-of-Service metrics. In this approach, the security requirements are expressed in terms of lower-level measurable components by applying a decomposition approach. Security metrics are then developed based on the leaf components of the decomposition. The paper also analyzes the benefits and shortcomings of the metrics development approach and introduces a trust, confidence and trustworthiness calculation model for basic measurable components of the decomposition.**

Keywords-security metrics; security indicators; security strength; security requirements; messaging systems

I. INTRODUCTION

In order to obtain sufficient and credible evidence of the security performance of a system, service or product, a systematic approach to measuring security is required. Systematic definition of security metrics is a young field that still lacks widely accepted approaches, mainly because the current practice of security is still a highly diverse field.

This study's primary contribution is that it analyzes and defines an initial collection of security metrics and parameter dependencies for the security-enforcing mechanisms of one particular system that was used as an example, using the development method introduced in earlier work [1]. The paper also analyzes the benefits and shortcomings of the development method used in the study, and introduces a framework for calculating trust, confidence and trustworthiness of security metrics. This study advances the state of the art in security metrics in practical and concrete measurement methods, in measurable components, and in semi-formal models of security measurement and metrics. The scope of the study did not include formal modeling and validation of the defined metrics.

At a high level, the objectives measured by security metrics can be classified into three groups: security correctness, effectiveness and efficiency [2]. The discussions on metrics in this paper concentrate on the effectiveness and correctness of security-enforcing mechanisms, although it also discusses efficiency. The Security Metrics Objectives Segments (SMOS) model for the taxonomization of security metrics [2] classifies the main viewpoints of the metrics of the System under Investigation (SuI) into three categories: (*i*) security-enforcing mechanisms, (*ii*) the security quality of the system, and (*iii*) secure system lifecycle, project and business management. This study focuses on the first category: security-enforcing mechanisms. It should be noted that, from the point of view of the security metrics completeness for the target system, metrics are also required for the other two categories. The goal of this study was to provide extensive identification and high-level definition of metrics for security-enforcing mechanisms, while also being selective in the details thereof.

The study investigated security metrics, and how they were developed in an example system called GEMOM (Genetic Message Oriented Secure Middleware) [3]. GEMOM has been developed in the GEMOM EU FP7 ICT project, which focuses on security measurability, adaptive security, and the resilience of complex, distributed information systems. Security solutions with varying strength levels are required in resilient and distributed business-critical systems such as GEMOM so that they can manage security in an adaptive way according to the needs of varying situations. In adaptive security management, security metrics provide the means with which score different solutions during the system's operation. For instance, different authentication and authorization mechanisms can be utilized based on metrics. In addition, metrics are used off-line during Research and Development (R&D) and when the system configuration is changed.

This paper is organized as follows. Section II provides an introduction to security metrics. Section III presents the security metrics development process that was originally introduced in the earlier work of the authors of this paper, and analyzes its benefits and challenges. Section IV briefly introduces the GEMOM system and its monitoring approach; then Section V discusses GEMOM security threats and security requirements. Section VI identifies Basic Measurable Components (BMCs) for the effectiveness and correctness of security-enforcing mechanisms in GEMOM and proposes an initial collection of metrics and parameter dependencies. Section VII presents some observations from the study and discusses the feasibility and potential research directions of security metrics. Section VIII discusses related work and Section IX offers concluding remarks and poses some questions for future research.

II. Security Metrics, Security Indicators and Security Strength

It is often claimed that an activity cannot be managed well if it cannot be measured. System developers, managers, and security assurance personnel, as well as automated security monitoring approaches, require sufficient and credible evidence that the SuI implements the intended security level or performance. An improvement in the value of a measurement result makes it more likely that the related objective or sub-objective will be met. The term *metrics*, as used in the context of Information Technology (IT), is misleading because it implies that traditional concepts in metrology, as used in physics and other areas of science and technology, apply equally to IT [4]. There are unknown multi-disciplinary dependencies in IT, as well as doubts and often subjective judgments about technical maturity due to the novelty of applications and other technical solutions. Terms such as *security indicators* or *security strength* might be more appropriate in the case of security related objectives. The term *security strength* has traditionally been used among cryptographers and has only recently been used in reference to more general security concepts. This study uses the term *security metrics*, while recognizing the imperfections of the term.

A. Metrics and Measurements

Measurement is the process by which numbers or symbols are assigned to attributes of real world entities in such a way that describes them according to clearly defined rules [5]. In general, measurements provide single-point-in-time views of specific, discrete factors, while *metrics* are derived by comparing two or more measurements taken over time with a predetermined baseline [6].

B. Use of Metrics

Security metrics can be used for decision support, particularly in assessment, monitoring, and prediction. Security measurement targets can include a technical system, service, or product, or an organization, its processes, and resources [7]. Some of the ways in which security metrics can be used include [8]:

- Risk management activities for mitigating security risks,
- Comparison of security-enforcing mechanisms or solutions,
- Obtaining information about the security posture of an organization, process, or product,
- Security assurance (analysis, testing, monitoring) of a product, organization, or process,
- Security testing (functional, red team and penetration testing) of a system,
- Certification and evaluation of a product or organization,
- Adaptive security monitoring and management during system operation, and
- Intrusion detection and prevention in a system.

The intended use and target audience influences the security metrics requirements. Complex metrics structures with various metrics and sub-metrics can be used in automatic calculations and decision-making. However, if the goal is to develop security metrics for a human audience, such as a company's senior management in a company, it is important to visualize the result and the final metrics should be clearly understandable.

C. Dimensions to be Measured

Information security, as a target in itself, cannot be satisfactorily measured because it is such an abstract concept and has many multi-disciplinary dependencies. Therefore, security objectives should be investigated in greater detail. The security dimensions that the metrics should address depend largely on the application domain and environment.

The most commonly recognized dimensions of information security are Confidentiality, Integrity and Availability (CIA) [9], often referred to as the *CIA model*. Confidentiality objectives ensure that information is accessible only to those authorized to have access. Integrity encompasses safeguards of several aspects of accuracy and completeness of information. The Availability dimension can be defined by the objectives to ensure that authorized users have access to the information and associated assets they require within a reasonable timeframe [2]. The CIA model has some limitations, such as the fact that authenticity and non-repudiation of critical business transactions do not fit naturally in the model. Moreover, authorization depends on authentication. A more concise collection of security objectives includes various 'lower-level' dimensions like confidentiality, integrity, availability, authentication, authorization, and non-repudiation. This more accurately emphasizes the objectives of security-enforcing mechanisms [2]. The International Telecommunication Union (ITU) [10] has defined a larger set of security dimension: access control, authentication, non-repudiation, data confidentiality, communication security, data integrity, availability, and privacy. Several other factors affect the security of information systems, such as accountability, auditing, controllability, correctness, identification, recovery, reliability, robustness, safety, dependability, supervision, and trustworthiness, as well as functionality [11][12][13]. Security, Trust, Dependability, and Privacy (STDP) are often grouped together when defining security-relevant objectives for technical systems and services. It is important to note, however, that these terms are overlapping and, in some cases, even contradict each other [14]. Avižienis *et al.* [15] presented a detailed taxonomy of security and dependability quality attributes that can be used in the selection of adequate dimensions to be investigated [2].

III. Security Metrics Development Approach

The authors' earlier work [16] proposed an iterative process for security metrics development, which this paper enhances and clarifies. The process aims to develop a balanced and detailed security metrics collection for the SuI and the related measurement architecture. Measurement architecture is the operational structure for measurement and

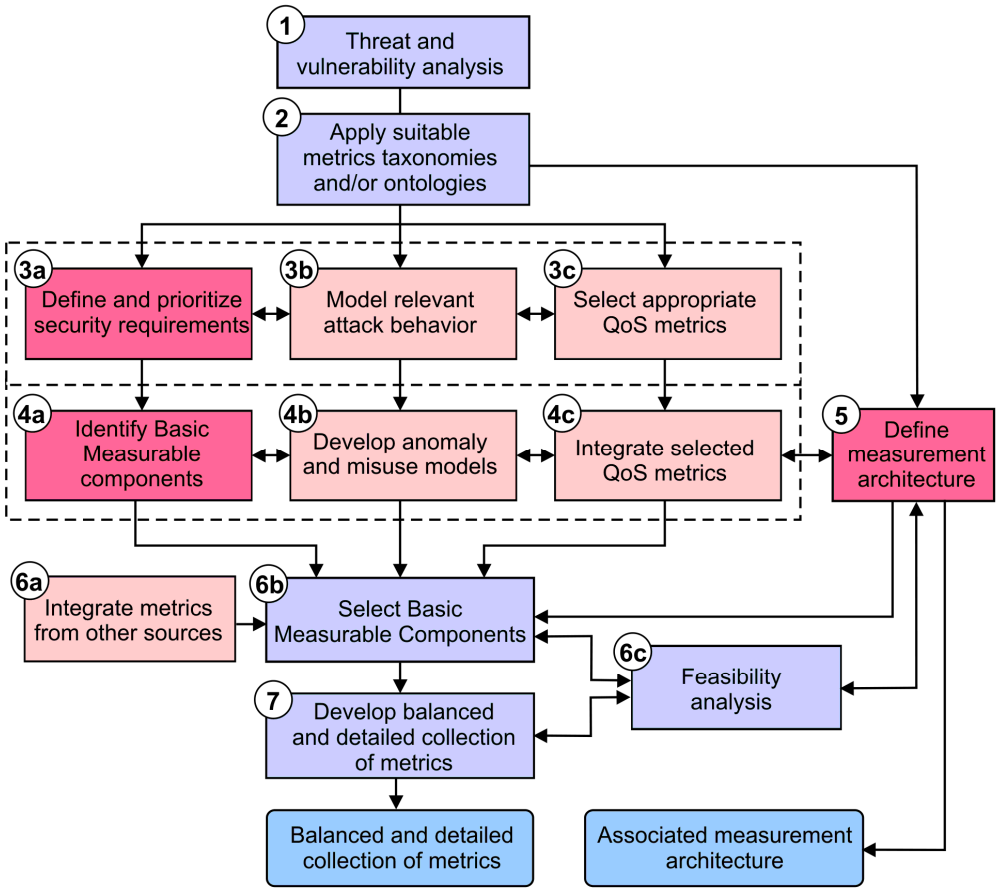


Figure 1. Security metrics development method for GEMOM. The left-most branch concentrates on security requirement decomposition.

evidence collection. The steps are as follows (points (a), (b), and (c) in Steps 3, 4 and 6 represent parallel activities):

1. Conduct a **threat and vulnerability analysis** of the SuI and its use environments, with appropriate impact and risk exposure analyses**.** This phase can be bypassed if there are valid pre-existing analysis results;
2. If applicable, utilize suitable security metrics **taxonomies and/or ontologies** (see, for example [2]) to further plan the measurement objectives and metrics types;
3. Develop **security requirements and start modeling**: (a) Define and prioritize the security requirements holistically, based on the results of Steps 1 and 2, giving the most attention to the most critical security requirements; (b) Model relevant attack strategies in prioritized order and carry out an attacker cost-benefit analysis; and (c) Select appropriate QoS metrics for the security-oriented availability metrics;
4. **Decomposition, modeling, and integration:** (a) Identify Basic Measurable Components (BMCs) from the requirements using a decomposition approach. BMCs are leaf components of the decomposition that clearly manifest a measurable property of the system; (b) Develop possible anomaly and/or misuse models; and (c) Integrate the selected QoS metrics into the collection of BMCs;
5. Define **measurement architecture** with sufficient intrinsic security-measurability (i.e., self-contained readiness for security measurement). Pay attention to readily available counters, measurement points, etc;
6. **Integrate metrics and select BMCs:** (a) Integrate metrics from other sources; and (b) Carry out BMC selection based on (c) Feasibility analysis. The feasibility analysis is an iterative stage that takes into account the measurement architecture, individual metrics, and the entire collection of metrics;
7. Develop an appropriate **balanced and detailed collection of metrics** with on-line/off-line division and the functionalities and processes in which they are used.

All steps in the process are highly iterative and the sequence of the steps can be varied if relevant information becomes available in a different sequence. Steps 1 to 3 should be started as early as possible in the system development and elaborated iteratively as the design becomes more mature. Steps 4 and 5 can be carried out in parallel. Step 5 can also already be partially started during the architectural design phase.

TABLE I. BENEFITS AND CHALLENGES OF THE PROPOSED STEPS

Step	Benefits	Challenges
1	Security-enforcing mechanisms, expressed by the requirements, can be tailored as accurately as possible to mitigate or cancel the actual threats.	Actual information about threats and vulnerabilities is sometimes difficult to obtain. It might also be too time-consuming to carry out a thorough analysis.
2	Taxonomical information systematizes metrics development and helps in planning the metrics types.	Validated security metrics taxonomies and ontologies based on experimental results are difficult to find.
3 a	Security requirements steer the R&D in the right direction and act as the baseline for evidence gathering by security metrics.	Definition of sufficient security requirements is demanding. "Negative requirements" (Section III.B) require a great deal of effort.
3 b	Knowledge about attack strategies is required for anomaly and misuse models, acting as the basis for attack-oriented metrics.	There are several ways to compromise a system, only some of which can be modeled with a reasonable amount of time and effort.
3 c	QoS metrics also reflect availability of the SuI from a security perspective.	The line between performance and security-oriented QoS metrics is fuzzy.
4 a	The decomposition expresses the relationship between the components and the requirements.	In some cases, it might be difficult to decompose the essential subcomponents or impossible to carry out the related measurements.
4 b	Anomaly and misuse models can be used in QoS and security monitoring.	The development of feasible models is time-consuming and it can be difficult to obtain proper training data.
4 c	Integration of QoS and other availability metrics increases the amount of availability evidence from a security perspective.	It is difficult to choose feasible QoS metrics from a security perspective. The type of QoS might be different from the security metrics.
5	A practical measurement architecture with proper evidence collection is necessary. Intrinsic security-measurability enables smart evidence collection.	Performance constraints and other conflicting functionality goals might complicate the design and measurement architecture and measurement support in components.
6 a	Metrics from sources other than the actual metrics development process increase the completeness of the metrics collection.	The relationship between the metrics from other sources and the security requirements are not directly visible as they are in the decomposition process.
6 b	The systematic selection of individual metrics is needed in order to increase the feasibility of the final collection of metrics.	The selection process is challenging. The need for individual metrics and the entire metrics collection must be taken into acount.
6 c	Feasibility analysis of the chosen metrics is needed in order to select the final practical collection of metrics	Feasibility analysis could require substantial information from realistic situations in which the SuI is used.
7	Eventually, detailed metrics will be needed in order to use the metrics system.	The detailed development of metrics involves several challenges, such as a lack of data from realistic situations, scaling, assessment of confidence values, and fine-tuning of decision support.

This approach is based on earlier work by the same authors: the early approach presented in [17] was enhanced in [18] with decomposition and in [16] with QoS metrics and anomaly monitoring branches. The present approach adds optionality to the threat and vulnerability analysis, changes the order of taxonomical and ontological work, emphasizes the importance of intrinsic security-measurability support, simplifies the BMC selection step (Step 6), and adds feasibility analysis as a separate stage with connections to the measurement architecture, individual metrics, and the metrics collection. The present paper also analyzes the benefits and challenges of the proposed steps (see Table I). The process is visualized in Figure 1, with the pink boxes depicting the steps for optional QoS and other metrics development.

A. Threat and Vulnerability Analysis

The first step in a risk-driven methodology is threat analysis, with the goal of identifying security threats and their sources, and analyzing their likelihood. It is also the starting point of security metrics development, unless sufficient threat information exists beforehand. There are various ways to carry out a threat analysis, from simply listing threats to modeling them in a more rigorous way.

The extent of threat analysis depends, for example, on the criticality of the planned applications of the SuI. The Microsoft threat risk modeling process [19] suggests the following steps:

1. Identify security objectives,
2. Survey the SuI architecture,
3. Decompose the SuI architecture to identify functions and entities that impact security,
4. Identify threats, and
5. Identify vulnerabilities.

Vulnerability analysis can be carried out once appropriate technological choices have been made. Technology and implementation-dependent vulnerabilities cause different kinds of threats to the system. Well-known vulnerability listings and repositories, such as Open Web Application Security Project (OWASP) Top 10 [20], can be used in vulnerability analysis. OCTAVE tools and methods [21] offer support for threat and vulnerability analyses.

B. Security Requirements

Security requirements – high-level statements of countermeasures that adequately mitigate the identified risk [22] – form the reference basis for security metrics development. They can derive from *threats*, *general organizational policies*, and *environment properties*. Security requirements derived from threats represent countermeasures, or security-enforcing mechanisms. Note the distinction between general organizational policies and *security policies*. A security policy is concerned with the design, analysis, implementation, deployment, and use of efficient and secure technology that handles the SuI in accordance with the relevant set of security rules and procedures, and is based on security requirements [22]. Environment properties contribute to the security of the SuI from the outside [23]. A security requirement of the SuI r_i is

derived from applicable threat(s) ξ_i, general organizational policies p_i and the environment properties e_i [23]:

$$r_i = (\xi_i , p_i, e_i), \quad r_i \in R, \xi_i \in \Xi, p_i \in P, e_i \in E, \tag{1}$$

where i is index number, R is the collection of all security requirements of SuI, Ξ is the collection of all security threats to be canceled or mitigated, P is the collection of all general organizational policies applied to SuI, and E is the collection of all environment properties that contribute to the security of the SuI from the outside. The effectiveness of security policies, derived from security requirements, is crucial for achieving adequate security performance and the security objectives should be inline with the security requirements. According to Firesmith [24], the most current software requirement specifications (*i*) are totally silent regarding security, (*ii*) only specify vague security goals, or (*iii*) only specify commonly used security mechanisms, such as encryption and firewalls, as architectural constraints.

Non-security requirements can have a significant effect on the quality of the system's security. Business constraints can affect the impact of security risks and the SuI's exposure to these risks. The usability and performance of security-enforcing mechanisms are also important objectives of system design. Ideally, the characteristics of excellent software requirements, including security requirements, include completeness, correctness, feasibility, necessity, prioritization, unambiguity, and verifiability [25]. The main difference between security requirements and software requirements is that most non-security requirements stipulate that the SuI must take specific necessary or desired action, while security requirements often concentrate on avoiding the occurrence of something that is undesired (*negative behavior* requirement). The lack of an understanding of and attention to negative requirements is at the root of many security problems [17].

C. Security Requirement Decomposition

A substantial mechanism in this paper's requirement-centric security metrics development approach is *requirement decomposition*. The following decomposition process, based on the work by Wang and Wulf [26], is used to identify measurable components from the security requirements:

1. Identify successive components from each security requirement (goal) that *contribute to the correctness, effectiveness*, *and/or efficiency* of the goal. Correctness or effectiveness goals are emphasized depending on the needs for metrics in either dimension;
2. Examine the subordinate nodes to determine whether further decomposition is needed. If it is, repeat the process with the subordinate nodes as current goals, breaking them down to their essential components; and
3. Terminate the decomposition process when none of the leaf nodes can be decomposed any further, or when further analysis of these components is no longer necessary.

When the decomposition terminates, all leaf nodes should be measurable components.

D. Measurement Architecture

In practice, it is necessary to identify measurable information and the mechanisms of how to obtain and process that data. Both on-line measurement architecture and off-line evidence collection should be designed. On-line and off-line measurements often depend on one another. In the example GEMOM system, the Monitoring Tool is the central module of the measurement architecture, with connections to the GEMOM Broker, publish/subscribe clients, Authentication and Authorization functionality, and Adaptive Security Management at the overlay level. Furthermore, the Monitoring Tool has an interface for tracking resources that are outside the GEMOM node, such as storage, memory, I/O devices, and network interfaces.

E. More Detailed Metrics Development

The potential BMCs should be selected on the basis of the feasibility, complexity and availability of information needed for the metric. The detailed development of the chosen security metrics should include definition of the following issues [1]:

- **Purpose** of the metric,
- **Target** description of the metric, for example, using composition-decomposition,
- **Formalization** of the metric into a computational or understandable form,
- **Value scale or ordering**,
- Close-to-optimal or *appropriate* **value range** depicting the 'desired level of security' and
- **Thresholds** (if needed).

Metrics can be used for many different purposes, which means that the above suggestions are not valid for all situations. Security metrics can be classified in many different ways and one metric can incorporate several metric characteristics.

IT system security comprises two independent aspects: security correctness and security effectiveness. In practical research and development, security efficiency objectives are also important. Security correctness denotes an assurance that security has been correctly implemented [2]. Security effectiveness, on the other hand, denotes an assurance that the security solutions meet the stated objectives: that is, they satisfy expectations for resilience in the use environment while not doing anything else other than what they are intended to do [2][4]. Security efficiency is concerned with the productivity dimension: the resources, time and money spent on security work and solutions [2].

The final choice of metrics depends on their use and feasibility: the metrics should add value to the decision-making process. Metrics can also assist in making the best decision based on incomplete knowledge. In addition, good security metrics are aligned with business objectives and

should allow comparisons to both internal and external benchmarks.

F. Integration of Metrics

Different weights can be associated with different component metrics in order to indicate the relative importance among the component metrics. A 'close to correct' weight assignment is used in practice, because there are no analytical results for determining the relative priorities of the elements, other than the careful use of one's expertise and judgment [27]. Two important dimensions strongly affect the weight: the potential *impact* of the threat and the SuI's *exposure* to that threat Impact analysis can be iterated to react to changes in the threat environment. Accordingly, the actual threat exposure estimate, the *exposure weight*, can vary dynamically depending on the system and the use of the application. The impact and exposure weights can be integrated in the component metrics weight factors using suitable heuristics that can interpret their interaction. Figure 2 shows the effect of impact and exposure. The 'high impact, high exposure' region obviously depicts the most critical zone, resulting in higher weight coefficients. Threats that fall into the 'low impact, high exposure' or 'high impact, low exposure' categories also result in increased weighting compared to the 'low impact, low exposure' zone [1].

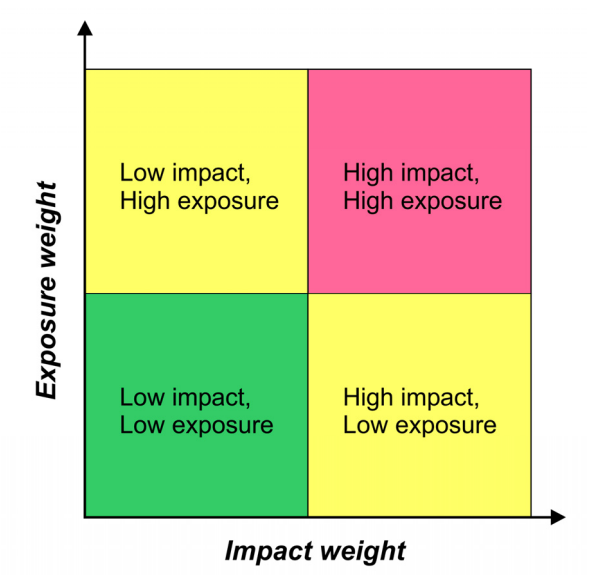


Figure 2. Threat exposure and impact dimensions [1].

The dynamic nature of threats, their impact, and the system's exposure to them can be reflected in the collection of security metrics by developing a method that relates these parameters to the actual weights used in a combination of different security metrics. In this case, the weighting acts as the 'interface' to threat dynamics from a more stable collection of security metrics.

The overall collection of chosen metrics can be managed, for example, in the form of a *balanced scorecard*, in which a score is assigned for all metrics components [1]. Different component scores are aggregated into an overall score using a suitable function. The following considerations are important when developing a balanced scorecard for security metrics [28]:

- Scorecards are a raw approximation of security risks;
- Attention should be paid to the selection of the correct component metrics;
- Scales must be normalized;
- Special care is needed if the scale types (nominal, ordinal, interval, ratio) are mixed;
- The mathematical functions used for the metrics should be carefully designed (average, sum, minimum, maximum, logical functions, and inference rules);
- Explicit rules must be defined for interpreting the aggregate score;
- The interdependencies between threats, threat agents, vulnerabilities, assets, etc., should be identified;
- The method for dealing with uncertainty, vagueness, missing information, imprecision, and contradictory information should be incorporated;
- The scorecard should support an increased security level and awareness of it;
- Essential data should be kept visible and open to peer review; and
- Standard terminology and definitions should be used.

IV. System under Investigation: GEMOM

Message Oriented Middleware (MOM) increases the interoperability, portability, and flexibility of architectures by enabling applications to exchange messages with other programs without having to know the platform on which the other application resides within the network [29][30][31]. GEMOM (Genetic Message Oriented Secure Middleware) [3] is an MOM based on the publish/subscribe messaging paradigm, the most efficient method of integrating medium to high complexity distributed systems. The GEMOM project use scenarios include a collaborative business portal, a financial market data delivery system, a road management system and a money transfer banking system [16].

A. Characteristics of the GEMOM System

The term *resilience* refers to a system's ability to return to its normal operational state after encountering an attack or other problem and continue its planned tasks. The GEMOM system is a resilient and scalable MOM that supports adaptive security management with the help of a monitoring functionality that is based on security and Quality of Service (QoS) metrics. The Adaptive Security Management system in GEMOM is able to learn and adapt to the changing threat environment without significantly sacrificing the efficiency, flexibility, reliability, and security of the system. This involves gathering relevant information both from within the system and from the environment, analyzing the collected information, and responding to changes by adjusting security functions such as encryption schemes, security protocols, security algorithms, and different authentication and

authorization mechanisms. Information gathering is carried out by security and QoS monitoring services and related administration services [1]. The publish/subscribe paradigm in GEMOM is based on publishing to *topics* and subscribing to them. Topics belong to *namespaces*, a higher-level concept in a hierarchy.

B. GEMOM System Architecture

The GEMOM system architecture is composed of a set of communicating entities known as GEMOM Nodes or G-Nodes. Some of these G-Nodes are operational (micro nodes, depicted in blue in Figure 3 [16]) and some are managerial (macro nodes, shown in pink in the figure). The operational G-Nodes, including Message Brokers, Clients (for publishing and subscribing messages), and Authentication and Authorization Modules, interact with relevant managerial nodes according to the situation. The managerial G-Nodes include Adaptive Security Managers, Audit and Logging Modules, Anomaly Monitors, and Monitoring Tools with associated Security Measurement Managers and QoS Managers. These G-Nodes make runtime operation decisions and require a wider perspective of the system than the individual operational G-Nodes. A Message Broker is a core GEMOM functionality package that consists of an application server, numerous plug-and-play objects, configuration files, and database schemas [16]. Several components have been built in GEMOM in such a way that they exhibit properties that support security measurements. In other words, they can be considered *intrinsically security-measurable* components [4], which are entities that are inherently attuned to security measurement.

C. Use of Security Metrics in GEMOM

Security evidence in the form of metrics is central to GEMOM. The resulting metrics are used for different purposes [16], including:

- Security and QoS monitoring (on-line activity),
- Adaptive Security Management (a combination of on-line and off-line activities), and
- Security engineering, management and software security assurance of the system, and service (off-line activities).

D. GEMOM Monitoring Concept

The GEMOM Monitoring Tool is responsible for collecting security and QoS evidence, and for maintaining an appropriate metrics database in GEMOM. There is one Monitoring Tool for each Message Broker. The Monitoring Tool consists of the Monitor Core (MC) software process functionality and the Monitor Modules. The MC runs in the background and the Monitor Modules can be preconfigured or added runtime. The MC offers database and messaging services to the Monitor Modules: it connects to a GEMOM Broker and the modules use it to publish and subscribe to

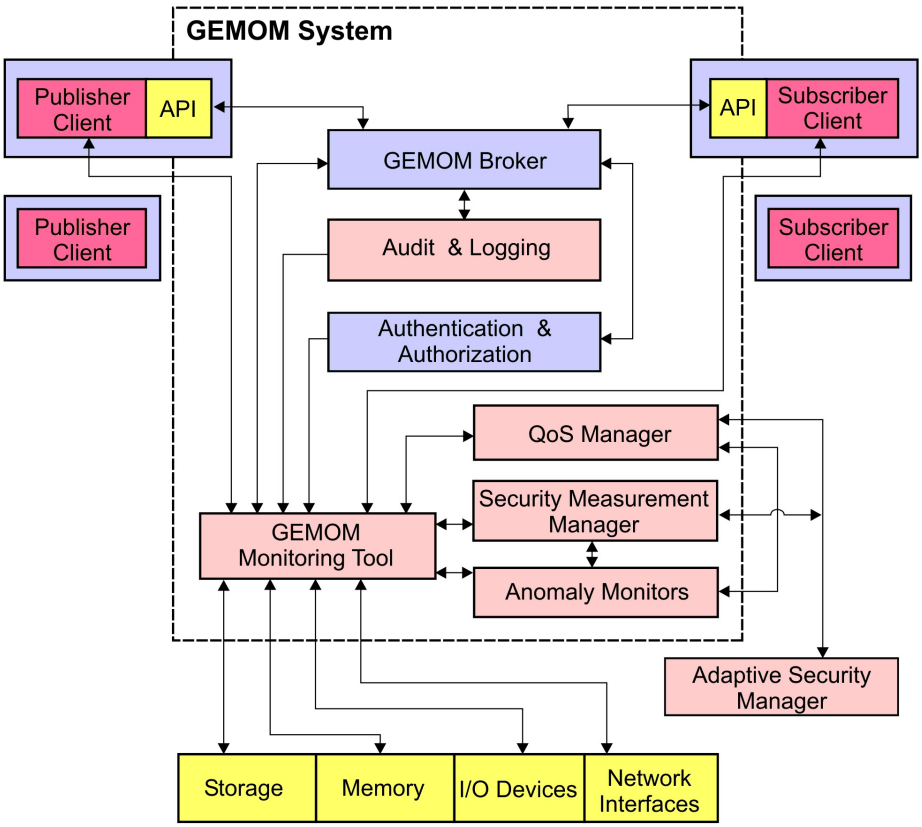


Figure 3. Example of information flows to and from the GEMOM system [16].

relevant topics in a measurement namespace [16]. A Monitoring Tool is connected to a Message Broker, an Audit and Logging Module, an Authentication and Authorization Module, a QoS Manager, an Anomaly Detector Module, a Security Measurement Manager and relevant memory (used and free), storage (hard disks, memory sticks), and network interfaces and Input/Output devices, such as a keyboard. In addition to logs, the monitoring system is able to monitor messages and metadata. Figure 3 depicts the information exchange connections of the GEMOM modules. GEMOM supports multi-point monitoring in the following way: Monitor modules can be added that are able to communicate with other Monitor modules that are monitoring other brokers, clients or modules, via publish/subscribe topics runtime. Consequently, all distributed Monitoring Tools have up-to-date information at their disposal.

E. Adaptive Security Manager

At the managerial G-Nodes level, the Monitoring Tools co-operate with the Adaptive Security Manager (ASM). The ASM monitors and analyzes security details based on metrics and other evidence, plans adjustments, and executes the planned adjustments through a global control loop, using both manual and automated information. In this way, the ASM manages the behavior of the overall system with the help of Monitor modules.

V. Security Threats and Security Requirements of GEMOM

A. Threats and Vulnerabilities of GEMOM

The main security threats to GEMOM are the Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks. In addition to activities, a malicious authorized node can execute a message flooding attack by sending a stream of false publish or subscription messages on behalf of unauthorized nodes. Availability is the most fundamental security dimension, particularly in telecommunication systems, as shown in an example study in [32], due to the effects of DoS attacks. Availability threats have a high impact because the system, or a part of it, is most vulnerable during this type of attack. By exploiting the high vulnerability time-window, attackers can not only achieve their own specific goals, but potentially causing threats to other security dimensions [32]. If the resilience and self-protection mechanisms of GEMOM fail, an intruder could even seize the system using this strategy. Spoofing attacks can be made by sending false registration or de-registration requests concerning an authorized node. An unauthorized agent could also send registration or de-registration requests. A malicious node could also replay entire message(s) that had previously been sent by an authorized agent. General security threats of distributed messaging systems also include unauthorized nodes accessing messages, functionalities or services, masquerading attacks, eavesdropping and modifying, deleting, or tampering messages. The corruption of topics, namespaces, messages, requests, metadata, and functionalities processing data can jeopardize a system's integrity and confidentiality. An eavesdropper may be able to obtain information that was not meant to be divulged.

One potential source of authentication and authorization threats is the fact that client applications pertaining to different organizations are part of the GEMOM system. These organizations could use different authentication and identification technologies and standards. Furthermore, user credential management will not usually be under a single organization control [1].

B. Security Requirements of GEMOM

The results of the GEMOM threat and vulnerability analysis were mapped into security requirements that concentrate on *security-enforcing mechanisms*. The main security requirements set the basis for security metrics development in GEMOM. Note that the collection of security requirements is simplified and the following lists only the main requirements [1], each of which was allocated a prioritization description of *high*, *medium*, or *low*. Due to space limitations, only high and some medium requirements are discussed. The security requirements were originally developed during the architectural design phase of the GEMOM system [1]. They were later prioritized and iterated to remove gratuitous overlap and to increase coherence. The diagram in Figure 4 shows how the GEMOM security requirements can be classified. Adaptive security requirements are high-level and common to all other categories. In the following, the term *node* refers to a GEMOM node.

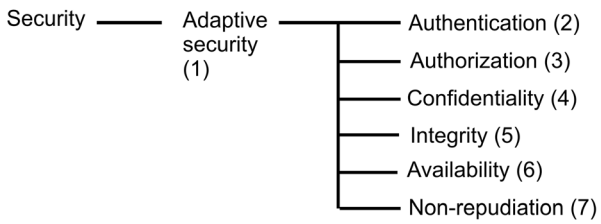


Figure 4. Categories of GEMOM security requirements [1].

TABLE II. Adaptive Security Requirements [1]

Requirement	Description
Adaptive authentication (Req. 1.1)	The system should be able to choose among different authentication methods based on an adaptive trust metric.
Adaptive authorization (Req. 1.2)	The system should be able to choose among different authorization methods based on an adaptive trust metric and authentication strength.
Adaptive conf-identiality (Req. 1.3)	The encryption strength should be able to be adapted according to an adaptive trust metric.
Self-protection (Req. 1.4)	The system should be able to predict, prevent, detect, and identify attacks, and to protect itself against them.

The main *adaptive security requirements* for GEMOM, which impact all other requirements, are listed in Table II

[1]. The core functionality of the adaptive security is to be able to carry out self-protection and resilience activities. In addition, authentication, authorization, and confidentiality mechanisms are varied in GEMOM. A self-protecting system can anticipate, detect, identify, and protect itself by taking corrective actions against threats.

TABLE III. AUTHENTICATION REQUIREMENTS [1]

Requirement	Description
Authentication mechanism (Req. 2.1)	The system should be able to choose from different authentication methods.
User authentication (Req. 2.2)	Any user accessing any node should be authenticated and the node should be aware of the authentication level.
Node to node authentication (Req. 2.3)	Any node accessing any other node should be authenticated and the node should be aware of the authentication level.
Message or metadata authentication (Req. 2.4)	The node should verify the authenticity of a message or metadata.
Identity federation (Req. 2.5)	GEMOM must be able to operate in an identity federated environment.
Identity management (Req. 2.6)	User identities should be securely managed.

Authentication refers to actions in which a user's credentials are used to verify the user's identity. Table III lists the requirements for the authentication of users and other system entities. The basic generic requirements for authentication are not explicitly discussed here.

TABLE IV. AUTHORIZATION REQUIREMENTS [1]

Requirement	Description
Authorization policy (Req. 3.1)	The policy identifies specific users, user groups, and types of users, specifies the operations permitted and authorization levels on authorization objects, and specifies the delegation privilege and depth.
Access control mechanism (Req. 3.2)	Access control can use different identity-based and role-based mechanisms depending on authentication strength.
User authorization (Req. 3.3)	The Authorization Manager should verify user identity and grant access to the resource allowed.
Revoking authorization (Req. 3.4)	The authorization functionality should support the revocation of authorization, for example, to users identified as harmful.
Authorization objects (Req. 3.5)	Authorization objects are namespaces, topics, metadata, and messages. Authorization rights can be granted for single authorization objects.
Delegation of privileges (Req. 3.6)	It should be possible to delegate security credentials to an entity so that it can act on the delegators behalf. The chain of delegation depth is three.

Authorization refers to the parties that are authorized to access specific resources of the system. Table IV lists GEMOM's main authorization requirements. Authorization functionality is responsible for granting rights, including access control based on access rights. The requirements start with a definition of the authorization policy. The authorization objects must be identified.

TABLE V. CONFIDENTIALITY REQUIREMENTS [1]

Requirement	Description
Message, log and metadata conf-identiality (Req. 4.1)	Messages, logs and metadata should only be delivered to authorized receivers.
Traceability info confidentiality (Req. 4.2)	Traceability information can only be accessed by authorized users.
Confidentiality classification (Req. 4.3)	The system should be able to assign confidentiality levels to both users and message contents.
Cryptography strength (Req. 4.4)	Cryptographic algorithms should be assigned a strength value.
Storage confidentiality (Req. 4.5)	Messages stored in non-volatile media at the Message Broker should be protected.

Confidentiality countermeasures ensure that information is protected from unauthorized disclosure [1]. Table V presents GEMOM's main confidentiality requirements. Different confidentiality levels are required depending on the sensitivity requirements of information and the level of trust in specific users. The scalability of confidentiality is important; confidentiality should be ensured despite the addition or removal of system users.

TABLE VI. INTEGRITY REQUIREMENTS [1]

Requirement	Description
Message, log and metadata integrity(Req. 5.1)	The system should ensure message, log, and metadata integrity.
Environment integrity (Req. 5.2)	The integrity of node system software, add-on components, and underlying operating systems shall be verified.
Persistent data integrity (Req. 5.3)	The integrity of data passing through, stored in, or persistent to the node shall be protected.

Integrity means that data or the system processing it is not altered or destroyed in an unauthorized manner [1]. Integrity requirements are presented in Table VI. As will be seen later, integrity is a horizontal requirement area, which means that it is part of other security dimensions.

TABLE VII. AVAILABILITY REQUIREMENTS [1]

Requirement	Description
Robustness to faults (Req. 6.1)	GEMOM functionality should be available to authorized users even in the case of several node faults.
Self-healing (Req. 6.2)	The system should be able to automatically create new redundancy in case of node faults.
Sudden reconfiguration (Req. 6.3)	The system should allow for the sudden reconfiguration of the available resources.
Metadata availability (Req. 6.4)	It should be possible to distribute the metadata of Brokers to an authorized user (in order to spawn new redundancy).

Availability is the property of being accessible and useable upon demand by an authorized entity [1]. As the resilience of GEMOM is a critical need, loss of availability can represent a major security complication. Table VII lists GEMOM's availability requirements. Based on the GEMOM threat analysis, DoS types of attacks are considered a very significant threat to the availability of the GEMOM system.

TABLE VIII. NON-REPUDIATION REQUIREMENTS [1]

Requirement	Description
Non-repudiation of origin or reception (Req. 7.1)	The system should protect against an originator's false denial of having published a message or a recipient's false denial of having received a message.
Non-repudiation of published and received messages (Req. 7.2)	The Broker should be able to prove retrospectively that a specific message was published by a specific publisher or that a specific message was received by a specific subscriber at a specific time.
Non-repudiation of triggered push (Req. 7.3)	The Publisher agent should, in retrospect, be able to prove the identity of a user that triggered a message.

Non-repudiation is the property of preventing users from later denying that they performed an action (sending or receiving a message, and publishing or subscribing) [1]. Table VIII includes non-repudiation requirements.

VI. EFFECTIVENESS AND CORRECTNESS METRICS OF SECURITY-ENFORCING MECHANISMS IN GEMOM

This section decomposes the requirements [1] presented above and introduces an initial collection of security metrics and parameter dependencies based on the BMCs identified in the decomposition. Adaptive security is dealt with last because it contains requirements that depend on lower-level constructs. Most of the introduced metrics require data collection before the results can be used in automated adaptive security management or off-line security management activities.

Reliability and integrity can be considered *horizontal metrics perspectives*, because they are both part of other security requirement decompositions at the leaf level. On the other hand, the security requirement categories identified in Figure 6 can be considered *vertical metrics perspectives*, which are decomposed below. Note that horizontal and vertical perspectives in this context are only abstractions, and decompositions are presented only from the vertical perspective. However, the horizontal metrics could be also expressed as decompositions, in which the child entities of the root (reliability or integrity) are composed of all decompositions mentioned in this section.

Integrity is addressed from both horizontal and vertical perspectives. The vertical integrity metrics below address integrity-enforcing algorithms that focus on data integrity, whereas horizontal integrity metrics address the integrity objectives for different system components as a precondition to data integrity. Note that it would also be possible to arrange all integrity metrics into a decomposition representation.

A. Reliability Metrics – A Horizontal Metrics Perspective

In the context of a security-enforcing mechanism reliability typically refers to software reliability, but depending on the type of mechanism, can potentially be the composite of hardware and software reliability. Software reliability can be seen as a part of software quality in general. According to the widely acknowledged reliability model, time-dependent reliability $R(t)$ has the following exponential function [33]:

$$R(t) = e^{-\lambda \cdot t}, \tag{2}$$

where λ is the failure rate, t is time, and e is Napier's constant (2.71828…). This equation is valid when the failure rate is constant over time and implies a Poisson probability distribution of failures. In addition, the Mean Time Between Failures (MTBF) is calculated as follows: MTBF = $1/\lambda$. Reliability is often quantified by MTBF for repairable systems. On the other hand, Mean Time To Failure (MTTF) is used for non-repairable systems. According to Bernstein [33], the general equation can be extended as:

$$R(t) = e^{-k \cdot C \cdot t / E \cdot \varepsilon}, \tag{3}$$

where k is a scaling constant, C is software complexity, t is the continuous execution time of the software, E is the development effort, and ε denotes the investment in software engineering tools, processes and code expansion that makes the development work more effective. The goal of software testing and other assurance activities is to make the reliability as close as possible to $R(0) = 1$, which represents perfect reliability. Note that the deterioration of reliability is normally an unintentional process. More detailed analyses on software reliability are provided in [33] and [34].

The following observations affect the development of reliability metrics in GEMOM:

- Reliability can be increased by adding redundancy and diversity to the functionality in question. Both constructs support high resiliency of the system; and
- Maintainability is a key consideration in reliability. *Adaptive maintainability* functionality increases reliability and is relevant in GEMOM, which uses adaptive security management.

B. Integrity Metrics – A Horizontal Metrics Perspective

The component integrity of a system is a precondition to data integrity, which, in turn, indicates that data has not been modified or destroyed in an unauthorized manner. The system and its components must generate, process, maintain or transmit the data so that data integrity is preserved. *Integrity errors* are central to integrity metrics and typically include the unauthorized alteration, deletion, addition, publication and subscription of data. In general, integrity I can be measured during a data gathering time period as:

$$I = \frac{n(AC)}{n(AC) + m \cdot n(IE)}, \quad (4)$$

where $n(AC)$ is the number of data processing actions by the system component in question, $n(IE)$ is the number of integrity errors, and m is a weighting factor.

Integrity errors can be reported by built-in self-tests implemented in the system, run at specified time instants or intervals. The self-tests can investigate, among other things, the correct operation of the module, interface, memory and/or system function in question, such as the control area of a security-enforcing mechanism, consistency of the data between the original and processed data, correct sequencing of the data, and the data value range. Furthermore, integrity errors can be reported off-line by users and operational personnel. Note that system and data integrity can be disrupted accidentally or intentionally. As a result, integrity is closely linked to reliability [34].

C. Authentication

The general authentication decomposition model depicted in Figure 5 [26] is used during the process of identifying potential metrics for GEMOM authentication strength.

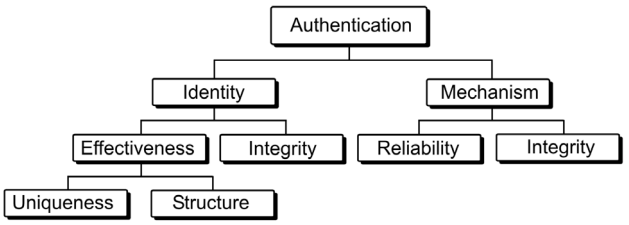


Figure 5. General authentication decomposition [26] used in GEMOM.

TABLE IX. BMCs OF AUTHENTICATION

Symbol	Basic Measurable Component
AIU	Authentication Identity Uniqueness
AIS	Authentication Identity Structure
AII	Authentication Identity Integrity
AMR	Authentication Mechanism Reliability
AMI	Authenticaiton Mechanism Integrity

Req. 2.1 states that the GEMOM system should be able to choose between different authentication mechanisms, such as a smart card, user name/password pair, and digital certificate, which represent different authentication mechanism security levels. Authentication mechanism requirements are varied by the GEMOM Adaptive Security Management functionality. If multi-modal authentication is used, that is, mechanisms combined from different authentication categories, the strength values are often higher. The model in Figure 7 suggests that the identity solution and the authentication mechanism make a significant contribution to the correctness and effectiveness of authentication. The identity tree of the decomposition emphasizes that the user identity federation (Req. 2.5) requirement is met. Identity management (Req. 2.6) is part of the authentication mechanism. There are differences in the case of user authentication (Req. 2.2), node-to-node authentication (Req 2.3), and message/metadata (Req. 2.4) authentication, and authentication metrics should be developed separately in each of these cases. The identity of the user is harder to validate than the identity of a node. See Table IX for the list of identified BMCs.

AIU, *AIS* and *AII* are mainly dependent on the identity solution, whereas *AMR* and *AMI* are mainly dependent on the authentication mechanism. Authentication Identity Uniqueness (*AIU*) is a function of the number of all identity information values divided by the values for which a *uniqueness condition* does not hold:

$$AIU = \frac{n(ID)}{n(ID) + w_{NSID} \cdot n(NSID)}, \quad (5)$$
$$NSID = \{x \in ID \wedge x \notin SID\},$$
$$SID = \{y : \exists y(ID(y) \wedge \forall z(ID(z) \rightarrow y = z))\},$$

where w_{NSID} is a weighting factor, $n(ID)$ is the total number of identity information values in use, ID is the collection of all identity information values, $ID(x)$ denotes the correspondence of identity information value x between a related actual real-world identity, $n(NSID)$ is the total number of non-unique identity information values, and $NSID$ is the collection of identity information values for which the uniqueness condition $\exists y(ID(y) \wedge \forall z(ID(z) \rightarrow y = z)$ does not hold. In other words, the maximum value of uniqueness is the case in which for every real-world identity used, there is one and only one identity information value that corresponds to a real-world identity. In the calculations, the unambiguity or ambiguity of identity information should be observed from an adversary's point of view.

Authentication Identity Structure (*AIS*) represents the security quality of the identity solution structure. The identity solution can be physical (e.g., a smart card), digital (e.g., user name/password pair), or a combination thereof. Identity structure that results from identity federation should also be taken into account. In general, the structure of a physical identity solution is stronger than one that is purely digital, provided that physical security measures have been well handled. In GEMOM, this metric has 'high, ' 'medium,' or 'low' ordinal values for different identity solutions. Attack modeling and long-term comparative data collection are needed in order to investigate the details of *AIS* quantification further.

Authentication Identity Integrity (*AII*) is dependent on the *AIS* and authentication integrity errors due to the identity solution IE_{ID}, and has ordinal values for different identity solutions. Therefore, *AII* can be denoted as a function of *AIS* and IE_{ID}:

$$AII = f(AIS, IE_{ID}). \quad (6)$$

Failures of the authentication mechanism include the approval of unauthorized access to the part of the SuI controlled by authentication mechanism, and denial of access for authorized users. Authentication Mechanism Reliability (*AMR*) is the ability of the authentication mechanism to correctly and effectively perform its required functions under potentially hostile conditions in the operational environment. *AMR* metrics design can be based on the general software reliability metrics, via parameter *E* in Eq. 3. In particular, the following issues affect *AMR*:

$$AMR = f(T_{ANRreq}, T_{AMRtest}, R_{AMRasm}), \quad (7)$$

where T_{AMRreg} is the time spent on authentication requirements engineering, $T_{ACMRtest}$ is the time spent testing the authentication mechanism, and $R_{ACMRasm}$ is the reliability of adaptive authentication functionality and authentication maintenance activities carried out by the GEMOM Adaptive Security Management.

Authentication Mechanism Integrity (*AMI*) is a precondition for data integrity in the control area of the authentication mechanism. It means that the authentication mechanism must function correctly in order to enable data integrity. Integrity errors include the unauthorized alteration, deletion, addition, publishing, and subscribing of data. Following the example of Eq. 4, *AMI* can be measured as follows during the data gathering period:

$$AMI = \frac{n(AU)}{n(AU) + w_{IEam} \cdot n(IE_{AM})}, \quad (8)$$

where $n(IE_{AM})$ is the number of integrity errors, $n(AU)$ is the total number of authentication actions, and w_{IEam} is a weighting factor. Note that identity management solutions are also part of the authentication mechanism.

Authentication Strength (*AS*) is an aggregated metric that depicts the overall security level of the authentication solution. User-dependent Authentication Strength (AS_{usr}) can be composed from the normalized and scaled component metrics for that user:

$$AS_{usr} = w_{AIU} \cdot \overline{AIU_{usr}} + w_{AIS} \cdot \overline{AIS_{usr}} + w_{AII} \cdot \overline{AII_{usr}} + w_{AMR} \cdot \overline{AMR_{usr}} + w_{AMI} \cdot \overline{AMI_{usr}}, \quad (9)$$

where w_x is the weighting factor of component x, and '¯' denotes normalization and uniform scaling of the component metrics. Note that the weighting should be carefully designed to avoid instability of the overall equation. Overall Authentication Strength is the average of the Authentication Strengths of all users:

$$AS = \frac{1}{N} \cdot \sum_{i=1}^{N} AS_i, \quad (10)$$

where *N* is the number of users controlled by the authentication mechanism.

D. Authorization

The main measurement interest in authorization is the security strength of authentication and access control. Authorization decomposition for GEMOM is shown in Figure 8, with the corresponding BMCs in Table X.

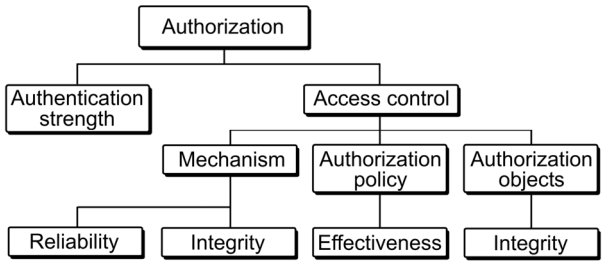


Figure 6. GEMOM authorization decomposition.

TABLE X. BMCs OF AUTHORIZATION

Symbol	Basic Measurable Component
AS	Authentication Strength (see Eq. 10)
ACMR	Access Control Mechanism Reliability
ACMI	Access Control Mechanism Integrity
APE	Authorization Policy Effectiveness
AOI	Authorization Object Integrity

Authorization policy (Req. 3.1) forms the basis for the entire authorization mechanism, while seamless co-operation with the authentication mechanism is also crucial (Req. 3.3). Consequently, the correctness and effectiveness of authentication – Authentication Strength (*AS*) – is a core metric. In GEMOM, the access control mechanism can be varied adaptively based on the authentication strength (Req. 3.2), using either *identity* or *role-based* mechanisms. Access control mechanisms can be assigned strengths that correspond to the authentication strength. As shown in Figure 6, the access control mechanism should have an adequate level of integrity and reliability. Authorization Policy Effectiveness is the main effectiveness dimension. Functionalities that revoke authorization (Req. 3.4) and delegate privileges (Req. 3.6) are part of the access control policy. The integrity of the authorization objects is also important (Req. 3.5).

Access control failures include actions that do not correspond to access control rules, such as granting access to

unauthorized users and denying of access to authorized ones. Access Control Mechanism Reliability (*ACMR*) addresses the ability of the access control mechanism to correctly and effectively perform its required functions under the conditions in the operational environment. *ACMR* depends on the following parameters:

$$ACMR = f(T_{ACNRreq}, T_{ACMRtest}, R_{ACMRasm}), \quad (11)$$

where $T_{ACMRreg}$ is the time spent on access control policy and engineering requirements, $T_{ACMRtest}$ is the time spent on testing the access control mechanism and $R_{ACMRasm}$ is the reliability of adaptive access control maintenance carried out by the Adaptive Security Management. The first two affect the parameter *E* in Eq. 3.

Authorization has an important role for data integrity in that the data should not be changed without proper authorization. The objective of the Access Control Mechanism Integrity (*ACMI*) is for the access control mechanism to function correctly according to access control rules and support data integrity. Access control failures are actions that do not correspond to access control rules. In general, *ACMI* can be measured during the data gathering period as:

$$ACMI = \frac{n(AC)}{n(AC) + w_{IEac} \cdot n(IE_{AC})}, \quad (12)$$

where w_{IEac} is a weighting factor, $n(IE_{AC})$ is the number of integrity errors in access control, and $n(AC)$ is the total number of access control actions.

Authorization Policy Effectiveness (*APE*) denotes how effective the authorization policy is at performing its required functions under the potentially hostile conditions of the operational environment. Authorization policy identifies the specific users, user groups and types of users controlled by the authorization mechanism. It also specifies the permitted operations and authorization levels on authorization objects, along with delegation privileges and depth. Authorization Policy Effectiveness is enforced by the authentication and access control mechanisms. Table XI shows two operational security metrics used to address Authorization Policy Effectiveness. Development of predictive *APE* metrics requires attack modeling.

TABLE XI. EXAMPLES OF OPERATIONAL *APE* METRICS

Symbol	Basic Measurable Component
APE^{op1}	Number of authorization incidents for each reporting period/average number of authorization incidents
APE^{op2}	Hours used to manage authorization policy/average of hours used to manage authorization policy

In GEMOM, authorization objects are namespaces, topics, metadata, and messages. Authorization rights can be granted for a single authorization object. Authorization Object Integrity (*AOI*) depends on the Authorization Object Structure (*AOS*) and integrity errors caused by authorization objects IE_{AO}:

$$AOI = f(AOS, IE_{AO}). \quad (13)$$

Access Control Effectiveness (*ACE*) can be based on normalized and scaled Access Control Mechanism Reliability and Authorization Policy Effectiveness:

$$ACE = w_{ACMR} \cdot \overline{ACMR} + w_{APE} \cdot \overline{APE}, \quad (14)$$

where w_x is the weighting factor of *x*. Access Control Correctness (*ACC*) can be based on the normalized and scaled Access Control Mechanism Integrity and Authorization Object Integrity:

$$ACC = w_{ACMI} \cdot \overline{ACMI} + w_{AOI} \cdot \overline{AOI}, \quad (15)$$

where w_x is the weighting factor of *x*. Authorization strength can be calculated from the normalized and scaled authorization BMCs similar to how Authentication Strength was calculated in Eq. 9.

Metrics from the Common Vulnerability Scoring System (CVSS) [35], which is part of the Security Content Automation Protocol (SCAP) [36], can be used to depict how easy or difficult it is to access and exploit a known vulnerability in the system. During the risk management process, a known vulnerability might be deliberately allowed to remain in the system. CVSS' *access vector metric* measures whether vulnerability is exploitable locally or remotely, and the *access complexity metric* measures the complexity of an attack that would be required to exploit the vulnerability once an attacker has access to the SuI.

E. Confidentiality

The decomposition of confidentiality requirements is shown in Figure 7 along with the identified BMCs in Table XII. The main components of confidentiality are cryptographic protection, physical security, and access control.

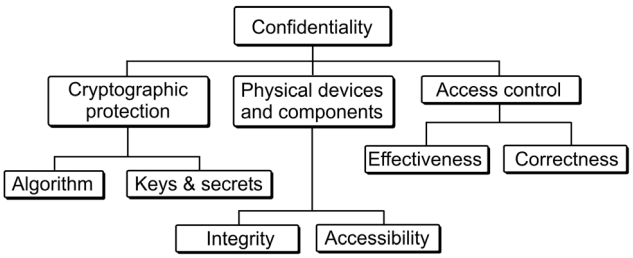


Figure 7. Confidentiality decomposition.

Special cryptographic algorithm metrics [37] can be used to measure the strength of cryptographic protection implemented by end-to-end confidentiality algorithms (Req. 4.4). It is possible to use different algorithms based on the required confidentiality level. See examples of cryptographic

algorithm metrics in Table XIII. Adequate access control is a prerequisite for end-to-end-confidentiality, assured by access control correctness and effectiveness metrics. Messages, logs, metadata, and traceability information should only be delivered to authorized recipients (Req. 4.1 and Req. 4.2). The confidentiality classification requirement (Req. 4.3) also belongs to the access control function. Protection of physical devices and components which process and conserve data (storage and memory), including protection of unauthorized access to them, is vital, as stated by Req. 4.5.

TABLE XII. BMCS OF CONFIDENTIALITY

Symbol	Basic Measurable Component
CCA	Cryptographic protection of confidentiality algorithm(s)
CCS	Cryptographic protection of keys and secrets
ACE	Access Control Effectiveness (see Eq. 14)
ACC	Access Control Correctness (see Eq. 15)
PDCI	Physical Devices and Components Integrity
PDCA	Physical Devices and Components Accessibility

TABLE XIII. SOME CRYPTOGRAPHIC ALGORITHM METRICS [36]

Symbol	Metrics
KL	Key Length
ACO	Algorithm Complexity
ATS	Attack Steps
ATT	Attack Time

CCA metrics address the cryptographic strength of the overall confidentiality algorithm solution, whereas *CCS* metrics concentrate on the correctness and effectiveness of key and other secret management mechanisms. The quality of the confidentiality key and secret management architecture, *CECKSM*, also has a strong effect on *CCS*. In other words:

$$CCS = f(KL_C, CECKSM), \quad (16)$$

where KL_C is the key length of the confidentiality algorithm. The overall confidentiality algorithm solution strength is:

$$CCA = f(CCS, ACO_C, ATS_C, ATT_C), \quad (17)$$

where ACO_C is algorithm complexity, ATS_C attacks steps metric, and ATT_C attack time metric of the confidentiality algorithm.

Physical devices and components of the GEMOM system include server and client computers, smart cards, and networking equipment. The physical protection of accessibility and integrity of the servers in GEMOM is typically assumed to be high; they reside in buildings that have high physical security measures and protect the accessibility and integrity of client computers. A three-value ('high-medium-low') ordinal scale is practical in GEMOM for the assessment of the physical accessibility *PDCA*, and integrity, *PDCI*.

Confidentiality strength can be calculated as a weighted summation of the normalized and scaled confidentiality BMCs, similar to the way in which Authentication Strength is calculated in Eq. 9. CVSS includes the *confidentiality impact metric* of CVSS for measuring the impact that successful exploitation of vulnerability in the system would have on confidentiality.

F. Integrity-enforcing Mechanisms

The following sub-section discusses *integrity-enforcing mechanisms*: integrity algorithms and related keys and secrets. GEMOM requirements emphasize messages, logs, and metadata integrity (Req. 5.1), environment integrity (Req. 5.2) and persistent data integrity (Req. 5.3), with the latter two also addressed in the confidentiality decomposition. In general, environment integrity can be measured by security assurance metrics, like test coverage, and results from security and robustness testing tools.

TABLE XIV. BMCS OF INTEGRITY-ENFORCING ALGORITHMS

Symbol	Basic Measurable Component
CEIA	Correctness and Effectiveness of Integrity Algorithm(s)
CEIAS	Correctness and Effectiveness of Cryptographic Keys and Secrets used in Integrity Algorithm(s)

Cryptographic algorithm metrics, as in the case of confidentiality algorithms, can be used to measure the security strength of integrity-enforcing mechanisms, which consist of Correctness and Effectiveness of the Integrity Algorithm(s) (*CEIA*) and the Correctness and Effectiveness of Cryptographic Keys and Secrets used in Integrity Algorithm(s) (*CEIAS*), see Table XIV. *CEIAS* depends on the security strength of keys and secrets and the Correctness and Effectiveness of the Integrity Key and Secret Management (*CEIKSM*):

$$CEIAS = f(KL_I, CEIKSM), \quad (18)$$

where KL_I is the key length of the integrity algorithm. The overall correctness and effectiveness of integrity-enforcing the algorithm(s) is:

$$CEIA = f(CEIAS, ACO_I, ATS_I, ATT_I), \quad (19)$$

where ACO_I is the algorithm complexity, ATS_I is the attack steps metric, and ATT_I is the attack time metric of the integrity algorithm.

The *Integrity impact metric* of CVSS measures the impact of a successfully exploited vulnerability (none, partial, complete) on integrity.

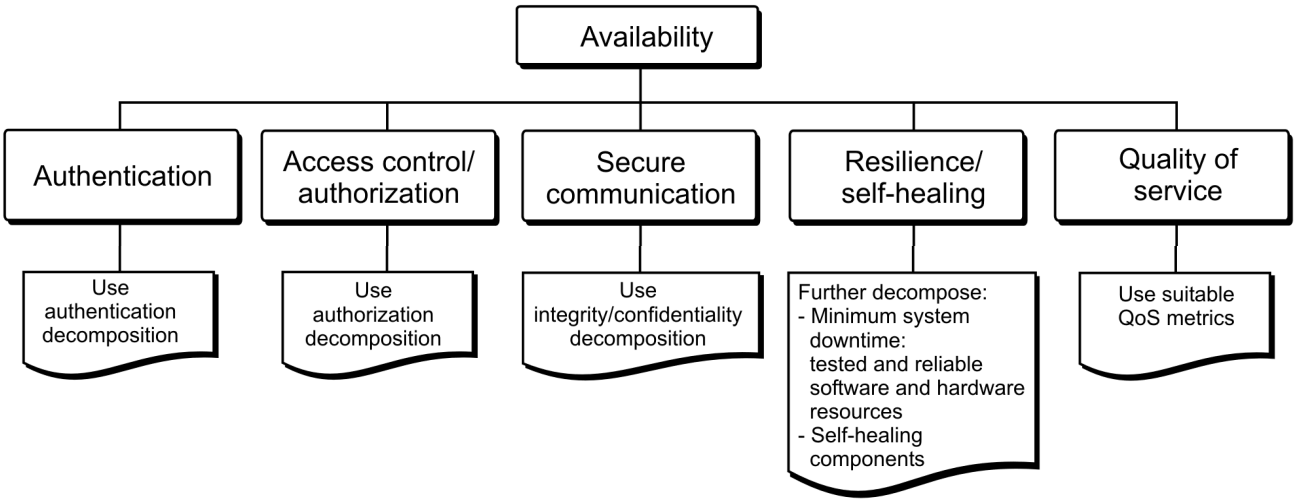


Figure 8. Availability decomposition.

G. Availability

Decomposition for GEMOM availability requirements is shown in Figure 8. As the resilience of the GEMOM system is a critical need, the loss of availability can be a major security complication. Furthermore, when executing resilience or self-protection actions, it is very important to preserve the overall security performance and level. The main availability requirements of GEMOM – robustness to faults (Req. 6.1), self-healing capabilities (Req. 6.2), and sudden reconfiguration capabilities (Req. 6.3) – emphasize the system's resilience.

Metadata must also be available when spawning new redundancy (Req. 6.4). Moreover, authentication, authorization and secure communication (via confidentiality and integrity) are crucial objectives that are considered to be preconditions for availability.

QoS performance metrics are also part of the availability decomposition, offering important availability information, especially for the detection of DoS attacks.

Availability has traditionally been measured as the percentage of time for which the target *system is* '*up*' or, in other words, when *information is available* from the system. However, this notion does not capture degraded states, a non-binary scale between a system being 'up' and 'down', which a resilient system such as GEMOM can demonstrate. GEMOM Availability metrics AV can be based on the following parameters:

$$AV = f(AS, AU, I, C, RI, QI), \tag{20}$$

where AS is average authentication strength, AU is authorization effectiveness, I is integrity effectiveness, C is confidentiality effectiveness, RI is the Resilience Indicator, and QI is the QoS indicator. The Resilience Indicator (RI) is based on the following parameters during the data gathering period:

$$RI = f(\min(SD), SHP, SRP), \tag{21}$$

where min(SD) is the minimum system down-time, SHP is Self-Healing Performance and SRP is Sudden Reconfiguration Performance. The self-healing requirement requires the system to be able to automatically create new redundancy of its operation.

The Self-Healing Performance (SHP) depends on the following parameters:

$$SHP = f(SHSR, TPRA), \tag{22}$$

where $TPRA$ is the Temporal Performance of Self-Healing Actions and $SHSR$ is the Self-Healing Success Rate.

The Self-Healing Success Rate ($SHSR$) is

$$SHSR = \frac{n(SSHA)}{n(SHA)}, \tag{23}$$

where $n(SSHA)$ is the number of successful self-healing actions and $n(SHA)$ is the total number of self-healing actions during the data gathering period. Similarly, the Sudden Reconfiguration Performance (SRP) of resources can be tracked by logging the success rate and temporal performance.

Until recently, QoS and security metrics lived in separate worlds. Some security attacks have affected application performance, and the most important objective of QoS management is to ensure application performance [16]. The GEMOM monitoring approach uses both metrics for security availability and application performance measurement. In GEMOM, QoS monitoring is founded on anomaly monitoring, which consists of anomaly detection processes and a hierarchy of anomaly correlation processes, and combining the output anomalies from a set of models in anomaly detectors. The *anomaly models* describe the various

aspects of the normal patterns, while *misuse models* refer to the abnormal patterns of the system. Space limitations prevent a more detailed investigation of QoS metrics in this study. The QoS indicator (*QI*) is a high-level indicator of the overall QoS level and is calculated from the metrics based on the anomaly and misuse models.

TABLE XV. GEMOM MACHINE-RELATED COUNTERS

Symbol	Counter
CIT, CPT	CPU Idle Time, CPU Processing Time
AM	Available Memory
PA	Paging Activity
BU, max(*B)*	Bandwidth Utilization, Maximum Bandwidth
L, V	Latency, Visibility between two machines

TABLE XVI. PUBLISH/SUBSCRIBE ACTIVITY COUNTERS

Symbol	Counter
PPN, PPT	Publications Per Namespace, Topic
PPB, PPC	Publications Per Broker, Client
MPB, MPC	Messages Per Broker, Client
DPB, DPC	Protocol Breaches Per Broker, Client
NN, NT	Number of Namespaces, Topics

In GEMOM, some averaged *counters* that measure machine- and functionality-related information are calculated and updated at certain time intervals. They are examples of intrinsic measurement functionalities that support the measurability of availability. Counters utilized in the GEMOM availability, resilience, and QoS measurements are listed in Tables XV and XVI [16]. Increased bandwidth utilization and latency are symptoms of DoS attacks. Publication and subscription activity counters are used to further investigate whether namespaces or topics are under attack.

The *Availability impact metric* of CVSS measures the impact of a successfully exploited vulnerability on availability, using values of *none*, *partial*, and *complete*.

H. Non-Repudiation

Intuitively, non-repudiation can be seen as a stronger variant of authentication, in which identities must be verified by proof-of-identity mechanisms. Core evidence in non-repudiation is the identity of origin, receipt, submission and/or delivery of messages. Non-repudiation can be implemented using a trusted third party or, in some cases, without using one. A decomposition of non-repudiation [26] is shown in Figure 9 and the identified BMCs are shown in Table XVII.

The evidence should be consistent and reliable and its integrity should be protected. The originators and receptors must provide proof-of-identity (Req. 7.1) as well as published and received messages (Req. 7.2).

Proof-of-identity evidence is fully consistent if all *identity conclusions* from the collection of proof-of-identity evidence E match. Identity conclusion is a real-world identity, as shown by the proof-of-evidence material. Let E_{ID} be the subset of E containing evidence that all identity conclusions are the same and presenting the majority of the results in E. Then Consistency of the Proof-of-Identity Evidence (*CPIE*) can then be defined as follows:

$$CPIE = \frac{n(E_{ID})}{n(E)}, \tag{24}$$

where $n(E_{ID})$ is the number of elements in E_{ID} and $n(E)$ is the number of elements in E.

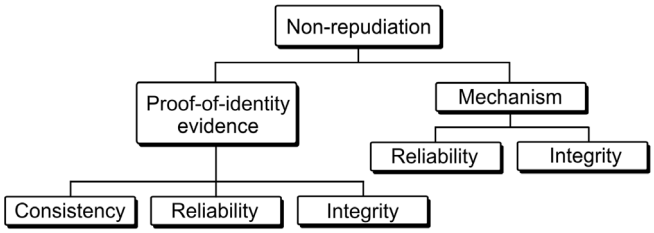


Figure 9. Non-repudiation decomposition [26].

TABLE XVII. BMCS OF NON-REPUDIATION

Symbol	Basic Measurable Component
CPIE	Consistency of Proof-of-Identity Evidence
RPIE	Reliability of Proof-of-Identity Evidence
IPIE	Integrity of Proof-of-Identity Evidence
RNRM	Reliability of Non-Repudiation Mechanism
INRM	Integrity of Non-Repudiation Mechanism

The reliability of Proof-of-Identity Evidence (*RPIE*) depends on the following factors:

$$RPIE = f(T_{3p}, R_{POIA}), \tag{25}$$

where T_{3p} is a trust and reputation function of the third party providing proof-of-identity and R_{POIA} is the reliability of the proof-of-identity algorithm(s) in use.

The Integrity of Proof-of-Identity Evidence (*IPIE*) metrics can be designed based on the general integrity metrics of Eq. 4. Integrity errors are possible in the chain-of-proof leading to the identity conclusions: in message communication, relaying of data, storage processing, and archival and backup procedures. Furthermore, problems in application software, hardware, operating systems telecommunications equipment, user data, system data and

external interfaces [34] can contribute to integrity errors in the proof-of-identity evidence.

The Reliability of Non-Repudiation Mechanism (*RNRM*) addresses the reliability of the non-repudiation mechanism and its associated processes, while the Integrity of Non-Repudiation Mechanism (*INRM*) is concerned with the integrity of them. The technical part of *RNRM* and *INRM* metrics can be based on the relevant general metrics models discussed above. The human behavioral (process) part depends on reputation and trust issues.

I. Adaptive Security

Requirements 1.1, 1.2, and 1.3 refer to the need for an adaptive trust metric that can be used when making decisions to change system parameters in authentication, authorization, and confidentiality management. In particular, authentication mechanisms, authorization mechanisms and the confidentiality algorithms will be varied based on this metric.

A user-dependent trust metric, T_{UID}, is a function of context, authentication strength, threat information, and a user-dependent trust function:

$$T_{UID} = f(\kappa, \tau, AS, \xi, t), \tag{26}$$

where *UID* is user ID, κ is context information, τ is time instant, AS_{UID} is the user's authentication strength, ξ is a variable reflecting the threat situation, and t is a trust function. Threat variable ξ can be based on off-line threat and vulnerability information and measured evidence of applicable increased threat(s) in the system.

Let the system's confidence in the trust metric T_{UID} be c_{trust}. The adaptive trust indicator AT_{UID} used for adaptive security management decision-making is:

$$AT_{UID} = c_{trust} \cdot T_{UID}. \tag{27}$$

The trust function t represents the system's trust in the user's behavior. If the user behaves well, the value of the trust function increases as a function of time asymptotically, according to a planned trust management solution. Bad behavior decreases the next value of the user's trust function. Examples of bad behavior in GEMOM include attempts to read or destroy a topic without rights, spamming, and simultaneous logging attempts.

J. Summary of BMC Sources

Table XVIII summarizes the origin of BMCs for security metrics focusing on security-enforcing mechanisms in GEMOM.

The numbers in the columns represent the security dimensions of Figure 6 (1 = adaptive security, 2 = authentication, 3 = authorization, 4 = confidentiality, 5 = integrity, 6 = availability, and 7 = non-repudiation). An '×' indicates the use of the security metrics of the respective BMC type in GEMOM [16].

TABLE XVIII. SOURCES OF BASIC MEASURABLE COMPONENTS [16]

BMC Source	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Metrics by security requirement decomposition	×	×	×	×	×	×	×
Cryptographic strength metrics		×		×	×		×
Metrics based on anomaly and misuse models (attacker-oriented weakness metrics)		×	×			×	
Availability metrics based on QoS application performance metrics						×	
System intrinsic security-relevant metrics						×	
Trust and reputation metrics	×	×					
Vulnerability metrics	×	×	×	×	×	×	×

K. Confidence, Trust and Trustworthiness Calculation

The following sub-section introduces a framework for assessing and calculating the trustworthiness of the measurements of the overall security of the system through a combination of security-based trust and trust-based security [38], see Figure 10. As sub-frameworks, the framework (*i*) contains security, trust, and confidence level calculations and (*ii*) maps trust and confidence into a trustworthiness metric. These frameworks are based on an earlier framework presented in [18], which is modified by separating trust and confidence, and then combining them to form a trustworthiness metric. The definitions of trust, confidence, and trustworthiness are similar to those presented in [39], with the following exceptions:

- **Trust** means the level of trust in the reliability of the estimation of the security level of each BMC;
- **Confidence** means the level of accuracy of and/or the assurance in the above mentioned trust relationship; and
- **Trustworthiness** means the level of trust in the reliability of the estimation of the security level of each BMC and a measurement of the degree to which the accuracy of and/or the assurance in this trust is trustworthy and can be verified.

The values of trust and confidence are both expressed as a number between zero and one, based on Bayesian statistics. A trust value equal to one indicates absolute trust and a value close to zero indicates low trust. A confidence value equal to one indicates high confidence in the accuracy of the trust value and a value close to zero indicates low confidence. Furthermore, a trustworthiness value close to zero indicates untrustworthiness and a value close to one indicates high or complete trustworthiness.

Initially, the user or expert assigns the default security levels. The normalized Security Level SL_n, $1 \le SL_n \le 10$, is calculated as follows from the measured Security Level (SL):

$$SL_n = \frac{SL}{\frac{\max(SL) - \min(SL)}{9}} + 1, \tag{28}$$

where max(*SL*) is the maximum value of *SL* and min(*SL*) is its minimum value. *SL* values from 1 to 4 indicate low security, from 5 to 7 represent good security, and from 8 to 10 indicate high security.

TABLE XIX. TRUSTWORTHINESS OF THE SECURITY MEASUREMENT

Decomposition	**BMC**	**Meas. sec. level** [1...10]	**Est. trust value** [0...1]	**Conf-idence value** [0...1]	**Trust-worth-iness** [0...1]
Authentication	AIU	*AIU*	t_{AIU}	c_{AIU}	T_{AIU}
	AIS	*AIS*	t_{AIS}	c_{AIS}	T_{AIs}
	…	…	…	…	…
Authorization	AS	*AS*	t_{AS}	c_{AS}	T_{AS}
	…	...	…	…	…
…	…	…	…	…	…

In the following, $BMC \in B$, where *BMC* denotes a BMC under investigation and *B* is the collection of all BMCs of the SuI. Similarly, $SL \in S$, where *SL* denotes a Security Level and *S* is the collection of all security levels associated with *BMC*. In order to calculate trustworthiness, like [39] and [40], some notations, associated with each $BMC \in B$ and its Security Level *SL* are defined as follows:

- $t\{BMC, SL\}$: trust value of *BMC* for security level *SL*. It has the property of $0 \le t\{BMC, SL\} \le 1$;
- $\sigma\{BMC, SL\}$: standard deviation of trust value of *BMC* for security level *SL*; and
- $c\{BMC, SL\}$: confidence value of *BMC* for security level *SL*. It also has the property of $0 \le c\{BMC, SL\} \le 1$.

Our framework uses the modified Bayesian approach [39][40][41] to evaluate trust in the security measurement of a BMC. The measurement of the security level of a BMC is assumed to be trusted with the probability of θ. In the modified Bayesian approach, several distributions can be used to represent θ, such as Beta, Gaussian, Poisson, and Binomial. The Beta distribution is the most promising of these due to its flexibility and simplicity, and because its conjugate is Beta distribution [40][41]. The trust value of the *SL* of a *BMC* can be calculated as the expectation value of the Beta distribution $Beta(\theta, \alpha, \beta)$:

$$t\{BMC, SL\} = E(Beta(\theta, \alpha, \beta)) = \frac{\alpha}{\alpha + \beta}, \tag{29}$$

where α and β denote the degree of normal behaviors and misbehaviors, respectively. In this case, while normal behavior represents a security level that can be trusted, misbehavior means that the security level is low and cannot be trusted. Trust (t) values of $0 \le t < 0.2$ indicate no trust, $0.2 \le t < 0.5$ represent low trust, $0.5 \le t < 0.8$ represent good trust, and values of $0.8 \le t \le 1.0$ indicate a high level of trust (see Table XX).

The standard deviation of the trust value $t\{BMC, SL\}$ is calculated as follows, based on [40][41]:

$$\sigma\{BMC, SL\} = \sigma(Beta(\theta, \alpha, \beta)) = \sqrt{\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}}. \tag{30}$$

The confidence value of *BMC* for Security Level *SL* is calculated as follows, based on [40][41]:

$$c\{BMC, SL\} = 1 - \sqrt{12\sigma(Beta(\theta, \alpha, \beta))} = 1 - \sqrt{\frac{12\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}}. \tag{31}$$

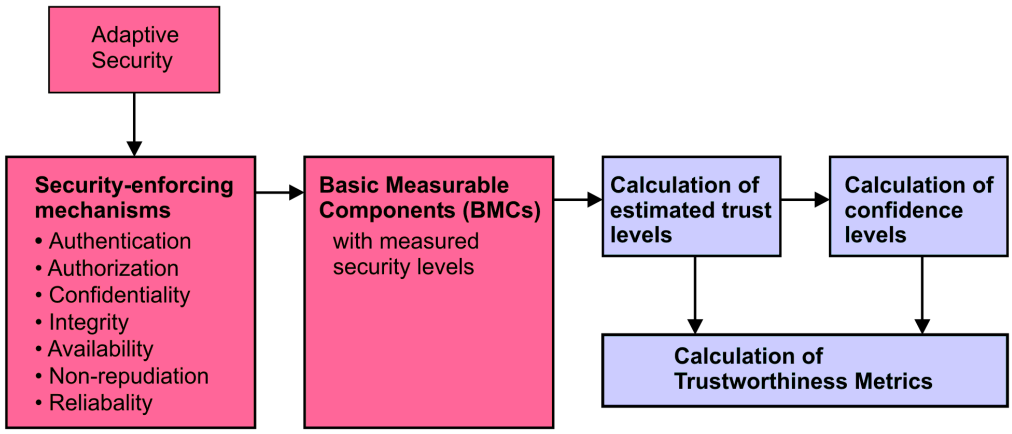


Figure 10. Trustworthiness, confidence and trust calculations.

Confidence (c) values of $0 \le c < 0.2$ indicate no confidence, $0.2 \le c < 0.5$ indicate low confidence, $0.5 \le c < 0.8$ represent good confidence, and values of $0.8 \le c \le 1.0$ indicate a high level of confidence (see Table XX).

In order to facilitate trust-based decisions, trust and confidence have been combined into a single value, as noted above. This value, trustworthiness $T\{BMC, SL\}$, is measured in the same way by combining the estimated levels of trust and confidence with some rules for the interpretation, and has the following property: $0 \le T\{BMC, SL\} \le 1$. As in [40], if the confidence value of *BMC* is high, the trust value of BMC plays a more important role for the trustworthiness. Thus, the trust of *BMC* should have a larger weight than the confidence value of *BMC*. Conversely, if the confidence value of *BMC* is low, the confidence value of *BMC* is clearly more important than the trust value of *BMC*. Therefore, the trust value of *BMC* should have less weight than the confidence value of *BMC*. The trustworthiness $T\{BMC, SL\}$, associated with $t\{BMC, SL\}$ and $c\{BMC, SL\}$ is defined as:

$$T\{BMC,SL\} = 1 - \frac{\sqrt{\frac{(t-1)^2}{x^2} + \frac{(c-1)^2}{y^2}}}{\sqrt{\frac{1}{x^2} + \frac{1}{y^2}}} \qquad (32)$$

where x and y are parameters that determine the relative importance of the trust value of BMC, with t denoting $t\{BMC, SL\}$ versus the confidence value of BMC, and c denoting $c\{BMC, SL\}$. [41] showed that the appropriate values of x and y are $\sqrt{2}$ and $\sqrt{9}$, respectively, for mapping trust and confidence to trustworthiness, and can be adjusted to the needs of a particular application. Trustworthiness (T) values of $0 \le T < 0.2$ indicate a result that is not trustworthy, $0.2 \le T < 0.5$ represent low trustworthiness, $0.5 \le T < 0.8$ indicate good trustworthiness, and values of $0.8 \le T \le 1.0$ indicate high trustworthiness (see Table XX).

The assessment and calculation of the trustworthiness of the security measurement of the system as a whole is based on the aggregation and propagation of measurements carried out in the system at different levels. This can mostly be automated since the BMCs are modeled automatically on the basis of the structural and functional relations between them.

The *trustworthiness* of the security measurement of the GEMOM system (see Tables XIX and XX) is defined by the following combination of the trustworthiness-octuple:

$$\begin{aligned} &\langle\ T_1(Authentication), T_2(Authorization), \\ &T_3(Confidentiality), T_4(Integrity), \\ &T_5(Availability), T_6(Non-repudiation), \\ &T_7(AdaptiveSecurity), \\ &T_w(T_1, T_2, T_3, T_4, T_5, T_6, T_7)\ \rangle \end{aligned} \qquad (33)$$

where $\langle T_1()...T_7() \rangle$ is the trustworthiness-septuple, and:

- Each term $T_1() \ldots T_7()$ has one or more BMCs, each of which has an associated trust value calculated as above;
- The values of the trustworthiness-septuple, in combination, form trustworthiness of the measured security level of the system as a whole; and
- $T_w(T_1, T_2, T_3, T_4, T_5, T_6, T_7) = r_1 \cdot T_1 + r_2 \cdot T_2 + r_3 \cdot T_3 + r_4 \cdot T_4 + r_5 \cdot T_5 + r_6 \cdot T_6 + r_7 \cdot T_7$ defines a measure in which each measure of trustworthiness, ($T_1 \ldots T_7$) is dependent on the weighting factor ($r_1 \ldots r_7$) attributed to it.

TABLE XX. LINGUISTIC EQUIVALENCES OF SECURITY, TRUST, CONFIDENCE AND TRUSTWORTHINESS LEVELS

Descr.	Security Level *SL* [1…10]	Trust Level *t* [0…1]	Conf. Level *c* [0…1]	Trustw. Level *T* [0…1]
High	$8 \le SL \le 10$	$0.8 \le t \le 1.0$	$0.8 \le c \le 1.0$	$0.8 \le T \le 1.0$
Good	$5 \le SL < 8$	$0.5 \le t < 0.8$	$0.5 \le c < 0.8$	$0.5 \le T < 0.8$
Low	$2 \le SL < 5$	$0.2 \le t < 0.5$	$0.2 \le c < 0.5$	$0.2 \le T < 0.5$
No	$1 \le SL < 2$	$0.0 \le t < 0.2$	$0.0 \le c < 0.2$	$0.0 \le T < 0.2$

VII. DISCUSSION

This study has concentrated on the metrics of effectiveness and the correctness of security-enforcing mechanisms of GEMOM. Intuitively, these security metrics form the core technical metrics of the system. However, there is a need for security metrics that address the overall security quality of the system from a technical perspective and with regard to secure lifecycle management. In addition, metrics must be aligned with business management objectives.

Every security metric has challenges. Three different categories of metrics presented in this study can be identified on the basis of their feasibility challenge:

1. **Hard-to-measure metrics**, for which the main challenges are measurability, attainability, availability, scalability, and portability. This category includes special metrics such as integrity and non-repudiation metrics;
2. **Hard-to-outline metrics**, the main challenges for which are objectivity, non-bias, representativeness, and contextual specificity. The category includes, in particular, reliability metrics and cryptographic algorithm metrics; and
3. **Integrated metrics**, the main challenges for which are controllability, scalability, portability, meaningfulness, representativeness, and contextual specificity. This category contains integrated metrics that are composed of several other metrics.

Many of the metrics and parameter dependencies introduced in this study require long-term data to be gathered in order to establish a sufficient reference value.

Consequently, these metrics are not directly applicable in a realistic system. If the reference is available, the metrics are applicable. Data gathering for non-repudiation metrics might take even longer since investigations of non-repudiation are quite rare. The feasibility criteria of security metrics for software-intensive systems in general are investigated in [42].

Measurement techniques of software-intensive systems, as a whole, are not yet particularly feasible; the same applies to security measurement. The disintegration of the security research field is a challenge to the development of security metrics. Moreover, the field is suffering from the lack of a common notation to describe security issues, its different components, and multi-disciplinary dependencies. Extremely subjective measures are often used, despite their lack of value [18]. There is also entrenched reliance on subjective, human, and qualitative input [4].

Several studies have criticized the feasibility of measuring security and developing security metrics to present actual security phenomena [43][44][45]. One important source of challenges is the major role that luck plays, especially in the weakest links of security solutions. In designing a security metric, it is important to consider the fact that the metric simplifies a complex socio-technical situation down to numbers or partial orders.

The U.S. National Institute of Standards and Technology (NIST) published a report on directions in security metrics research [4]. According to this report, security metrics are an important factor in making sound decisions about various aspects of security, ranging from the design of security architectures and controls to the effectiveness and efficiency of security operations. Strategic support, quality assurance, and tactical oversight are seen as the main uses of security metrics. The NIST report proposed several lines of research for security metrics: (*i*) formal models of security measurement and metrics, (*ii*) historical data collection and analysis, (*iii*) artificial intelligence assessment techniques, (*iv*) practical and concrete measurement methods, and (*v*) intrinsically measurable components.

The present study makes preliminary advances in items (*i*), (*iv*), and (*v*). Historical data collection and analysis is still required in order to further develop the ideas presented here, with the goal of formalizing and standardizing security measurement and metrics. As reiterated in the NIST report, security metrics development poses difficult and multifaceted problems for researchers. A quick resolution is not expected and it is likely that not all aspects of the challenges are resolvable [4].

VIII. Related Work

This section investigates related work from the point of view of (*i*) metrics for security-enforcing mechanisms and the overall system's security quality, (*ii*) security assurance metrics, and (*iii*) metrics addressing the secure system lifecycle. It also notes some relevant standards and related work in trust, confidence and trustworthiness calculation. Surveys of security metrics can be found in [34][46][47][48].

Metrics research is more mature in software engineering than it is in security engineering. There are software metrics for software specifications, designs, code coverage, cohesion, complexity, performance, software development processes and resources. Fenton and Pfleeger presented a comprehensive investigation of software metrics in [5]. Particular software metrics have the potential to be used in the measurement of the overall security quality of systems.

A. Metrics for Security-enforcing Mechanisms and Security Quality

Wang and Wulf [26] described a general-level framework for measuring security based on a decomposition approach. The present study enhances Wang and Wulf's idea by introducing a security metrics development process and by proposing actual security metrics and parameter dependencies for an example system. Heyman *et al*. [49] utilized a security objectives decomposition approach in order to define a security metrics framework and to interpret the results. They associated security metrics with security patterns and exploited the relationships between security patterns and security objectives to enable the interpretation of measurements. The security metrics development approach in the present study can integrate their approach: security patterns can be investigated during the security requirement and modeling phase.

The approaches of Wang and Wulf and Heyman *et al*., along with the ideas in the present study, show the feasibility of decomposition approaches for identifying measurable issues and for relating the developed metrics to original security objectives.

TABLE XXI. Comparison of Metrics Approaches for Security-Enforcing Mechanisms and Security Quality

Reference	Pros	Cons	Advances in the Present Approach
Wang and Wulf [26]	Relationships between requirements and component metrics are visible	Lack of systematic overall methodology description	Complete methodology description from threat and vulnerability analysis to a balanced and detailed metrics collection
Heyman *et al.* [49]	Association between patterns and metrics		
SCAP [36]	Aims at a standardized approach	Lack of systematic threat-driven solutions; emphasis on vulnerabilities	Complete methodology addresses both threats and vulnerabilities
Howard [51], Manadhata and Wing [52]	Attack surface is an interesting concept for overall system security quality	No systematic connection to requirements of security functions	Emphasis on requirements of security functions

The CVSS [35] (Common Vulnerability Scoring System) is a global initiative designed to provide an open and standardized method for rating information technology vulnerabilities, an example of weakness metrics. The CVSS,

along with some other security vulnerability and weakness enumerations, has been integrated by the NIST into its Security Content Automation Protocol (SCAP) [36]. While this is an important standardization effort in vulnerability management, CVSS and SCAP are not complete solutions. They lack approaches to obtain evidence of the security strength of security-enforcing mechanisms and methodologies to relate actual metrics to original security objectives. For example, CVSS can tell that the highest risk of a certain application is buffer overflow, but it cannot identify the potential operational impact of a buffer overflow [50].

Another weakness metric, the *attack surface* of a software system, is parts that can be accessed by unauthenticated users, such as attackers, including the set of entry points, exit points, the set of channels and the set of non-trusted data items. Howard [51] informally introduced the notion of attack surface, and Manadhata and Wing [52] proposed an abstract attack surface measurement method, based on Howard's notion. Table XXI compares these approaches.

B. Security Assurance Metrics

The metrics framework proposed by Bulut *et al*. [53] addresses the security assurance of telecommunication services and can be integrated to the present study's metrics development approach via reliability metrics – security assurance increases the reliability of security-enforcing mechanisms. In Bulut *et al*.'s approach, a metric defines the process towards a normalized assurance level (one out of five levels) for an object in the infrastructure. Assurance metrics include test coverage and software maturity metrics. Unlike the present study, Bulut *et al*. did not present any metrics development methodology, instead emphasizing metrics selection. Another NIST effort, the Software Assurance Metrics and Tool Evaluation (SAMATE) project [54], sought to help answer various questions on software assurance, tools, and metrics.

C. Metrics for Secure System Lifecycle

The scope of this study does not include secure system lifecycle, organizational, and business management. However, it does investigate some approaches that could be integrated into the security metrics development process. Chandra and Khan [55] introduced a three-stage security estimation life cycle. The first stage, the input stage, is analogous to the threat and vulnerability analysis stage of the present study's metrics development approach. The second stage, the 'security estimation stage,' corresponds with the subsequent stages in the present approach. Finally, the 'output stage' emphasizes the overall analysis.

D. Standardization

There have been a number of major standardization and recommendation efforts for security evaluation and the certification of technical systems. However, each of these have only achieved limited success in advancing security measurability [4]. This is largely because the standards are rigid, created for certification, and carrying out their processes is time and money-consuming. The approach presented in this study and similar ones, in which the relationships between security requirements and the resulting metrics are clearly visible in the early phases of R&D, can considerably reduce the time and money needed for certification due to early problem solving. The most widely used of these efforts is the Common Criteria (CC) ISO/IEC 15408 International Standard [56] for security evaluation. During the CC evaluation process, an Evaluation Assurance Level (EAL), from EAL1 to EAL7, is assigned to the SuI. The CC standard is based on a combination of several earlier standards, including TCSEC (Trusted Computer System Evaluation Criteria) [57], ITSEC (Information Technology Security Evaluation Criteria) [58], CTCPEC (Canadian Trusted Computer Product Evaluation Criteria) [59], and FC (Federal Criteria for Information Technology Security) [60]. Interpretations of the TCSEC have been published so that they can be applied to other contexts such as the TNI (Trusted Network Interpretation of the TCSEC) [61].

E. Trustworthiness Calculation

Zouridaki *et al*. [39][41] and Li and Li [40] introduced methodologies for calculating trustworthiness. They use modified Bayesian approaches by combining trust and confidence in their methodologies. The lack of protection against malicious attacks and/or the leakage of sensitive information are challenges in their approaches, however. The present study advanced these approaches by providing a synthesis of risk-based security, security-based trust and trust-based security into one supra-additive synergistic framework.

IX. Conclusions and Future Work

While systematic approaches for security metrics development are desirable, they are rare because the current practice of information security is a highly diverse field; widely accepted models, methods, and tools are missing.

The present study has proposed high-level security metrics and parameter dependencies for an example distributed messaging system, GEMOM. The metrics have been developed utilizing a novel security metrics development approach based on risk-driven security requirement decomposition. The emphasis is on the effectiveness and correctness of security-enforcing mechanisms for authentication, authorization, confidentiality, integrity, availability, and non-repudiation. The developed metrics can be utilized as decision-support evidence in runtime adaptive security management and off-line system security engineering and management. Some metrics require long-term data gathering before they can be utilized for on-line monitoring and management. The main challenges relate to difficulties in attaining measurements, outlining the scope, and integrating the metrics.

Reliability and integrity are generally considered to be important objectives in security solutions, especially in authentication and authorization systems. A notable observation from the study is that reliability and integrity metrics are a solid part of all decompositions of security requirements for security-enforcing mechanisms. They can

therefore be considered as horizontal Basic Measurable Components, common to all security functions. The study clearly shows the dependencies between them and other issues. Intrinsic reliability and integrity-measurability increasing mechanisms and self-check functionalities built in the system and its modules increase the coverage of security evidence gathering.

This paper has described the synthesis of the risk-based assessment of BMCs, a security-based trust model, and a trust-based security model into one framework for assessing and calculating the trustworthiness of the development of measurable security. This combination extends the capabilities of each model and leverages their best features in order to support the adaptive development of quantifiable or measurable security.

Further work is needed in the feasibility analysis and validation of the proposed metrics as well as the metrics development approach. Another interesting direction for further experimentation and analysis is building intrinsic security-measurability functionality in the system and its components. As this study has concentrated on the security-enforcing mechanisms, further work is also needed in terms of investigating metrics for security quality in the system in general and metrics addressing secure system lifecycle, organizational, and business management. Future work will also include more rigorous algorithms for analyzing functional relationships of metrics and their composite effect, and how they influence the calculation of the trustworthiness of the system as a whole.

ACKNOWLEDGMENTS

The work presented in this article has been carried out in the GEMOM FP7 research project, partly funded by the European Commission. The authors acknowledge the contributions to the GEMOM system description, threat analysis, security requirements, GEMOM Monitoring Tool and anomaly monitoring made by various GEMOM partners.

REFERENCES

[1] R. Savola and H. Abie, "Identification of basic measurable security components for a distributed messaging system", SECURWARE 2009, Athens/Glyfada, Greece, Jun. 18-23, 2009, pp. 121-128.

[2] R. Savola, "A security metrics taxonomization model for software-intensive systems", Journal of Information Processing Systems, Vol. 5, No. 4, Dec. 2009, pp. 197-206.

[3] H. Abie, I. Dattani, M. Novkovic, J. Bigham, S. Topham, and R. Savola, "GEMOM – Significant and measurable progress beyond the state of the art", ICSNC 2008, Sliema, Malta, Oct. 26-31, 2008, pp. 191-196.

[4] W. Jansen, "Directions in security metrics research", U.S. National Institute of Standards and Technology, NISTIR 7564, Apr. 2009, 21 p.

[5] N. E. Fenton and S. L. Pfleeger, "Software metrics – a rigorous & practical approach", 2nd Ed., PWS Publishing Company, 1997, 638 p.

[6] G. Jelen, "SSE-CMM Security metrics", NIST and CSSPAB Workshop, Washington D.C., Jun., 2000.

[7] R. Savola, "Towards a taxonomy for information security metrics", QoP 2007, Alexandria, Virginia, USA, Oct. 29, 2007, pp. 28-30.

[8] R. Savola, "A taxonomical approach for information security metrics development", Nordsec 2007 Supplemental Booklet, Reykjavík, Iceland, Oct. 11-12, 2007, 11 p.

[9] D. B. Parker, "Computer Security Management", Reston Publishing Company, Reston, Virginia, USA, 1981.

[10] ITU-T Recommendation X.805, "Security architecture for systems providing end-to-end communications", 2003.

[11] D. Longley and M. Shain, "Data and computer security: dictionary of standards, concepts and terms", Macmillan, 1987.

[12] D. Gollmann, "Computer security", John Wiley & Sons, 1999.

[13] R. C. Summers, "Secure computing, threats and safeguards", McGraw-Hill, 1997.

[14] R. Savola, "Current and emerging security, trust, dependability and privacy challenges in mobile telecommunications", DEPEND 2009, Athens/Glyfada, Greece, Jun. 18-23, 2009, pp. 7-12.

[15] A. Avižienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing", IEEE Tr. on Dependable and Secure Computing, Vol. 1, No. 1, Jan./Mar. 2004, pp. 11-33.

[16] R. Savola and H. Abie, "Development of security metrics for a distributed messaging system", AICT 2009, Baku, Azerbaijan, Oct. 14-16, 2009, 6 p.

[17] R. Savola, "Requirement centric security evaluation of software intensive systems", 2nd Int. Conf. on Dependability of Computer Systems DepCOS-RELCOMEX '07, Szklarska Poreba, Poland, Jun. 14-16, 2007, pp. 135-142.

[18] R. Savola and H. Abie, "On-line and off-line security measurement framework for mobile ad hoc networks", Journal of Networks, Vol. 4, No. 7, Sep. 2009, pp. 565-579.

[19] M. Howard and D. LeBlanc, "Writing secure code", Microsoft Press, 2003, 768 p.

[20] OWASP: Open Web Application Security Project. http://www.owasp.org./

[21] C. Alberts and A. Dorofee, "Managing information security risks: the OCTAVE (SM) approach", Addison-Wesley, 2003. 512 p.

[22] H. Abie and Å. Skomedal, "A conceptual formal framework for developing and maintaining security-critical systems", International Journal of Computer Science and Network Security, Vol. 5, No. 12, Dec. 2005., pp. 89-98.

[23] R. Savola, "Development of security metrics based on decomposition of security requirements and ontologies", ICSOFT 2009, Vol. 2, Sofia, Bulgaria, Jul. 26-29, 2009, pp. 171-174.

[24] D. Firesmith, "Specifying reusable security requirements", Journal of Object Technology, Vol. 3, No. 1, Jan./Feb. 2004, pp. 61-75.

[25] A. M. Davis, "Software requirements: objects, functions and states", Prentice Hall, Englewood Cliffs, NJ, 1993.

[26] C. Wang and W. A. Wulf, "Towards a framework for security measurement", 20th National Information Systems Security Conference, Baltimore, MD, Oct. 1997, pp. 522-533.

[27] M. Howard and D. LeBlanc, "Writing secure code", Microsoft Press, 2003, 768 p.

[28] R. C. Thomas, "12 tips for designing an infosec risk scorecard (its harder than it looks)", newschoolsecurity.com, Sep. 14, 2009.

[29] H. Li and G. Jiang, "Semantic message oriented middleware for Publish/Subscribe networks", C31 Technologies for Homeland Security and Homeland Defense III. SPIE, Vol. 5403, 2004, pp. 124-133.

[30] D. Lewis, J. Keeney, D. O'Sullivan, and S. Guo, "Towards a managed extensible control plane for knowledge-based networking", LNCS Large Scale Management of Distributed Systems, Springer, 4269/2006 (0302-9743), 2006, pp. 98-111.

[31] S. Parkin, D. Ingham, and G. Morgan, "A message oriented middleware solution enabling non-repudiation evidence generation for reliable web services", LNCS, Springer, 4526/2007 (0302-9743), 2007, pp. 9-19.

[32] R. Savola and T. Frantti, "Core security parameters for VoIP in ad hoc networks", WPMC 2009, Sendai, Japan, Sep. 7-10, 2009, 5 p.

[33] L. Bernstein and C. M. Yuhas, "Trustworthy systems through quantitative software engineering", IEEE Computer Society, John Wiley & Sons, 2005, 437 p.

[34] D. S. Herrmann, "Complete guide to security and privacy metrics – measuring regulatory compliance, operational resilience and ROI", Auerbach Publications, 2007, 824 p.

[35] M. Schiffman, G. Eschelbeck, D. Ahmad, A. Wright, and S. Romanosky, "CVSS: A Common Vulnerability Scoring System", National Infrastructure Advisory Council (NIAC), 2004.

[36] M. Barrett, C. Johnson, P. Mell, S. Quinn, and K. Scarfone, "Guide to adopting and using the Security Content Automation Protocol (SCAP)", NIST Special Publ. 800-117 (Draft), U.S. National Institute of Standards and Technology, 2009.

[37] N. Jorstad and T. S. Landgrave, "Cryptographic algorithm metrics", 20th National Information Systems Security Conference, Baltimore, MD, Oct. 1997.

[38] H. Abie, "Adaptive security and trust management for autonomic message-oriented middleware", MASS 2009, IEEE Symp. on Trust, Security and Privacy for Pervasive Applications (TSP 2009), Macau, China, Oct. 12-14, 2009, pp. 810-817.

[39] C. Zouridaki, B. L. Mark, M. Hejmo, and R. K. Thomas, "E-Hermes: a robust cooperative trust establishment scheme for mobile ad hoc networks", Ad Hoc Networks, Vol. 7, Issue 6, Aug. 2009, pp. 1156-1168.

[40] R. Li and J. Li, "Towards neutral trust management framework in unstructured networks", IEEE Symposium on Trust, Security and Privacy for Pervasive Applications (TSP '09), Macau SAR, China, Oct. 12-15, 2009, pp. 771-776.

[41] C. Zouridaki, B. L. Mark, M. Hejmo and R. K. Thomas, "A quantitative trust establishment framework for reliable data packet delivery in MANETs", ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '05), Alexandria, VA, Nov. 7, 2005.

[42] R. Savola, "On the feasibility of utilizing security metrics in software-intensive systems", International Journal of Computer Science and Network Security, Vol. 10, No. 1, Jan. 2010, pp. 230-239.

[43] S. M. Bellovin, "On the brittleness of software and the infeasibility of security metrics," IEEE Security & Privacy, Jul./Aug. 2006, p. 96.

[44] D. McCallam, "The case against numerical measures of information assurance", Workshop on Information Security System Scoring and Ranking (WISSSR), ACSA and MITRE, Williamsburg, Virginia, May, 2001 (2002).

[45] J. McHugh, "Quantitative measures of assurance: prophecy, process or pipedream?", Workshop on Information Security System Scoring and Ranking (WISSSR), ACSA and MITRE, Williamsburg, Virginia, May, 2001 (2002).

[46] A. Jaquith, "Security metrics: replacing fear, uncertainty and doubt", Addison-Wesley, 2007.

[47] N. Bartol, B. Bates, K. M. Goertzel, and T. Winograd, "Measuring cyber security and information assurance: a state-of-the-art report", Information Assurance Technology Analysis Center IATAC, May 2009.

[48] R. Savola, "A novel security metrics taxonomy for R&D organisations", 7th Annual Information Security South Africa (ISSA) Conf., Johannesburg, South Africa, Jul. 7-9, 2008, pp. 379-390.

[49] T. Heyman, R. Scandariato, C. Huygens, and W. Joosen, "Using security patterns to combine security metrics", 3rd Int. Conf. on Availability, Reliability and Security (ARES), 2008, pp. 1156-1163.

[50] Systems Engineering Research Center, "Security metrics", Draft, Jan. 2010.

[51] M. Howard, J. Pincus, and J. M. Wing, "Measuring relative attack surfaces", Workshop on Advanced Developments in Software and Systems Security, 2003.

[52] P. K. Manadhata, D. K. Kaynar, and J. M. Wing, "A formal model for a system's attack surface", Technical Report CMU-CS-07-144, Jul. 2007.

[53] E. Bulut, D. Khadraoui, and B. Marquet, "Multi-agent based security assurance monitoring system for telecommunication infrastructures", CNIS 2007, Berkeley, CA, USA, Sep. 24-27, 2007, 6 p.

[54] P. E. Black, "SAMATE's contribution to information assurance", IAnewsletter, Vol. 9, No. 2, 2006.

[55] S. Chandra and R. A. Khan, "Object oriented software security estimation life cycle – Design phase perspective", Journal of Software Engineering, 2008, Vol. 2, Issue 1, pp. 39-46.

[56] ISO/IEC 15408-1:2005, "Common Criteria for information technology security evaluation – Part 1: Introduction and general model", ISO/IEC, 2005.

[57] United States Department of Defense: Trusted Computer System Evaluation Criteria (TCSEC) "Orange Book", DoD Standard, DoD 5200.28-std, 1985.

[58] Information Technology Security Evaluation Criteria (ITSEC) Version 1.2, Commission for the European Communities, 1991.

[59] Canadian System Security Centre, "The Canadian Trusted Computer Product Evaluation Criteria", Version 3.0e, Jan. 1993, 233 p.

[60] U.S. National Institute for Standards and Technology and National Security Agency, "Federal Criteria for Information Technology Security – Draft Version 1.0", 2 volumes, 1993.

[61] U.S. National Computer Security Center, "Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria – Version 1", NCSC-TG-005, 1987.

www.iariajournals.org

International Journal On Advances in Intelligent Systems
ICAS, ACHI, ICCGI, UBICOMM, ADVCOMP, CENTRIC, GEOProcessing, SEMAPRO, BIOSYSCOM, BIOINFO, BIOTECHNO, FUTURE COMPUTING, SERVICE COMPUTATION, COGNITIVE, ADAPTIVE, CONTENT, PATTERNS
issn: 1942-2679

International Journal On Advances in Internet Technology
ICDS, ICIW, CTRQ, UBICOMM, ICSNC, AFIN, INTERNET, AP2PS, EMERGING
issn: 1942-2652

International Journal On Advances in Life Sciences
eTELEMED, eKNOW, eL&mL, BIODIV, BIOENVIRONMENT, BIOGREEN, BIOSYSCOM, BIOINFO, BIOTECHNO
issn: 1942-2660

International Journal On Advances in Networks and Services
ICN, ICNS, ICIW, ICWMC, SENSORCOMM, MESH, CENTRIC, MMEDIA, SERVICE COMPUTATION
issn: 1942-2644

International Journal On Advances in Security
ICQNM, SECURWARE, MESH, DEPEND, INTERNET, CYBERLAWS
issn: 1942-2636

International Journal On Advances in Software
ICSEA, ICCGI, ADVCOMP, GEOProcessing, DBKDA, INTENSIVE, VALID, SIMUL, FUTURE COMPUTING, SERVICE COMPUTATION, COGNITIVE, ADAPTIVE, CONTENT, PATTERNS
issn: 1942-2628

International Journal On Advances in Systems and Measurements
ICQNM, ICONS, ICIMP, SENSORCOMM, CENICS, VALID, SIMUL
issn: 1942-261x

International Journal On Advances in Telecommunications
AICT, ICDT, ICWMC, ICSNC, CTRQ, SPACOMM, MMEDIA
issn: 1942-2601