Dynamic Optimal Countermeasure Selection for Intrusion Response System

Alireza Shameli-Sendi, Habib Louafi, Wenbo He, Member, IEEE, and Mohamed Cheriet, Senior Member, IEEE

Abstract—Designing an efficient defense framework is challenging with respect to a network’s complexity, widespread sophisticated attacks, attackers’ ability, and the diversity of security appliances. The Intrusion Response System (IRS) is intended to respond automatically to incidents by attenuating the attack damage and countermeasure costs. The existing approaches inherit some limitations, such as using static countermeasure effectiveness, static countermeasure deployment cost, or neglecting the countermeasures’ negative impact on service quality (QoS). These limitations may lead the IRS to select inappropriate countermeasures and deployment locations, which in turn may reduce network performance and disconnect legitimate users. In this paper, we propose a dynamic defense framework that selects an optimal countermeasure against different attack damage costs. To measure the attack damage cost, we propose a novel defense-centric model based on a service dependency graph. To select the optimal countermeasure dynamically, we formulate the problem at hand using a multi-objective optimization concept that maximizes the security benefit, minimizes the negative impact on users and services, and minimizes the security deployment cost with respect to the attack damage cost.

Index Terms—Intrusion, Response system, Attack damage cost, Vulnerabilities surface coverage, Multi-objective optimization.

1 INTRODUCTION

Designing a proper defense framework requires comprehensive understanding of attackers’ and defenders’ abilities. The Intrusion Detection System (IDS) detects hostile activities or exploits in a network, while the Intrusion Response System (IRS) protects the network by selecting appropriate countermeasures to effectively handle malicious activities (e.g., disabling a daemon or adapting the firewall configuration) [1]. The IRS model comprises four main components: attack damage cost, countermeasure positive effect, countermeasure negative impact, and countermeasure deployment cost. The attack damage cost represents the amount of damage to an attack target when the existing security controls are either unavailable or ineffective [2]. The countermeasure positive effect represents how much a countermeasure can protect resources, while the negative impact refers to the impact of a countermeasure on services’ availability and users’ access. The countermeasure deployment cost refers to the administration cost or complexity of adding a security policy in security appliances. Designing a proper IRS framework inherits some challenges:

- IRS Components Evaluation: The attack damage cost should be evaluated accurately; otherwise, the automated IRS may needlessly reduce network performance or wrongly disconnect users from the network. Thus, a trade-off between network performance and network security should be established. The countermeasure positive effect component should consider several factors: 1) actual protection of services at different defense points, 2) already deployed active countermeasures, and 3) countermeasures success/failure to mitigate attacks. For the countermeasure negative impact, we need to take into account the impact of countermeasures on online users and affected services.

- IRS Components Combination: To select an optimal countermeasure at attack time, we need to combine the IRS components. However, those components impact different security attributes and do not use the same measurement unit, which makes the combination not straightforward.

- IRS Dynamicity: The static evaluation of IRS components results in several weaknesses: 1) the countermeasures can be predicted by attackers, 2) there is no mechanism for evaluating the effectiveness of deployed countermeasures over time (non-adaptive), or 3) it does not reflect the dynamicity of network status; i.e., added/removed services, vulnerabilities, or online users. Hence, a dynamic IRS is needed. However, each IRS component presents specific challenges. The attack damage cost component should consider the number, frequency, and steps of attacks, which vary considerably. The countermeasure positive effect component should take into account the deployment cost over time and improved security state. The countermeasure negative impact component should consider the dynamicity of network status. The countermeasure deployment cost component should consider the state of security appliances. For example, inserting a firewall rule requires an analysis of all the existing rules in a firewall in order to determine the proper order for this rule and to commit the updates [18]. As the number of rules increases, the insertion processing time increases.

In this paper, we propose a framework for selecting and deploying optimal countermeasures to intrusions dynamically. The advantage of such deployment is that countermeasures can be fine-
tuned according to the context in which the intrusions are detected. In other words, the IRS will take into consideration the resources being targeted by the attack, resources being used by the user, the load of the system, the severity of dependency between services, the history of previous deployments (countermeasure effectiveness), and the current complexity of security defense points. Thus, countermeasures are not predictable since they are not statically determined. Since attack damage and countermeasure costs impact different attributes, we modeled our problem as a multi-objective optimization problem. Thus, our framework provides a set of best solutions (different trade-offs) and selects the optimal one based on the severity of attack damage cost. Compared to the previous work, which consider only two objectives, maximizing the security performance and minimizing the security impact on service quality, in this paper, in addition to these two objective functions, we consider a third objective function, minimizing the security deployment cost.

The main contributions of the proposed framework are as follows: 1) To evaluate the attack damage cost, we consider both the current and the potential damage costs using a service dependency graph. The former represents the damage on non-goal services, whereas the latter refers to future damage in relation to goal service (attacker target), 2) We propose a new model to evaluate the countermeasure positive effect dynamically which is based on the vulnerabilities surface coverage and countermeasure goodness. Moreover, this process takes into account the already deployed active countermeasures; thus, it avoids deploying useless countermeasures, 3) We evaluate the countermeasure negative impact on the basis of direct and backward impacts on services and users. These impacts are modeled using a service dependency graph, 4) We evaluate the countermeasure deployment cost by means of incompatibility and administration costs. We present a dynamic model to measure the complexity of deployment cost over time, and 5) In the case of detection of a compromised vulnerability in our attack defense tree, we identify all attack paths to the goal node and then for each path one optimal countermeasure is selected to secure the path. In other words, we stop the ability of the attacker from different attack paths to compromise the target by a combination of optimal countermeasures.

The rest of this paper is organized as follows: Section 2 provides the IRS background. Section 3 presents our problem statement. Section 4 provides a mathematical formulation of our optimization problem. The proposed framework will be discussed in Section 5. Section 6 discusses the various experimental results. Finally, Section 7 makes some concluding remarks.

2 RELATED WORK

In designing a cost-sensitive IRS, several models have been proposed to attain the attack damage and countermeasure costs. As explained, they do not use the same measurement unit [1], [25]. Some models attempt to put these two costs in the same measurement unit by simplifying the problem [4]–[6]. For example, Balepin et al. [6] propose to consider only availability loss of services affected by the intruder and countermeasure. Other models evaluate attack and countermeasure costs without putting them into the same measurement unit (e.g., [10], [20], [22]–[24]). These models define different parameters to calculate costs separately and then use several techniques to compare them. Since our proposed model lies in this category, to continue, we discuss some highly related frameworks.

Chung et al. [10] propose an attack graph model to detect a multi-step attack. When an attack is detected on a node in the attack graph, a countermeasure selection module measures the benefit of all relevant countermeasures on different places from that node to the target by considering all possible paths in the attack graph. The best countermeasure is the one that brings the highest protection and lowest resource cost and intrusiveness to the service level agreement (SLA). This approach suffers from a number of limitations. First, although they state that the problem is a multi-objective optimization one, they reduce it to a single-objective optimization problem, which provides a single solution. They combine the objectives using the well-known ROI (Return of Investment) function, which does not belong to any studied combination methods (known as scalarization [8]). Second, countermeasure cost and intrusiveness on SLA are static values initialized in advance. Therefore, the only objective that varies in function of countermeasures is the benefit objective. That is, their problem, by its nature, is a single-objective optimization problem (maximizing the benefit). Third, they define the effectiveness as the percentage of probability changes of the node, for which the countermeasure is applied, meaning that it is dynamic. However, in the experiments, they use static values. Finally, they do not consider the effectiveness overlap between all the candidate countermeasures and the already deployed active ones.

Granadillo et al. [22] propose a Return on Response Investment (RORI) model to evaluate countermeasures. Several parameters are considered to find the optimal solution: attack damage, countermeasure positive effect, countermeasure cost, and security equipment cost. They propose the countermeasure surface coverage to calculate the amount of risk reduction. This approach suffers from several limitations. First, the countermeasure surface coverage is defined as the percentage of attack surface it covers. However, in practice, it is not straightforward to establish a mapping between countermeasures and attacks. Second, in this model, it is possible to apply more than one countermeasure simultaneously. In this case, the surface of one countermeasure may be partially or totally covered by others. Since the countermeasure surface coverage is presented by a percentage, we cannot decide whether or not there is a joint surface. In the case of a possible joint surface, they use the average between the maximum and minimum surface coverage, which is not always reliable from the security perspective. Third, they do not consider the countermeasure’s negative impact on services. Sometimes, the maximum benefit brings the maximum impact on service availability. Finally, regarding multi-objective optimization, this model suffers from the same problem as [10] does.

In this paper, we propose a framework that identifies all the best countermeasures and selects the optimal one, without the aforementioned limitations. The proposed framework answers the following questions: 1) how much are the current and potential damage costs when the attack scenarios and attacker’s abilities vary, 2) how accurate is the countermeasure risk reduction, 3) how big is the countermeasure impact on QoS, and 4) how can the countermeasure and attack damage costs be efficiently balanced?

3 PROBLEM STATEMENT

Let $DS$ be a defense system deployed on a set of defense points $DP = \{d_{pi}\}_{i=1}^{n}$, each of which is intended to protect a set of services. Let $CM = \{cm_{ki}\}_{k=1}^{r}$ be a set of possible countermeasures that can be deployed on all the defense points, where $r$ is their
total number. When an attack is detected by the IDS, the attack damage cost is assessed to determine the value of the loss incurred by a compromised service. Then, the optimal countermeasure is identified and deployed to mitigate the attack with respect to the attack severity. Each countermeasure, \( cm_k \), provides three attributes: (1) Security performance: it represents the security benefit provided to the defense system when the countermeasure \( cm_k \) is activated. We note this property by \( S_P(cm_k) \), (2) Security impact on service quality: it represents how much the quality of the services is affected when \( cm_k \) is deployed. This can be restricted only to availability, since countermeasures are basically not intended to affect the service’s confidentiality or integrity. We denote this property by \( S_Q(cm_k) \), and (3) Security cost: it represents the countermeasure deployment cost. It is expressed as the incompatibility (e.g., adding a firewall rule) and administration (e.g., patching a vulnerability) costs. We denote this property by \( S_C(cm_k) \).

The goal is to identify the optimal countermeasure \( cm_k^* \) that maximizes the security performance, minimizes the security impact, and minimizes the security cost. Therefore, \( cm_k^* \) can be computed by solving the following multi-objective optimization problem.

\[
\text{MOPT} \quad \max \ S_P(cm_k) \quad \min \ S_Q(cm_k) \quad \min \ S_C(cm_k) \\
\text{s.t.} \quad S_P(cm_k) \geq s_p, \quad S_Q(cm_k) \leq s_q, \quad \text{and} \quad S_C(cm_k) \leq s_c
\]

where the constraints \( s_p, s_q, \) and \( s_c \) represent the threshold for the accepted minimum security performance at attack time, accepted maximum security impact on service quality, and accepted maximum cost of countermeasure deployment, respectively. Later in the paper, we will show how these functions can be evaluated.

4 Multi-Objective Formulation

In this section, first we formulate our problem with the multi-objective concept and show how to identify the optimal solution. Then, we provide the advantage of multi-objective formulation compared to the existing works in state-of-art.

4.1 Pareto Optimal Set Selection

Let \( \Omega \subseteq CM \) be the set of feasible countermeasures that satisfy the constraints of (1). Let \( \tilde{\Omega} \) be the set of the corresponding triplets \( \left(S_P(cm_k), S_Q(cm_k), S_C(cm_k)\right) \) of \( \Omega \) in the objective (also called criteria [8]) space. We define \( \mathcal{H} \) as a function between \( \Omega \) and \( \tilde{\Omega} \) as follows:

\[
\mathcal{H} : \Omega \rightarrow \tilde{\Omega} \\
cm_k \mapsto \left( S_P(cm_k), S_Q(cm_k), S_C(cm_k) \right)
\]

Let \( cm_i \) and \( cm_j \) be two feasible countermeasures, and \( \mathcal{H}(cm_i) \) and \( \mathcal{H}(cm_j) \) their images by \( \mathcal{H} \), respectively, which are given by:

\[
\mathcal{H}(cm_i) = \left( S_P(cm_i), S_Q(cm_i), S_C(cm_i) \right) \\
\mathcal{H}(cm_j) = \left( S_P(cm_j), S_Q(cm_j), S_C(cm_j) \right)
\]

In the multi-objective optimization, we say that \( \mathcal{H}(cm_i) \) dominates \( \mathcal{H}(cm_j) \), and we denote it by \( \mathcal{H}(cm_i) \succ \mathcal{H}(cm_j) \), if the following inequalities are satisfied:

\[
S_P(cm_i) \succ S_P(cm_j) \land S_Q(cm_i) \leq S_Q(cm_j) \land S_C(cm_i) \leq S_C(cm_j) \\
\text{or} \\
S_P(cm_i) \geq S_P(cm_j) \land S_Q(cm_i) < S_Q(cm_j) \land S_C(cm_i) \leq S_C(cm_j) \\
\text{or} \\
S_P(cm_i) \geq S_P(cm_j) \land S_Q(cm_i) \leq S_Q(cm_j) \land S_C(cm_i) < S_C(cm_j)
\]

Clearly, solution \( cm_i \) is better than solution \( cm_j \), that is, \( cm_i \) provides security performance higher than or equal to \( cm_j \) with lower or equal impact on service quality, and lower or equal security cost. However, there are many solutions that provide different trade-offs between security performance, security impact on service quality, and security cost. In this sub-space of solutions, any improvement in one objective function is automatically compensated by a deterioration in one or more of the other objectives. Therefore, to solve (1), we evaluate all the solutions that cannot be dominated by any other solution, from which we identify the optimal one. Such solutions are called Pareto optimal solutions and their images in the objective space are called Pareto optimal set. Let \( \Omega_p \subseteq \Omega \) and \( \tilde{\Omega}_p \subseteq \tilde{\Omega} \) be, respectively, the Pareto optimal solution space and the Pareto optimal set (its image by \( \mathcal{H} \)). This Pareto optimal solution space is given by:

\[
\forall cm_i \in \Omega_p, \exists cm_j \in \Omega \mid \mathcal{H}(cm_j) \succ \mathcal{H}(cm_i)
\]

4.2 Optimal Solution Selection

Several methods can be used to extract the optimal solution from the Pareto optimal set [8], [11]. In this paper, SAW is used since it provides reasonable result (see section 6.6.1). The SAW method consists in weighting all the objective functions and summing them up. It comprises two steps: normalization of the objective functions and evaluation of the final score of each countermeasure.

For each countermeasure \( cm_k \in \Omega_p \), the normalization is as follows:

\[
\tilde{S}_P(cm_k) = \max_{cm \in \Omega_p} \frac{S_P(cm_k)}{S_P(cm_i)} \\
\tilde{S}_Q(cm_k) = \min_{cm \in \Omega_p} \frac{S_Q(cm_k)}{S_Q(cm_i)} \\
\tilde{S}_C(cm_k) = \frac{S_C(cm_k)}{S_C(cm_i)}
\]

where \( \tilde{S}_P, \tilde{S}_Q, \) and \( \tilde{S}_C \) are the normalized values of \( S_P, S_Q, \) and \( S_C \), respectively. Note that this is not the unique way of normalization [11].

The final score of each countermeasure \( cm_k \), which we denote by \( Q(cm_k) \), is evaluated as follows:

\[
Q(cm_k) = w_P \cdot \tilde{S}_P(cm_k) + w_Q \cdot \tilde{S}_Q(cm_k) + w_C \cdot \tilde{S}_C(cm_k)
\]

where \( w_P, w_Q, \) and \( w_C \) are weights used to favor one objective over the others in order to select the optimal security countermeasure from the Pareto optimal set. In general, to express the relative importance between the objective functions and not to use large values, these weights should satisfy the following properties:

\[
w_P + w_Q + w_C = 1 \\
w_P > 0, w_Q > 0, w_C > 0
\]
When the attack damage cost is high, a strong countermeasure should be selected, and vice versa. This means that damage cost is aligned with the security performance. Therefore, we propose to set the value of $w_P$ as the attack damage cost. For example, if the damage cost is 90%, the sum of two other weights is 10% ($w_Q$ and $w_C$). Thus, the second and third objectives are less important. To decide about the relative importance between the second and third objectives, we introduce $\sigma$ and $\theta$. If the services provided by the company are more important than the cost of deploying a countermeasure, a higher value of $\sigma$ should be set, and vice versa. Thus, we have:

$$w_P = \text{attack damage cost}$$

$$w_Q = (1 - w_P) \cdot \frac{\sigma}{(\sigma + \theta)}$$

$$w_C = (1 - w_P) \cdot \frac{\theta}{(\sigma + \theta)}$$

$$\sigma + \theta = 1, \sigma > 0, \theta > 0$$

The optimal countermeasure, $cm_k^*$, is the one that yields the highest $Q$ value. It is given by:

$$cm_k^* = \arg\max_{cm_k \in \Omega_P} Q(cm_k)$$

4.3 Advantages

As explained earlier, first we identify all the best possible solutions, those lying on the Pareto optimal set. Then, from the latter, we extract the optimal one using the SAW method. This way, we ensure that the obtained solution belongs to the Pareto optimal set. In contrast, other works [10], [22] combine the objective functions without evaluating the Pareto optimal set. They reduce the multi-objective problem into a single objective optimization one, and select the optimal solution from the combined solution space. The question is when to convert the problem into a single objective optimization one (e.g., SAW): 1) from the beginning, on all the solution space or 2) on the Pareto optimal solution space. According to research performed in multi-objective optimization, the methods that consist in combining the objective functions without evaluating the Pareto optimal set, from the beginning, do not guarantee a non-dominated solution [8].

5 Proposed Framework

Figure 1 illustrates the proposed architecture of our automated intrusion response system. There are two data structures in our model: Attack-Defense Trees (ADT) and Service Dependency Graph (SDG). In a large network model, different ADTs can be defined based on the SDG in such a way that each ADT protects one goal service. Note that our framework works with one ADT, which is selected by the security expert. Over time, we may select another ADT, following the new strategy of the company. When an alert is raised by IDS, our framework maps the alert to the selected ADT, from which all possible attack paths to the target are extracted. Then all defense points are identified for each attack path. On the basis of the type of defense point, appropriate countermeasures are selected. Next, for all of them three functions are evaluated independently: the security benefit, security impact, and security cost. A Pareto optimal set is generated to see which countermeasures satisfy the trade-off between these three functions. Finally, the optimal countermeasure is selected from the Pareto optimal set with respect to the severity of the attack. The latter is measured by the attack damage cost component which is based on the SDG. Once the optimal countermeasure of each attack path is identified, the useless ones (those whose effect is covered by another) are eliminated. Then, the remaining countermeasures are deployed and their effectiveness at the related defense points is measured and stored in the relevant ADT. To continue, we first explain the SDG and ADT data structures, then detail the main components of our IRS as illustrated in Figure 1.

5.1 Service Dependency Graph

We say that a service $s_i$ depends on another service $s_j$ if $s_i$ needs $s_j$ to function properly. For instance, in the case of dependency between a web server and a database, if the web server is down, the database must be available and its data not be corrupted. Conversely, if the database is not available (e.g., if it is down) or its data are altered, the web server will either not provide information or provide the wrong information. We call this, functional dependency between services.

However, the attacker may exploit non-functional connections to reach his target. Let machines M1 and M2 host Apache and DB, respectively. Suppose that sshd, running on M2, does not need to communicate with any service. The attacker first compromises Apache and obtains root access on M1. Then, he uses the connection between M1 and M2 to compromise sshd’s vulnerability and become rooted in M2. Finally, he becomes able to compromise DB without using the connection between Apache and DB. We call this, non-functional dependency between services.

In this paper, we model these types of dependencies by means of a service dependency graph (SDG), the vertices being the services, and the edges being the functional and non-functional dependencies between them. To each vertex $s_k$ in the graph, we associate a vector $I(s_k)$ which represents the importance of the service (also called service value), in terms of CIA (confidentiality, integrity, and availability). Besides, to each edge between $s_i$ and $s_k$ in the graph ($s_i$ depends on $s_k$), we associate a vector that consists of three weight values that represent the dependency severity $d_{i,k}^{C,F,A} \in [0,1]$ in terms of CIA. Note that for the non-functional dependency, the CIA values are set to zero. The severity models how strong the dependency between $s_i$ and $s_k$ is, which reveals how much the damage incurred on $s_k$ will affect $s_i$. For instance, in a denial of service (DoS) attack on $s_k$, the attacker slows down the functionality of $s_k$. This decreases the service availability (A) of $s_k$, and does so on all the services that depend
on \( s_k \). In the User to Root (U2R) or Remote to Local (R2L) attack types, since \( s_k \) is under the control of the attacker, all CIA parameters of \( s_k \) and those of the services that depend on it can be affected. Both the importance of services and the dependency severity between them are defined by the security expert. As seen in Figure 1, SDG is used in measuring the attack damage cost and security impact on QoS.

5.2 Attack-Defense Tree

To understand the attacker’s progress and find the attack path to the attacker’s target, there are two approaches to model network vulnerabilities: an attack graph [12], [13] and an attack tree [14], [15]. They provide an appropriate picture of different ways for compromising a target by exploiting a sequence of vulnerabilities. The attack tree tackles the state space explosion problem, which is present in the graph, and consequently addresses the problem of visualization complexity [16]. In this paper, we use the Attack-Defense Tree which consists of two types of actions: attacker action and defender action. We define our ADT as follows:

Definition 1 (Attack-Defense Tree (ADT)). The proposed attack-defense tree is a 4-tuple \( \text{ADT} = (s_g, S_{ng}, V, D\mathcal{P}) \), where:

1) \( s_g \) is the root of the ADT, which represents the attacker-targeted service (goal service).

2) \( S_{ng} = \{s_i\}^{d}_{i=1} - \{s_g\} \) represents the set of services that are compromised (non-goal services) to reach the attacker’s target.

3) \( V = \bigcup_{i=1}^{d} V(s_i) \) is the set of vulnerabilities that are present in the network topology. For a given service \( s_i \), let \( V(s_i) = \{v_j(s_i)\}^{m}_{j=1} \) be the set of its vulnerabilities. To compromise \( s_i \) through functional dependency, the predicate \( \bigwedge_{k=1}^{k} \bigwedge_{t} v_i(s_i) \) should be true. \( k \) represents the number of disjunctions and \( t \) refers to the number of conjunctions in each disjunction. That is, at least one term of this disjunction should be true. In non-functional dependency, a service may be compromised without exploiting any vulnerability. For more details and examples, the reader is referred to [17].

4) \( D\mathcal{P} = \{d_{dp}\}^{n}_{i=1} \) is a set of defense nodes placed in the ADT to protect services. Thus, each node, \( s_i \), may have one child, representing at least a defense point. For each defense point, we assign a list of appropriate countermeasures. Some of them are general countermeasures (e.g., restarting a service/machine) though others are defined with respect to the type of service.

In a multi-step attack, the attacker compromises a set of services (non-goal) to reach his target. To do so, he has to exploit one or more of the vulnerabilities of a non-goal service in each step. Figure 2 represents a general ADT. Two types of sub-graphs are presented in the figure: star and diamond. The star sub-graph includes at least one vulnerability that results in full access to the root of that sub-graph. In the diamond sub-graph, there is no vulnerability that leads to full-access to the root of the sub-graph (limited access to root). Thus, depending on the attack type impact on the root of the sub-graph, the attacker has different opportunities to reach the next level (see (15)). If the attacker comes from the star sub-graph, he is able to compromise all the vulnerabilities of the next level. Otherwise, he has limited access and may compromise only a set of them.

![Fig. 2: General Attack-Defense Tree.](image)

The ADT is created first and then updated over time, when services or vulnerabilities are added/removed. In case of updating the ADT during attack time, the update cost does not impact the optimal countermeasure selection process, since it uses the current (not being updated) ADT. That is, the attack is based on the current ADT version, as the updated one has not known to the attacker yet.

There are four cases to update ADT. (1) If a new vulnerability \( v_x \) is discovered for a given service \( s_x \), all the services that depend on \( s_x \) are identified (using SDG). If \( v_x \) needs limited access permission to be compromised, all the identified services are connected. Otherwise, only the services that can be compromised as full access are connected. (2) In case of removing a vulnerability, protected by a countermeasure, the ADT is updated by removing the sub-graph, the root of which is \( v_x \). This keeps the ADT optimized, by eliminating useless attack paths (see Section 5.4).

(3) If a new service is added, a new sub-graph, the root of which is that service and the children are its vulnerabilities, is created. If the root can be compromised as full access, it is connected to all the vulnerabilities of the services it uses. If there is no vulnerability that leads to full-access to root of sub-graph (limited access to root), the root is connected to the vulnerabilities, which need limited access permission to be compromised. Then, each child is connected to the ADT following the same process of connecting a new vulnerability. (4) Similar to vulnerability, removing a service consists in eliminating the sub-graph rooted by that service.

5.3 Attack Damage Cost Evaluation

Attackers are smart people in reality and they know that they may not achieve their malicious goals if they just follow the standard steps to compromise the target. They try to find the vulnerable services and defense points of the network and bypass them (non-goal services) to launch multi-step attacks and compromise the target (goal service). Two techniques can be used to calculate the attack damage cost in this context [7]: attack-centric and defense-centric. In attack-centric, there is one goal (e.g., a sensitive file modification) and all other completed steps are called non-goal. In this model, if the attacker does not reach the target goal, the attack damage cost is zero. However, the defender may not know the attack type and the attacker’s real intention until the end of an accomplished multi-step attack. Therefore, we should consider the uncertainty about the attackers’ activities in the selection of the best countermeasure. Thus, we propose a defense-centric model to take into account not only the current damage but also the potential damage of the multi-step attack. The current damage is the damage incurred on compromised services (non-goal) and the potential damage represents the attack damage in the next step of attack (non-goal/target). Thus, our model not only protects against
any attack on goal-service (e.g., compromising Database) but also any attack on non-goal services (e.g., DDoS on Web service). We evaluate these two damages using the service dependency graph (defined in Section 5.1). Thus, the total attack damage cost for the whole system \( S \) from the exploited vulnerability \( v_j \) in service \( s_k \) is the ratio of the importance of the services damaged by attacker to the total importance of all services in terms of CIA. This cost is calculated as follows:

\[
D(S, s_k, v_j) = \frac{\delta \in \{C, I, A\}}{\sum_{s_i \in S} \delta} I_\delta(s_i)
\]

(11)

where \( D_C(S, s_k, v_j) \), \( D_I(S, s_k, v_j) \), and \( D_A(S, s_k, v_j) \) are attack damage cost in terms of confidentiality, integrity, and availability, respectively. \( I_C(s_i) \), \( I_I(s_i) \), and \( I_A(s_i) \) represent the importance of the service \( s_i \) in terms of confidentiality, integrity, and availability, respectively. The attack damage cost for each attribute \((C, I, A)\) is evaluated as follows:

\[
D^c(S, s_k, v_j) = D^c_c(S, s_k, v_j) + D^c_p(S, s_k, v_j)
\]

(12)

where \( D^c_c(S, s_k, v_j) \) and \( D^c_p(S, s_k, v_j) \) are the current and potential attack damage costs, respectively. To calculate the current damage cost, direct and backward propagation are considered. In the backward sub-graph of service \( s_k \), \( B(s_k) \), the damage propagation is considered for all services that have mandatory dependency on \( s_k \) directly or indirectly. In mandatory dependency, a service requires the functionality of one or many service(s) on which it directly or indirectly depends on. Thus, the current damage cost is calculated as follows:

\[
D^c(S, s_k, v_j) = D^c_c(S, s_k, v_j) + D^c_p(S, s_k, v_j)
\]

(13)

where \( D^c_c(S, s_k, v_j) \) and \( D^c_p(S, s_k, v_j) \) are the direct and backward propagated attack damages, respectively. They are given by:

\[
\begin{align*}
D^c_c(S, s_k, v_j) &= A^c_m(s_k, A^c_i(v_j)) I(s_k) \\
D^c_p(S, s_k, v_j) &= \sum_{s_i \in B(s_k)} A^b_m(s_i+1 : s_i) d_{i,i+1} I(s_i)
\end{align*}
\]

(14)

where \( A^c_m(s_k, A^c_i(v_j)) \) is the attack impact on \( s_k \), which depends on the type of attack \( A^c_i(v_j) \), when vulnerability \( v_j \) is exploited (as seen in (15)).

\[
\begin{align*}
A^c_m(s_k, A^c_i(v_j)) &= \begin{cases} 
[1, 0, 0] & \text{if } A^c_i(v_j) = \text{information leakage} \\
[1, 1, 1] & \text{if } A^c_i(v_j) = \text{remote-2-root} \\
[0, 1, 0] & \text{if } A^c_i(v_j) = \text{integrity} \\
[0, 0, 1] & \text{if } A^c_i(v_j) = \text{denial-of-service} \\
\end{cases}
\end{align*}
\]

(15)

\[A^b_m(s_i+1 : s_i)\] is the backward attack impact propagation from \( s_{i+1} \) to \( s_i \). It is calculated as follows:

\[
A^b_m(s_{i+1} : s_i) = \begin{cases} 
A^c_m(s_{i+1}, A^c_i(v_j)) & \text{if } s_{i+1} = s_k \\
A^b_m(s_i+1 : s_i)d_{i,i+2} & \text{else}
\end{cases}
\]

(16)

Since the attack starts from \( s_k \), the impact on all the parents of \( s_k \) (in service dependency graph) is equal to the impact on \( s_k \), \( A^c_m(s_k, A^c_i(v_j)) \). For other services (e.g., \( s_i \)), which depend to the parents of \( s_k \), the backward attack impact propagation depends on the type of attack impact on the service \( s_{j+1} \) and CIA dependency vector between \( s_{i+1} \) and \( s_{i+2} \). The attack impact on CIA in \( s_k \) cannot be propagated to all services in the backward sub-graph. At each step of the evaluation, the impact propagation value should be moderated on the basis of the CIA dependency between services.

To perform the potential damage cost evaluation, we follow the forward dependency direction from service \( s_k \), \( F(s_k) \), to identify all the services that can be affected in the next step of the multi-step attack. The potential damage cost is calculated as follows:

\[
D^p(S, s_k, v_j) = \sum_{s_i \in F(s_k)} D^p_c(S, s_i, v_h^*) + D^p_p(S, s_i, v_h^*)
\]

(17)

where \( D^p_c(S, s_i, v_h^*) \) and \( D^p_p(S, s_i, v_h^*) \) are the potential attack damages on service \( s_i \), which is exactly one step ahead from the compromised service \( s_k \), and the backward propagated attack damage from it, respectively. Since the attacker’s intention cannot be known in advance, it is useless to evaluate the potential attack damage for all steps ahead from \( s_k \). Thus, we gradually evaluate the attack damage on the basis of the attacker’s behavior. The direct and potential damages are given by:

\[
\begin{align*}
\bar{D}^p(S, s_i, v_h^*) &= A^p_m(s_i, A^p_i(v_h^*)) I(s_i) \\
\bar{D}^p_p(S, s_i, v_h^*) &= \sum_{s_j \in B(s_i) - \{s_a\}} A^p_m(A^p_m(s_j+1 : s_j) d_{j,j+1} I(s_j))
\end{align*}
\]

(18)

where \( A^p_m(s_i, A^p_i(v_h^*)) \) is the attack impact on \( s_i \) when \( v_h^* \) is exploited. \( v_h^* \) is the vulnerability that creates the highest damage on \( s_i \) once it is exploited. It is given by:

\[
v_h^* = \arg \max_{v_h \in V(s_i)} A^p_i(v_h)
\]

(19)

\[A^b_m(A^p_m(s_{j+1} : s_j))\] is the backward impact propagation from \( s_{j+1} \) to \( s_j \) calculated using the same formula expressed in (16).

5.4 Attack Paths Extraction

Once an attacker compromises a vulnerability, he may use different attack paths to compromise the target. Thus, when an IDS generates an alert about a compromised vulnerability, all attack paths are extracted and secured. In other words, our model stops the ability of the attacker to use different attack paths to compromise the attacker’s target when a vulnerability is compromised. Note that in the attack paths extraction, we eliminate the redundant attack paths, which visit the same sequence of defense points. This is common in an attack tree, since the goal/non-goal node(s) may have more than one vulnerability (see the star shape in Figure 2).

As shown in Figure 2, the extracted attack paths are injected into an “Optimal CM selection” engine, where for each attack path, all defense points are identified and used in selecting the optimal countermeasure. In the next subsections, we will explain how security performance, security impact on QoS, and security cost are evaluated for each candidate countermeasure.

5.5 Objective Functions Evaluation

5.5.1 Security Performance Evaluation

Our security performance evaluation is based on the countermeasure positive effect, which takes into account two features: vulnerabilities surface coverage and goodness (or effectiveness).
In case of attack, we should calculate the goodness of all countermeasures based on the history of their deployments (success or failure). Since the most recent results must be considered more valuable than earlier ones, we use the aging algorithm by dividing the history into different windows (e.g., day, week, month) confined between 0 and 2. The value of zero represents the worst case when all the history consists of only failures, while the value of two indicates the best case, i.e., only successes. To compute the total goodness of a countermeasure, we propose the following formula:

\[
G(dp_i, cm_k) = \sum_{w=1}^{n} \frac{G_w(dp_i, cm_k)}{2^{(w-1)}}
\]

where \(G_w(dp_i, cm_k)\) is the goodness of \(cm_k\) computed for window \(w\) on defense point \(dp_i\), which is presented as follows:

\[
G_w(dp_i, cm_k) = \frac{s(dp_i, cm_k)}{s(dp_i, cm_k) + f(dp_i, cm_k)}
\]

where \(s\) and \(f\) are the number of successes and failures evaluated in window \(w\). If there is no history for a given countermeasure in a window, we ignore that window in evaluation. The initial value of success and failure of all countermeasures at starting time (first window) are set to 1 and 0, respectively. In this case, the goodness is equal to 1, which represents a middle goodness value. This value can be updated according to the effectiveness of countermeasures to mitigate attacks over time.

Now, using the countermeasure vulnerabilities surface coverage and goodness features, we show how to evaluate the security performance of countermeasure deployment. Let \(cm_k\) be a countermeasure to be deployed on defense point \(dp_i\), where a set of countermeasures \(CM_i\) are still active. The actual security performance \(S_P\), after activating \(cm_k\) is given by:

\[
S_P(dp_i, cm_k) = f_z(cm_k) + \sum_{cm_i \in CM_i} f_s(cm_i, cm_k)
\]

where \(f_z(cm_k)\) and \(f_s(cm_i, cm_k)\) are two functions that compute the security performance of \(cm_k\) for the disjoint surface and joint surface(s), respectively. They are given by:

\[
f_s(cm_k) = |V_i| \cdot G(dp_i, cm_k)
\]

\[
f_s(cm_i, cm_k) = \begin{cases} 
|V_{k,i}| \cdot (G(dp_i, cm_k) - G(dp_i, cm_i)) & \text{if } G(dp_i, cm_k) > G(dp_i, cm_i) \\
0 & \text{if } G(dp_i, cm_k) \leq G(dp_i, cm_i)
\end{cases}
\]

where, \(G(dp_i, cm_k)\) and \(G(dp_i, cm_i)\) are the goodness of \(cm_k\) and \(cm_i\), respectively. Let \(V_k\) and \(V_i\) be the set of vulnerabilities covered by \(cm_k\) and \(cm_i\), respectively. \(V_{k,i} = V_k \cap V_i\) is a set of vulnerabilities that belong to the joint surface of \(cm_k\) and \(cm_i\). \(V_i = V_k - V_{k,i}\) is a set of vulnerabilities that belong to the disjoint surface of \(cm_k\). \(|V_k, i|\) and \(|V_i|\) are the cardinalities of \(V_{k,i}\) and \(V_i\), respectively.

5.5.2 Security Impact on Service Quality Evaluation

To evaluate the impact of the countermeasure deployment on service quality, we use the directed service dependency graph (SDG) (defined in Section 5.1). In such a graph, for each service \(s_j\), we have two sub-graphs: forward and backward. The former comprises the set of services on which \(s_j\) depends, and the latter comprises the set of services that depend on \(s_j\). Using that...
dependency graph, the impact of activating a countermeasure \( cm_k \) on service quality on the defense point \( dp_i \) is given by:

\[
S_Q(dp_i, cm_k) = \sum_{s_j \in X^k_i} C_p(cm_k) C_t(cm_k) \sum_{s_j \in X^k_i} (U(s_j))^\alpha (I(s_j))^\beta + \tilde{S}_Q(X^k_i, cm_k)
\]

where \( X^k_i \) represents the set of services affected by \( cm_k \) from those covered by the defense point \( dp_i \). \( \tilde{S}_Q \) is the impact on service quality for the directly affected services \((X^k_i)\). \( \tilde{S}_Q \) and \( \overrightarrow{S}_Q \) are the impact on service quality for the backward (\( B(s_j) \)) and forward (\( F(s_j) \)) sub-graphs of each service \( s_j \in X^k_i \), respectively (see Section 5.3). We evaluate them as follows:

\[
\overrightarrow{S}_Q(X^k_i, cm_k) = C_p(cm_k) C_t(cm_k) \sum_{s_j \in X^k_i} \sum_{s_l \in B(s_j) \land s_l \notin X^k_i} d_{l,i+1} (U(s_l))^\alpha (I(s_l))^\beta
\]

\[
\overleftarrow{S}_Q(X^k_i, cm_k) = 0
\]

where \( U(s_j) \), \( I(s_j) \), and \( d_{l,i+1} \) are, respectively, the number of users of \( s_l \), its importance, and its dependency severity with its child \( s_{l+1} \). For those services that are not directly accessible by users (e.g., DB service), only the administrator is considered the direct user. The values of \( \alpha \) and \( \beta \) express the relative importance between the service and the number of users using it from the point of view of the company. Therefore, to express the relative importance between them, we have \( \alpha + \beta = 1 \). \( C_p(cm_k) \) and \( C_t(cm_k) \) are the power and type of \( cm_k \) to stop attack. Since countermeasures impact only on service availability, we assume:

\[
C_t(cm_k) = [0 \ 0 \ 1]^T
\]

The countermeasure can be deployed either on a service (e.g., restarting a web service) or on a defense point (e.g., a firewall rule). In both cases, one or more service(s) may be affected. Thus, the backward impact propagation of a countermeasure is defined as the impact on all the services that have a mandatory dependency on the affected service(s), directly or indirectly. It is clear that there is no forward impact propagation.

### 5.5.3 Security Cost Evaluation

We propose to evaluate the security cost by two criteria, incompatibility cost \( IN_C \) and administration cost \( AD_C \). \( IN_C \) refers to rule insertion in a firewall while \( AD_C \) expresses the human intervention after a countermeasure deployment. Thus, we define these two criteria as follows:

**Definition 3:** The incompatibility cost represents the complexity of adding a security policy in security appliances (e.g., adding a rule in firewall).

**Definition 4:** The administration cost represents the time required to return the affected configurations, caused by deployed countermeasures, to the previous states.

Note that \( AD_C \) can be broken down into installation cost and operation cost. For example, disconnecting a machine incurs operation cost, whereas patching a vulnerability causes installation cost. However, we group them as one element, namely, administration cost. Thus, we have:

\[
S_C(dp_i, cm_k) = IN_C(dp_i, cm_k)^\gamma \cdot AD_C(cm_k)^\delta
\]

where the values of \( \gamma \) and \( \delta \) express the relative importance between the incompatibility and administration costs, respectively.

Now, we formulate the incompatibility cost of rule insertion in firewall. To avoid creating anomalies in the firewall, inserting a rule requires an analysis of all the existing rules in order to determine its proper order, and committing the updates [18]. As the number of rules increases, the insertion processing time increases. We modeled the incompatibility cost of rule insertion by a modified Smf, sigmoidal membership function, in range [1,3]. The original Smf function was in range [0,1]. We modified it for two reasons: 1) because of the product operation in (28), an incompatibility cost value of zero reduces the product to zero, which is not reasonable. That is, if there is no incompatibility cost, the security cost should depend on the administration cost. Therefore, we need to change the range to exclude the zero value; and 2) since the administration cost is in range [1,3], we modified the incompatibility cost range to bring the two functions within the same range. The incompatibility cost function is evaluated as follows:

\[
IN_C(dp_i, cm_k) = \text{Smf} \left( \frac{r(dp_i)}{a} \right)
\]

with:

\[
\text{Smf} \left( \frac{x}{a} \right) = \begin{cases} 
1 & \text{if } x \leq a \\
1 + 4 \left( \frac{x-a}{b-a} \right)^2 & \text{if } a \leq x \leq \frac{a+b}{2} \\
3 - 4 \left( \frac{x-b}{b-a} \right)^2 & \text{if } \frac{a+b}{2} \leq x \leq b \\
3 & \text{if } x \geq b 
\end{cases}
\]

where \( r(dp_i) \) represents the number of firewall rules on defense point \( dp_i \). The value of \( a \) expresses the number of rules that do not create any incompatibility cost \( (IN_C = 1) \). Incompatibility cost is saturated when the total rules reach the boundary \( b \) \( (IN_C = 3) \). These values can be determined by experience or defined by the security expert [18].

When the countermeasure is not a firewall rule, such as a patch, the incompatibility cost value is 1 (the lowest value, since adding a patch has no impact on incompatibility).

The administration cost can be any score between 1 and 3. In our model, the administration cost of a patch has the highest value. Restarting a firewall or shutting down/restarting a machine has a medium value cost. Those that do not need the administrator’s intervention (e.g., adding a rule in firewall) have the lowest value.

### 5.6 Pareto Optimal Set and Optimal CM Extraction

In previous sections, we explained how to evaluate the security performance, security impact on QoS, security cost, and attack damage cost. Now, we can solve equation (1), by evaluating the Pareto optimal set, from which we extract the optimal countermeasure, as explained in Section 4.

5.7 Discard Useless Optimal CMs

Now, for each attack path we have selected an optimal countermeasure to secure that path. We may have useless countermeasures such that their effect can be covered by others either on the same defense point or different ones. In the case more than one countermeasure is selected on the same defense point; we select the strongest one according to a pre-ordered countermeasure list. For instance, shutting down a service is stronger than restarting that service. For the second case, let us have two attack paths comprising these defense points sequences: \( \{dp_1, dp_3\} \) and \( \{dp_1, dp_2, dp_3\} \). Imagine that the target service is protected by \( dp_3 \). Assume that the optimal countermeasure for the first attack path is shutting down service \( s_x \), protected by \( dp_1 \), and the optimal countermeasure for the second attack path is restarting the target service. As can be seen, the second one is useless if we shut down the service \( s_x \). To do so, for two countermeasures each of whose related defense points are located in the same path, we keep the first visited one in the path if it is stronger than the second one; otherwise, we keep both. After discarding the useless optimal countermeasures, the remaining ones are deployed on related defense points.

6 Experimental Results

6.1 Simulation Setup and Integration in Cloud

To show the feasibility of our approach and test it in a real cloud, we integrated our framework in Openstack. To do so, a Cloud Intrusion Countermeasure Selection (CICS) was designed to interact with openstack control services. To realize the service chaining and traffic steering in Openstack, we used OpenDaylight. To validate the proposed framework, we use a cloud network topology consisting of three servers connected to the Internet through a physical openFlow-capable switch, as seen in Figure 4.

The first cloud server hosts two VMs, used as two web servers (WS1 : HR and WS2 : Portal), that are connected to a virtual switch (OVS1). The second cloud server hosts two VMs, the first one hosts a web server (WS3 : Finance); the second one hosts a mail server (MS1 : Mail). Similarly, these VMs are connected to a virtual switch OVS2. The third cloud server hosts two VMs, used as a database server (DBS1 : DB) and a file server (FS1 : File) that are connected to a virtual switch OVS3. Each VM possesses a set of vulnerabilities, represented in Table 1.

6.2 Attack Scenario

Figure 5 shows the ADT for our virtual network topology. As seen, there are many attack paths to compromise the database system. We programmed the Snort IDS to generate alerts with CVE id. Then, the CICS framework can map the CVE alert to the ADT. The attacker in the first step exploits the vulnerability \( v_1 \), CVE-2007-1741, in Apache 2.2.3, which is running on WS1. After exploiting the first vulnerability, the attacker uses the connection to the Database Server to compromise it. When the vulnerability \( v_1 \) is compromised, two attack paths are identified to compromise the goal node. As explained in Section 5.4, we eliminate one of the attack paths, since they visit the same sequence of defense points. As Figure 5 shows, there are five defense points in the attack path to stop the attacker. Table 2 gives a list of possible countermeasures at the five defense points. In the next subsections, we show the results of the security performance, security impact on QoS, and security cost evaluations for each candidate countermeasure. Finally, the optimal countermeasure is selected by evaluating the Pareto optimal set with respect to the attack damage cost.

6.3 Security Performance Evaluation

Figures 6a and 6b represent the history of successes and failures of all the countermeasures over time. To be fair, we wanted to have a large history for each countermeasure. During six months, we generated 5,691 attacks to compromise 15 vulnerabilities presented in the ADT. Each time, our IRS selected a countermeasure, deployed it, evaluated its success, and stored the result in the database. The efficiency time \( (\Delta t) \) for evaluating the success and failure of a countermeasure is equal to 5 seconds. If we show the results for the first attack, without history, all the countermeasures have the same goodness, which is not a case in practice.

Based on our ADT model, each type of countermeasure can be applied on different defense points to stop an attack. We never use the same history of goodness for the same type of countermeasure on different defense points. For example, countermeasures \( cm_1 \), \( cm_8 \), and \( cm_9 \) are of a kind that blocks attacker IP, but they have different success and failure rates. 84 successes on defense point \( dp_1 \) means that the attacker did not change his IP address and he did not try or did not know the different paths to the root of tree. In contrast, 46 failures on defense point \( dp_1 \) refer to the fact that either the attacker changed his IP address and
in two services: the portal and finance web applications on cloud servers 1 and 2, respectively. Countermeasure $cm_y$ has been applied on $dp_6$ ($vFW$ in cloud server 2), to block an attempt to compromise the finance web application exploiting vulnerability $v_8$. Since the goodness of $cm_y$ ($0.969$) is greater than that of $cm_x$ ($0.774$), $v_7$ is considered protected by $cm_y$.

As seen in Figure 5, five defense points exist in the attack path. All possible countermeasures presented in Table 2 are inserted individually in our framework, which takes into account all already deployed active countermeasures (as shown in Figure 7). A countermeasure vulnerabilities surface coverage may have joint surfaces with the two already deployed active countermeasures: $cm_x$ and $cm_y$. For example, when $cm_3 = \text{BlockPort}(FW,r,http)$ is applied on $dp_1$ the total vulnerabilities surface coverage is 13. The joint surface of $cm_3$ with $cm_x$ (filtering shellshock) consists of two vulnerabilities ($v_2$ and $v_7$) and with $cm_y$ consists of three vulnerabilities ($v_7$, $v_8$, and $v_9$). When the blocking port countermeasure is applied on another defense point, its surface coverage and joint surface change. For example, if we apply $cm_{11} = \text{BlockPort}(vFW,r,http)$ on $dp_2$ the total vulnerabilities surface coverage is 6 and has (1) $v_2$ and 0 vulnerability in the joint surface coverage with $cm_x$ and $cm_y$, respectively.

Figure 6d shows the number of vulnerabilities surface coverage in disjoint and joint surfaces with countermeasures $cm_x$ and $cm_y$. As can be seen, the countermeasures on defense point $dp_1$ have the larger joint surface with $cm_y$ and $cm_y$, while the countermeasures on defense point $dp_4$ and $dp_5$ do not have any joint surface with the currently deployed active countermeasures. Figure 6e shows the partial surface coverage for each countermeasure when it is applied in the framework. The security performance
for disjoint surface is the number of covered vulnerabilities times the countermeasure goodness, see (23). In contrast, the security performance for joint surface with already deployed active countermeasures depends on their goodness. If a new countermeasure has less goodness compared to the one that has joint surface with it, it cannot increase the security performance and this surface is ignored, see (24). Thus, if a countermeasure has small disjoint surface coverage and small goodness, it cannot increase the total security performance. Consequently, it has little chance of being selected unless it occupies a big role in the other two objectives. Figure 6f represents the final result of the first objective for 59 countermeasures. We observe that cm9, which is blocking an http request on dp1, has the best Sp. In contrast, cm34, which is patching the vulnerability, has the worst Sp, while the security performance of cm9 and cm27 are very close to zero.

6.4 Security Impact on Service Quality Evaluation

In this subsection, we evaluate the results of our second objective, which is the negative impact of countermeasures on service quality. Figure 8a illustrates the power of each countermeasure, which represents its ability to deal with an attacker, and it is bounded between 1 and 3. As can be seen in Figure 8a, blocking sender/receiver ip/port (e.g., cm3) are strong countermeasures while restarting VM or FW/vFW (e.g., cm7) are weak countermeasures to stop the progress of an attacker. As explained in subsection 5.5.2, to evaluate the negative impact of a countermeasure, two types of impacts are considered: direct and backward. Figure 10 shows the service dependency graph setup in the defined virtual network topology and the severity of dependency between services in terms of CIA. Because of the lack of such data in the literature, we have set reasonable values, as presented in the figure.

To calculate any type of impact, we need to know the number of online users for each service. The number of online users at attack time for HR, Portal, Finance, Mail, FTP, and DB are 20, 50, 0, 300, 1, and 1, respectively. Since the finance web application has been blocked by countermeasure cm9, no user can connect to this service. Since DB and FTP services cannot be accessed by public users, one user who is the administrator is considered for them. Based on (26), α and β (which express the relative importance between users and services) are initialized to 0.4 and 0.6. Figure 8b shows the result of direct and backward impact evaluation.

As can be seen in Figure 8b, the highest direct impact belongs to countermeasures cm6 and cm7 that blocks all traffic and restarts the firewall, and so no online users can connect to the services. Countermeasures cm1 to cm34, which are deployed on defense points dp1, dp2, or dp3, do not have any backward impact. In contrast, countermeasures cm35 to cm59, which are deployed on either defense point dp1 or dp5, have backward impact on DB.

Figure 8c shows the result of the second objective function, which is evaluating the countermeasures’ impact on QoS. As can be seen, the highest impact is related to cm6, which blocks all traffic on dp1. This countermeasure blocks all online users and has a big impact on QoS. In the second place, we see countermeasures cm40 = BlockAllTraffic(vFW), cm43 = Disconnect(CloudServer3), and cm45 = Shutdown(CloudServer3) on dp1. Each of these countermeasures disconnects the relationship between services and DB service. In contrast, four countermeasures have the lowest impact on QoS (2.62): cm1, cm9, cm50, and cm23. They block attacker IP on dp1, dp2, and dp3 or block attacker IP-http port on dp3. The impact of these types of countermeasures is on one user (the attacker), and does not have any impact on other users. The next impact at the next level (2.9) belongs to cm27 = CloseConnection(http) on defense point dp3, which is a WAF firewall close to HR machine and has merely a transient impact only on HR online users.

6.5 Security Deployment Cost Evaluation

Figure 9 shows the results of security deployment cost, which is based on incompatibility and administration costs. We define the countermeasure incompatibility cost through the number of rules in each firewall, which are 150, 85, 55, 75, and 45 on dp1, dp2, dp3, dp4, and dp5, respectively. The boundaries of the Smf function in (29) are a = 20 and b = 200. These boundaries are defined for the worst case scenario when the new rule has to be compared with the entire set of rules [18]. As can be seen in Figure 9a, since there are more rules in the first firewall on dp1, there is more incompatibility cost for cm1 to cm8.

Figure 9b illustrates the administration cost when a countermeasure is deployed. As revealed, the insertion of all firewall rules does not have any administration cost (equal to 1). The highest value belongs to patching a vulnerability, which is equal to 3. After that, the highest value is related to shutting down or restarting a server or virtual machine on which an administrator needs to do some operations to return the system to its previous configuration.

Figure 9c illustrates the final result for the third objective function. Based on (28), γ and δ, which express the relative importance between the incompatibility and administration costs, are initialized to 0.7 and 0.3, respectively. If the system administrator does not have time to do the configuration after each deployment, he can set it to 0.9 and 0.1. In contrast, when the number of rules increases, the performance of the firewall degrades. In this case, γ and δ can be set to 0.1 and 0.9, respectively. As shown in Figure 9c, inserting firewall rules on defense point dp1 results in the highest security cost compared to other countermeasures.

6.6 Optimal Countermeasure Selection

Now, let us solve (1) to evaluate the different trade-offs between Sp, SQ, and SC and identify the set of best solutions as well as the optimal one. The obtained results are presented in Figure 11, which shows all the solutions and the Pareto optimal set. Each defense point’s countermeasures are shown with

---

TABLE 3: Pareto optimal solutions

<table>
<thead>
<tr>
<th>defense point</th>
<th>CMs Description</th>
<th>Sp</th>
<th>Sq</th>
<th>Sc</th>
</tr>
</thead>
<tbody>
<tr>
<td>dp1</td>
<td>cm13 BlockIPPort(FW,http)</td>
<td>11.03</td>
<td>16.98</td>
<td>2.00</td>
</tr>
<tr>
<td></td>
<td>cm14 BlockIPPort(FW,http)</td>
<td>9.44</td>
<td>4.35</td>
<td>2.00</td>
</tr>
<tr>
<td></td>
<td>cm17 Restart(FW)</td>
<td>10.00</td>
<td>20.83</td>
<td>1.13</td>
</tr>
<tr>
<td>dp2</td>
<td>cm11 BlockIPPort(FW,http)</td>
<td>8.40</td>
<td>16.98</td>
<td>1.34</td>
</tr>
<tr>
<td></td>
<td>cm12 BlockIPPort(FW,http)</td>
<td>7.53</td>
<td>4.35</td>
<td>1.34</td>
</tr>
<tr>
<td></td>
<td>cm13 CloseConnection(http)</td>
<td>3.93</td>
<td>5.66</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>cm14 Restart(CloudServer1)</td>
<td>7.69</td>
<td>11.32</td>
<td>1.23</td>
</tr>
<tr>
<td>dp3</td>
<td>cm21 BlockIPPort(FW,http)</td>
<td>1.56</td>
<td>2.62</td>
<td>1.10</td>
</tr>
<tr>
<td></td>
<td>cm24 BlockIPPort(FW,http)</td>
<td>6.42</td>
<td>8.70</td>
<td>1.10</td>
</tr>
<tr>
<td></td>
<td>cm27 CloseConnection(http)</td>
<td>0.004</td>
<td>2.90</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>cm29 Restart(WS1)</td>
<td>4.27</td>
<td>5.80</td>
<td>1.23</td>
</tr>
<tr>
<td>dp5</td>
<td>cm47 BlockIPPort(FW,http)</td>
<td>3.94</td>
<td>26.06</td>
<td>1.05</td>
</tr>
</tbody>
</table>
different colors and shapes. The optimal solutions in Figure 11 are summarized in Table 3. The returned optimal countermeasures (those lying on the Pareto optimal surface) belong to all defense points except $dp_4$. If we focus on security performance, $cm_{37} = \text{CloseConnection}(\text{http})$ represents the best countermeasure with the highest $S_P$ value (11.03). However, it generates a high impact on service quality ($S_Q = 16.98$). The worst in security performance from the Pareto optimal set is $cm_{37} = \text{CloseConnection}(\text{http})$, which has lower vulnerabilities surface coverage (see Figure 6d) and goodness (close to zero, see Figure 6c). However, it provides less impact on service quality (2.90) since it closes only the HR application users’ connections.

We observe that some countermeasures are selected from different defense points because they yield different trade-offs between the objective functions: 1) $\text{BlockPort}(\text{FW}, \text{r, http})$, 2) $\text{CloseConnection}(\text{http})$, and 3) $\text{BlockIPPort}(\text{FW or vFW}, \text{s, attacker: http})$. For instance, blocking the attacker’s IP and http port has been selected on three defense points with different security performance (9.44, 7.53, and 1.56), security impact (4.35, 4.35, and 2.62), and security cost (2.00, 1.34, and 1.10). If we ignore dimension $S_C$, $cm_{12}$ should likewise be ignored because of less security performance compared to $cm_{4}$.

Furthermore, as shown in Figure 11, the majority of countermeasures on $dp_4$ and $dp_5$ (except $cm_{47}$) are dominated (in the sense of multi-objective optimization) and far from the Pareto optimal surface. Since these countermeasures, which are applied on cloud server 3, are close to the target (DBS1), they provide less vulnerabilities surface coverage, and thus less $S_P$. Besides, they have a high direct impact on DBS1 and backward impact on four services that have direct dependency on DBS1 (see Figure 10). That is why these countermeasures are not selected and are far from the Pareto optimal surface (low $S_P$ with high $S_Q$).

Now, we select the optimal solution from the Pareto optimal set using SAW method, with respect to the attack damage cost. Figure 12 represents the result of the attack damage cost evaluation when vulnerability CVE-2007-1741 is exploited in service HR. As explained, the damage cost consists of current and potential damages which are 2.4 and 7.45 in terms of CIA, respectively. So, the total damage cost is 9.85. Since, no service uses HR, the current backward damage from HR is zero, while the backward damage from DB is high. As shown in Figure 12, the total value of services in our framework is 14.1, in terms of CIA. Therefore, the total damage cost for the whole system from the exploited vulnerability is 69%. For this attack damage cost value, $cm_{3}$ is selected as the optimal one.
TABLE 4: The first three optimal countermeasures selected by SAW, PW, and TOPSIS with respect to attack damage cost variation ($w_p$). Based on (9), the other two weights, $w_Q$ and $w_C$, are obtained using $w_p$ and $\sigma = 0.6$ and $\theta = 0.4$.

<table>
<thead>
<tr>
<th>Alternative</th>
<th>$w_p$</th>
<th>0.1</th>
<th>0.3</th>
<th>0.5</th>
<th>0.7</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAW</td>
<td>1$^\text{st}$ optimal cm</td>
<td>cm123</td>
<td>cm124</td>
<td>cm122</td>
<td>cm121</td>
<td>cm123</td>
</tr>
<tr>
<td></td>
<td>2$^\text{nd}$ optimal cm</td>
<td>cm127</td>
<td>cm128</td>
<td>cm129</td>
<td>cm121</td>
<td>cm123</td>
</tr>
<tr>
<td></td>
<td>3$^\text{rd}$ optimal cm</td>
<td>cm122</td>
<td>cm123</td>
<td>cm124</td>
<td>cm129</td>
<td>cm127</td>
</tr>
<tr>
<td>PW</td>
<td>1$^\text{st}$ optimal cm</td>
<td>cm123</td>
<td>cm124</td>
<td>cm122</td>
<td>cm121</td>
<td>cm123</td>
</tr>
<tr>
<td></td>
<td>2$^\text{nd}$ optimal cm</td>
<td>cm127</td>
<td>cm128</td>
<td>cm129</td>
<td>cm121</td>
<td>cm123</td>
</tr>
<tr>
<td></td>
<td>3$^\text{rd}$ optimal cm</td>
<td>cm122</td>
<td>cm123</td>
<td>cm124</td>
<td>cm129</td>
<td>cm127</td>
</tr>
<tr>
<td>TOPSIS</td>
<td>1$^\text{st}$ optimal cm</td>
<td>cm123</td>
<td>cm124</td>
<td>cm122</td>
<td>cm121</td>
<td>cm123</td>
</tr>
<tr>
<td></td>
<td>2$^\text{nd}$ optimal cm</td>
<td>cm127</td>
<td>cm128</td>
<td>cm129</td>
<td>cm121</td>
<td>cm123</td>
</tr>
<tr>
<td></td>
<td>3$^\text{rd}$ optimal cm</td>
<td>cm122</td>
<td>cm123</td>
<td>cm124</td>
<td>cm129</td>
<td>cm127</td>
</tr>
</tbody>
</table>

6.6.1 Reliability of the Obtained Optimal Result

In this section, we show that the SAW method used provides reasonable results, compared to other well-known scoring methods: PW and TOPSIS. Besides SAW, we also implemented PW and TOPSIS. Table 4 illustrates the three optimal solutions from the Pareto optimal set resulted by the three methods, through the severity of attack damage cost varies between 0.1 to 0.9. We observe that: (1) when $w_p$ is low, 0.1, SAW and PW select reasonable countermeasures compared to that selected by TOPSIS ($cm_{123}$ vs. $cm_{112}$). That is, $cm_{112}$ provides higher $S_P$ than $cm_{123}$, and that incurs higher impact on $S_Q$, see Table 3. $cm_{112}$ blocks two VMs in cloud server 1, while $cm_{123}$ only blocks one VM (HR). Such high impact on $S_Q$ is not needed as $w_p$ is low, (2) in case of high $w_p$, 0.9, the countermeasure selected by TOPSIS, $cm_{147}$, provides low $S_P$ (3.94) compared that selected by SAW and PW, which is $cm_{13}$ (11.03). Moreover, $cm_{147}$ incurs higher impact on $S_Q$ (26.06) compared to that selected by SAW and PW (16.98). From the security perspective, $cm_{147}$ provides less protection (only DB in the network), and high impact on service quality (all VMs: HR, Portal, Finance, and Mail, see Figure 10). On the other hand, $cm_{13}$ provides higher protection (all http servers), and less impact on service quality compared to $cm_{147}$ (no impact on Mail server), and (3) when $w_p$ is equal to 0.3 and 0.7, SAW performs better than PW. That is, when $w_p = 0.3$, which is somehow low, SAW selected less strong countermeasure than that selected by PW. On the opposite, when $w_p = 0.7$, which is somehow high, SAW selects stronger countermeasure than that selected by PW. On a final note, we conclude that SAW performs better all the time.

Now let us have a deep insight on the optimal countermeasures selected by SAW. If we consider the first optimal solutions for different weight combinations, we can see that when the attack is weak ($w_p < 0.5$), the block attacker IPPort(http) is selected ($cm_{23}$ on $dp_1$). When the severity of damage is a middle value, the same countermeasure is selected, but on different places with more severe effects ($cm_4$ on $dp_1$). It blocks the attacker from accessing any http service. When the severity of the attack is high, $cm_3$ on $dp_1$ is selected to block all users’ access to any http service. Countermeasures $cm_4$, $cm_{12}$, and $cm_{23}$ are of a kind, but have different effects because of their deployment positions. As can be seen, $cm_4$, which is stronger than others, appears when $w_p \geq 0.5$. $cm_{27}$, which closes the http connections on $dp_3$, behaves similarly to $cm_{23}$.

6.7 Performance Evaluation

In this subsection, we present the asymptotic complexity and real-time performance of the CICS framework. As shown in Table 5, the complexity in the worst case is $O(|CM|^2 + |S| + |W| + |V| + |CM_a| + |CM_s|)$, where $|CM|$ is the number of possible countermeasures that can be deployed on all defense points, $|S|$ is the number of services in the SDG, $|W|$ is the number of windows in goodness evaluation, $|V|$ is the number of vulnerabilities, and $|CM_a|$ represents the number of active countermeasures. ADC evaluation depends on the importance of services, severity of dependency between them in terms of CIA, and the number of service vulnerabilities. Since all these parameters are static, the attack damage cost of each vulnerability can be evaluated and stored in the database in advance. Over time, when any change happens to these parameters, the ADC is updated. Note that $S_P$, $S_Q$, and $S_C$ are dynamic, but $S_P$ and $S_C$ can be evaluated between two attack times, whereas $S_Q$ should be evaluated at attack time. Let us consider two attack times: $t_1$ and $t_2$ ($t_1 < t_2$). When an optimal countermeasure is applied at $t_1$, we wait $\Delta t$, (which is efficiency time), ($t_1 + \Delta t < t_2$) to monitor the success and failure status of the applied countermeasure and update its $S_P$. Similarly, $S_C$ is updated only when a new firewall rule is added. (Note that only one type of countermeasure, which is rule insertion, can impact (29).) In contrast, because $S_Q$ depends on the number of online users, it should be updated at attack time. In this case, the complexity of our framework is $O(|CM|^2 + |S| + |CM|)$.

The performance of our framework has been evaluated only for the response system when the CICS framework receives the alerts from the IDS. We distinguish two times: the time needed to obtain the optimal countermeasure and the time to deploy it. Obtaining the optimal countermeasure, from a list of 59 countermeasures, takes $314\ ms$. The optimal countermeasure to stop our attack scenario is $cm_{35}$, which blocks the http port in the physical firewall on $dp_1$. Since this firewall has 150 rules, the insertion incompatibility process takes $135\ ms$. Our framework was prototyped on a machine with 6 cores Intel Xeon (XEON L5638) clocked at 2.30 GHz with 24 GB RAM. In total, the CICS framework takes about $449\ ms$ for our attack scenario. It is reasonably fast in deciding which countermeasure is the optimal one.

7 Conclusion

In this paper a dynamic framework for selecting and deploying an optimal countermeasure to mitigate attack in an online fashion was proposed. The advantage of the proposed framework is that the selected countermeasure can be fine-tuned on the basis of the context in which the attack is detected. In other words, the selection mechanism takes into account the trade-off between maximizing the security benefit, minimizing the security impact on users and services, and minimizing the security deployment cost. Thus, the countermeasures are not predictable since they are not statically determined. This reduces the possibility of attackers exploiting the predictability and the potential security impact caused by unnecessary countermeasures. Compared to existing work, all the proposed objective functions are dynamic, which reflect the dynamicity of the network status (appearing or
disappearing users, services, and vulnerabilities), the complexity of security appliances, and the goodness of countermeasures over time. Experimental results show the applicability and performance of the proposed framework in, but not limited to, the cloud.

For future work, it would be worthwhile to investigate supporting multiple ADTs at the same time. As explained in Section 5, different ADTs can be defined based on the SDG in such a way that each ADT protects one goal service. Our framework was designed for one ADT at the time.

ACKNOWLEDGMENT

This work is partly funded by Natural Sciences and Engineering Research Council of Canada Research Chair on Sustainable Smart Eco-Cloud, NSERC-950-229052 and by the NSERC CRD PD 424371-11: ECOLOTIC Sustainable and Green Telco-Cloud.

REFERENCES


Ali Reza Samadlou-Sendi is currently an Assistant Professor at Shahid Beheshti University. Before joining SBU, he was Postdoctoral Fellow at Ericsson, Canada and Postdoctoral at ETS and McGill universities in collaboration with Ericsson. He received his Ph.D degree in computer engineering from Ecole Polytechnique de Montreal, Canada. He obtained his B.Sc. and M.Sc. from Amirkabir University of Technology. His primary research interests include information security, intrusion response system, and cloud computing. He is a recipient of Postdoctoral Research Fellowship Award and Industrial Postdoctoral Fellowship Award from Canada.

Habib Louafi (S’11) holds an Engineering degree in Computer Science from the University of Oran (Algeria), an M.Sc. from the Université du Québec à Montréal (UQAM), and a Ph.D. from Ecole de technologie supérieure (ETS is a part of the Université du Québec network). He is currently working as a postdoctoral fellow at Synchromedia Laboratory for multimedia communication in telepresence (ETS). His fields of interest include mobile cloud computing, context-aware systems, multi-objective optimization, and information security.

Wenbo He received the PhD degree from University of Illinois at Urbana-Champaign, Champaign, IL, USA, in 2008. She is currently an Associate Professor in Department of Computing and Software at McMaster University. Before that she was an Assistant Professor in School of Computer Science at McGill University from 2011 to 2016, and was an Assistant Professor in Department of Electrical Engineering at University of Nebraska-Lincoln from 2010 to 2011. She was with the Department of Computer Science at University of New Mexico from 2008 to 2010. Her research focuses on Big Data Systems, Cloud Computing, and Privacy-preserving Techniques, etc.

Mohamed Cheriet received his M.Sc. and Ph.D. degrees in Computer Science from the University of Pierre et Marie Curie (Paris VI) in 1985 and 1988 respectively. Dr. Cheriet is expert in cloud computing and network virtualization. In addition, he is an expert in Computational Intelligence, Pattern Recognition, Mathematical Modeling for Image Processing, Cognitive and Machine Learning approaches and Perception. Dr. Cheriet has published more than 350 technical papers in the field. He holds Canada Research Chair Tier 1 on Sustainable Smart Eco-Cloud.