Hindawi
Security and Communication Networks
Volume 2020, Article ID 7235043, 15 pages
https://doi.org/10.1155/2020/7235043

*Research Article*

# A Framework for Real-Time Intrusion Response in Software Defined Networking Using Precomputed Graphical Security Models

**Taehoon Eom**,[1] **Jin B. Hong**,[2] **SeongMo An**,[1] **Jong Sou Park**,[1] **and Dong Seong Kim**[3]

[1]*Department of Computer Engineering, Korea Aerospace University, Goyang, Republic of Korea*
[2]*Department of Computer Science and Software Engineering, University of Western Australia, Perth, Australia*
[3]*School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane, Australia*

Correspondence should be addressed to Taehoon Eom; eomth86@gmail.com

Received 17 August 2019; Revised 28 December 2019; Accepted 14 January 2020; Published 18 February 2020

Academic Editor: Stelvio Cimato



Software defined networking (SDN) has been adopted in many application domains as it provides functionalities to dynamically control the network flow more robust and more economical compared to the traditional networks. In order to strengthen the security of the SDN against cyber attacks, many security solutions have been proposed. However, those solutions need to be compared in order to optimize the security of the SDN. To assess and evaluate the security of the SDN systematically, one can use graphical security models (e.g., attack graphs and attack trees). However, it is difficult to provide defense against an attack in real time due to their high computational complexity. In this paper, we propose a real-time intrusion response in SDN using precomputation to estimate the likelihood of future attack paths from an ongoing attack. We also take into account various SDN components to conduct a security assessment, which were not available when addressing only the components of an existing network. Our experimental analysis shows that we are able to estimate possible attack paths of an ongoing attack to mitigate it in real time, as well as showing the security metrics that depend on the flow table, including the SDN component. Hence, the proposed approach can be used to provide effective real-time mitigation solutions for securing SDN.

## 1. Introduction

One of the key functionalities of an SDN (software defined networking) is to allow network administrators to dynamically change the logical network topology in real time [1, 2]. This is achieved by separating the controls from the data flows onto the control plane and data plane, respectively. Moreover, network disruptions have a minimum impact on the performance when the SDN dynamically reconfigures the network topology [3], which allows the administrators to optimize the load more efficiently in real time. Moreover, these SDN functionalities allow for new security mechanisms to be designed and deployed, such as moving target defense (MTD) systems, which require continuous changes made to the network [4, 5]. However, the SDN also introduces new networking components (e.g., such as the SDN controllers and forwarding devices), which opened new attack vectors for attackers to exploit the SDN [6].

There are various techniques developed to protect the SDN from attacks, such as DELTA [7], a security evaluation framework, athena [8], an anomaly detection development framework, and research to predict attack patterns using machine learning [9]. However, one must assess their effectiveness in order to optimize the security of the SDN. An approach is to use graphical security models (such as attack graphs (AG) and attack trees (AT)) to evaluate the security of the SDN [10–13], which provide an in-depth analysis of the security (e.g., various attack scenarios and multihop attack paths), as well as means to compute optimal countermeasures [14]. Applying this approach to a SDN

environment also requires considering the new SDN components into the security assessment, which were not previously captured and analyzed. In addition, intrusion detection systems may not always detect ongoing attacks in real time, causing a delay between the initiation of an attack to response. Hence, the attacker could already be in the reach of the target, so countermeasure efforts should be more focused on deterring the attacker from reaching the target, rather than hardening the point of detection. To do this, we must generate and analyze all possible attack paths, which can be used to understand possible targets the attacker is trying to compromise. However, computing all possible attack paths suffers from scalability and adaptability problems [15, 16]. Therefore, we need a more efficient technique that can evaluate all possible attack paths more efficiently while taking into account the new SDN components in the security assessment.

To address the aforementioned problems, we propose a precomputation approach with the SDN components incorporated into a graphical security model, namely, the hierarchical attack representation model (HARM) [17], to assess the security of the SDN in real time. The precomputed HARM allows us to evaluate all possible attack paths prior to an attack detected, which can be used to estimate possible attack paths from the point of detection to formulate effective countermeasures. In particular, we use a full AG [18–20] to generate the precomputed attack scenarios for the evaluation. And we used an attack scenario in which an attacker tried to break in to steal data from the outside. Once we identified these attack paths, we then take into account the case where the delay of the intrusion detection mechanism takes longer than the attack time. The precomputed full AG is used to identify relevant attack paths, which are then evaluated to deploy relevant countermeasures. We further conduct experimental analysis to demonstrate that our proposed approach can effectively trace an attack with delayed detections and mitigate an ongoing attack in real time. The contributions of our paper are summarized as follows:

(i) To conduct security assessment for the SDN that takes into account new SDN components and their associated attack vectors

(ii) To generate precomputed attack scenarios using a full AG for real-time security assessment and countermeasure in the SDN

(iii) To propose response and prevention for ongoing intrusions that take into account delays observed by attack detection mechanisms

(iv) To conduct experimental analysis to demonstrate the feasibility of the proposed approach for mitigating an ongoing attack with delayed detections in the SDN

The rest of the paper is organized as follows. Section 2 presents the overall framework and flowchart, and the details of how each module works are shown in Section 3. In Section 4, precomputation of attack scenarios and future attack scenario predictions are presented. The experimental analysis is presented in Section 5. The discussion and limitations of this paper are presented in Section 6, and Section 7 presents the related work. Finally, we conclude our paper in Section 8.

## 2. A Framework for Real-Time Intrusion Response in SDN

To overcome the limitations of IDSes, we propose a precomputed graphical security model (GSM) for real-time intrusion response in SDN. The general steps are described as follows: (1) collect configuration information of the SDN including security vulnerabilities and node connections/dependencies, (2) input the gathered information to generate the GSM for security assessments, (3) collect intrusion detection data from the SDN, and lastly (4) compute effective attack response by selecting optimal countermeasure. The relationships between these steps are shown in Figure 1, with the workflow of our framework presented in Figure 2.

*2.1. SDN Configuration.* To evaluate the security posture associated with the SDN, we first need to collect the required security information. Two main information are vulnerabilities associated with each SDN component and their connectivity/dependency. The vulnerability information can be gathered using various vulnerability scanning tools such as NESSUS [21] and OpenVAS [22]. The component dependencies can be gathered from the flow table and SDN controller settings. This information is then sent to the security modeling and analysis module.

In the SDN, there are also IDSes. Any detected intrusions are forwarded to the intrusion detection module directly. Note that IDSes are typically not placed on all SDN components (e.g., too costly), and therefore the attack scenario can be even more complex and unpredictable in SDNs with very sparse IDSes.

*2.2. Security Modeling and Analysis.* Using the inputs from the SDN configuration module, we then generate a GSM [10, 11]. For example, hierarchical attack representation model (HARM) is a scalable and adaptable GSM [17], which we will use in our paper (we have selected to use the HARM for demonstrations, but other GSMs can be used as well; however, the selection of an appropriate GSM to use is out of scope in this paper). GSMs can take into account various security vulnerabilities and compute various attack scenarios associated with different dependencies. However, they still suffer from the scalability problem when the size of the network gets larger. Hence, the need for a precomputation technique to achieve the real-time attack response. The precomputed security assessment information is then sent to the attack response module, which will be used when an intrusion is detected.

*2.3. Intrusion Detection.* The IDSes in the SDN collect the intrusion logs, which are sent to the attack response module. This module processes the raw intrusion detection data and

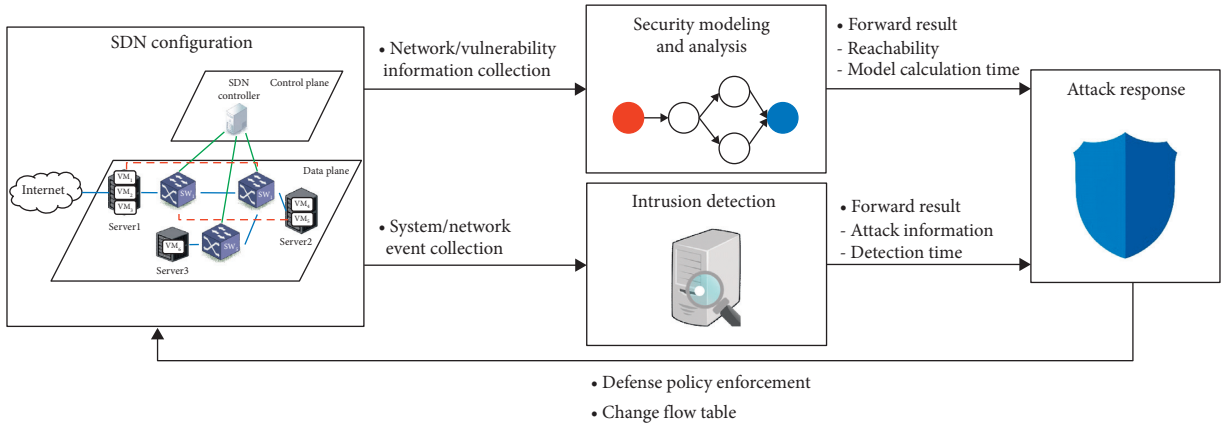


Figure 1: Framework for real-time intrusion response in SDN

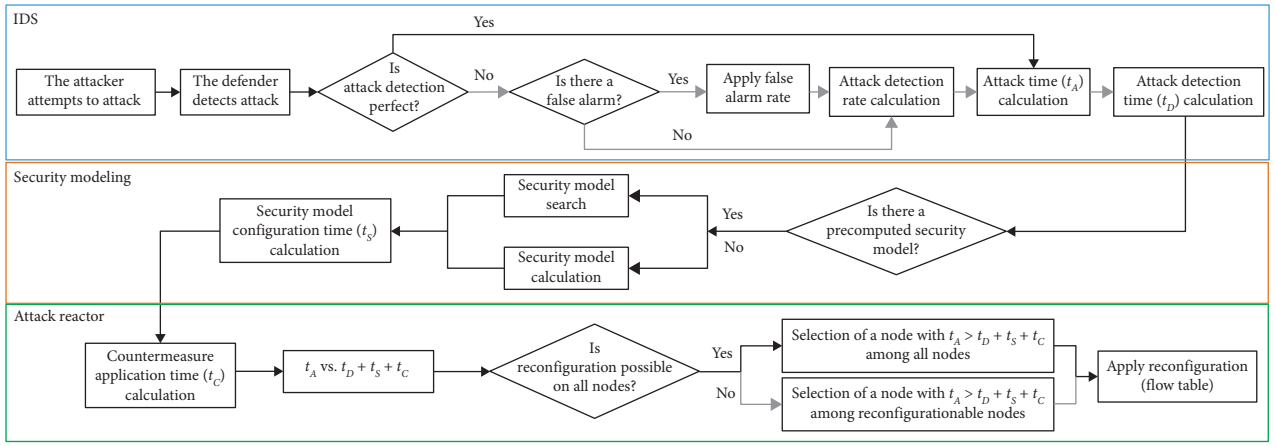


Figure 2: Overall system flowchart.

analyzes the attack information (e.g., type of attack) and its associated metadata (e.g., location, time, etc). Although this module will try its best to detect attacks accurately and fast, we cannot rely on its performance that it would be in real time and fully accurate.

*2.4. Attack Response.* The attack response module is one of our main contributions, where the impact of an attack is evaluated taking into account the intrusions detected and the location of the attack. For instance, if an attack is detected where the subnet contains many vulnerable computers, then the impact may be that one or more of computers may also be compromised soon after. Consequently, the goal of the attack response module is to reduce the impact of the attack by quickly locating, estimating the damage, and isolating the attack from its progression.

## 3. Real-Time Intrusion Response in SDN

*3.1. SDN Configuration.* To demonstrate the usability of our proposed solution, we take into account a running example as shown in Figure 3. The toy example includes nine nodes in the data plane (i.e., six virtual machines (VMs) and three switches). Table 1 shows the defined flow table in the SDN. We assume that only the VMs on web server are connected to the Internet, and the attacker is located outside the SDN (i.e., no attackers inside the SDN) (our proposed solution is also applicable for inside attackers as it takes into account both security models and IDSes; however, we initially focus on attackers outside the SDN first). The role of the VMs is to provide services within and to the external users (e.g., an enterprise network setup using the SDN). For example, a user requesting a service will access $VM_1$, $VM_2$, or $VM_3$ located in web server.

If there is no problem with the system, the system operates as follows. A user sends a request to the system, which requires the data stored in a database (e.g., $VM_6$). To establish this service, a VM in web server (e.g., $VM_1$) requests the data through a VM in app server (e.g., $VM_4$) for all valid requests. Then, this request gets passed to $VM_6$ for processing. Finally, the requested data get returned to the user through the VMs the request was processed from (e.g., in this instance, through $VM_1$, $VM_4$, and $VM_6$).

There are two redundancy connections between $VM_1$ and $SW_2$ and $VM_5$ and $SW_1$ to continue to provide functionalities in the event of an emergency (e.g., burst in requests or a DDoS attack). However, if the attacker compromises the SDN controller, these redundant connections can be used to form various attack paths. Based on

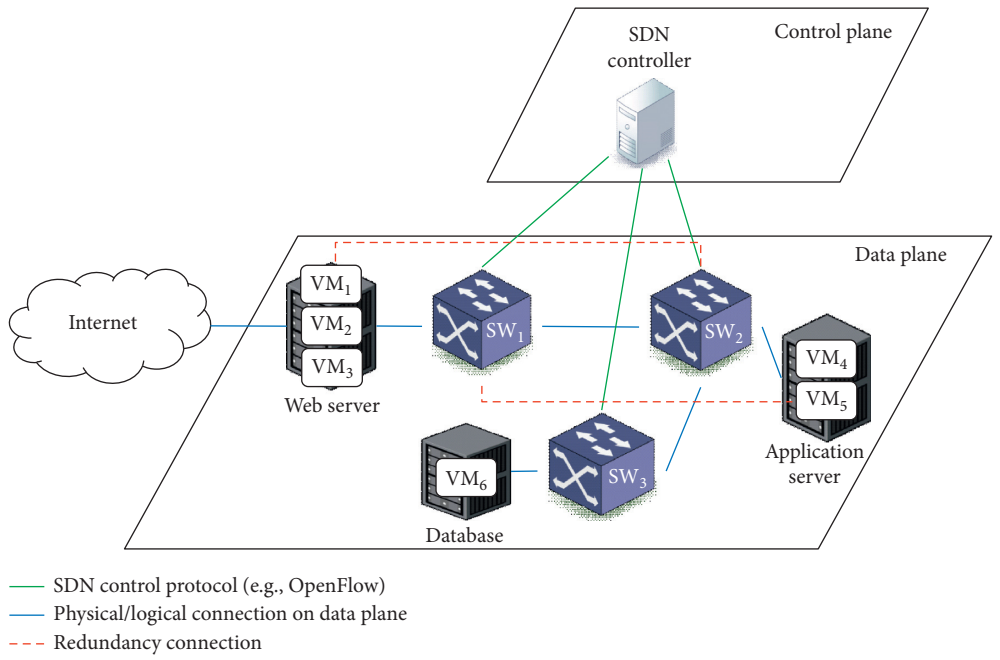


Figure 3: Example SDN configuration.

Table 1: The flow table.

| SW ID | Match fields | | | Action | Priority |
|---|---|---|---|---|---|
| | Port | Src | Dst | | |
| $SW_1$ | 1 | * | * | Forward port 2 | 1 |
| | 2 | * | $VM_4$ | Forward port 3 | 1 |
| | 2 | * | $VM_5$ | Forward port 3 | 1 |
| | 4 | * | $VM_5$ | Forward port 5 | 3 |
| | * | * | $VM_6$ | Drop | 9 |
| $SW_2$ | 1 | * | * | Forward port 2 | 1 |
| | 2 | * | $VM_6$ | Forward port 3 | 3 |
| | 4 | $VM_1$ | $VM_4$ | Forward port 2 | 3 |
| | 4 | $VM_1$ | $VM_5$ | Forward port 2 | 3 |
| | * | * | $VM_6$ | Drop | 2 |
| $SW_3$ | 1 | * | $VM_6$ | Forward port 2 | 1 |

Table 2: Operation system and vulnerabilities in each node.

| Node | OS | CVE ID | CVSS BS | Impact |
|---|---|---|---|---|
| $VM_1$ | Win 7 | CVE-2013-0013 | 5.8 | 4.9 |
| | | CVE-2012-0001 | 9.3 | 10 |
| $VM_2$ | Win 7 | CVE-2015-0006 | 6.1 | 4.7 |
| | | CVE-2015-1675 | 9.3 | 10 |
| $VM_3$ | Linux | CVE-2012-4546 | 4.3 | 2.9 |
| | | CVE-2014-0100 | 9.3 | 8.2 |
| $VM_4$ | Win 7 | CVE-2017-8495 | 6.0 | 4.8 |
| | | CVE-2017-8717 | 9.3 | 10 |
| $VM_5$ | Linux | CVE-2015-7312 | 4.4 | 2.9 |
| | | CVE-2015-4002 | 9.0 | 8.5 |
| $VM_6$ | Linux | CVE-2017-0626 | 4.3 | 2.9 |
| | | CVE-2017-6264 | 9.3 | 8.2 |
| $SW_1$ | Openflow 2.5 | CVE-2014-5035 | 6.8 | 6.4 |
| $SW_2$ | Openflow 2.7 | CVE-2017-9263 | 6.5 | 6.5 |
| $SW_3$ | Openflow 2.8 | CVE-2017-14970 | 5.9 | 6.8 |

the SDN configurations and settings, the operation system of each node and vulnerabilities can be found as shown in Table 2. For simplicity, we chose only a few vulnerabilities in the SDN for each node (OS vulnerabilities for VMs and OpenFlow vulnerability for SDN switches), but all vulnerabilities can be modeled as in [23].

### 3.2. Security Modeling and Analysis

*3.2.1. Common Vulnerability Scoring System (CVSS).* In order to measure the severity of vulnerabilities, we use the CVSS base score (BS) [24]. First, we mention a few key updates to the CVSS BS system. The base vector takes into consideration of the "User Interaction" and "Privileges Required," and "Physical Metric" has been added to the attack vector. Confidentiality, integrity, and availability measures are changed from {None, Partial, Complete} to {None, Low, High}, and "Access Complexity" has been changed to "Attack Complexity." The following equations are used to compute the CVSS BS metric, which we simply denoted as "BS" (as shown in equation (1)); "IM" represents the Impact Metric (as shown in equation (2)), and "$E$" represents the Exploitability Metric (as shown in equation (3)).

$$\mathrm{BS} = (0.6 \times \mathrm{IM} + 0.4 \times E - 1.5) \times f(\mathrm{IM}), \quad (1)$$

$$\mathrm{IM} = 10.41 \times (1 - (1 - C) \times (1 - I) \times (1 - A)), \quad (2)$$

$$E = 20 \times AC \times AU \times AV. \quad (3)$$

Based on risk computation in [5], we utilize the CVSS BS in order to compute the system risk, which is calculated as shown in equation (4) (i.e., the system risk is a factor of

impact and probability of an attack). In order to compute the system security risk, we need to know the probability of an attack success and the impact. Here, we use the exploitability metric associated with each vulnerability (as shown in Table 2) to represent the probability of an attack success as in equation (5) and use the impact metric directly from the CVSS.

$$\text{Risk}_{\text{Vul}} = \text{IM} \times P_{\text{attack}}, \tag{4}$$

$$P_{\text{attack}} = \frac{\text{BS}}{10}. \tag{5}$$

*3.2.2. Attack Graph for SDN.* Here, we describe the AG used to model SDN, which captures the sequence of vulnerabilities to be exploited to achieve the attack goal. We assume the attack goal is to execute arbitrary code on $\text{VM}_6$. First, we define an AG as follows.

*Definition 1.* An AG is a directed graph $\text{AG} = (V, E)$, where $V$ is a finite set of vulnerabilities in the networked system and $E \subseteq V \times V$ is a set of edges where a pair of vulnerabilities $(v_i, v_j) \mid v_i \in V, v_i \neq v_j$ is a mapping of nodes $v_i \longrightarrow v_j \; \forall \text{post}(v_i) = \text{pre}(v_j)$ such that the postcondition of $v_i$ satisfies the precondition of $v_j$.

Given the definition above, we can generate an AG to map attack scenarios of our example SDN as shown in Figure 4. Given the model and the system risk calculation steps above, we can compute the system risk associated with our example SDN. For instance, the attacker can exploit vulnerabilities $\text{WV}_1$ and $\text{WV}_2$ as specified in Table 2 for Windows 7-based VMs. If the attacker exploits $\text{WV}_1$ vulnerability, then $\text{Risk}_{\text{WV}_1}$ is 2.842 (i.e., the impact of 4.9 multiplied by the probability of 5.8). Similarly, $\text{Risk}_{\text{WV}_2} = 9.3$, $\text{Risk}_{\text{LV}_1} = 1.247$, $\text{Risk}_{\text{LV}_2} = 7.65$, and $\text{Risk}_{\text{OFV}_1} = 4.352$.

*3.3. Intrusion Detection.* In this section, we take into account the time factor when an attack has been detected. It is possible that an ongoing attack may have progressed further at the time of detection. Therefore, it is important to take into consideration which attack scenarios are important in order to mitigate the attack. Generally, attack detection should consider *Bayesian Theory*, but we assume the attack detection mechanisms in the SDN is correct (e.g., we can use detection mechanisms such as in [25–28]). If we consider Bayesian theory, attack detection is similar to applying the Threshold Random Walk with Credit-Based connection rate limiting (TRW-CB) algorithm in [28]. The detection rate is 92.54% and a false alarm rate is 7.48%.

Figure 5 shows the detection of an attack success at $\text{VM}_2$. Given the attacker has not yet progressed any further, the SDN administrator can deploy countermeasures. For example, we change the flow table rules to drop all outgoing packets of $\text{VM}_2$, disabling any further attacks. Figure 5(b) shows the result of the countermeasure.

However, if we assume that the detection of the attack has been delayed (i.e., the attack is detected after a $t$ amount of time has passed since the actual event of an attack), the attacker would consequently have progressed further from compromising $\text{VM}_2$ in our example. This is depicted in Figure 6(a). The attacker has successfully compromised $\text{SW}_1$ after compromising $\text{VM}_2$, but the attack detection only alerted the SDN administrator the progress of the attack at $\text{VM}_2$. In order to predict its current attack scenario, we use the full AG and focus on all possible attack paths from the given detection point as shown in Figure 6(c). Using the flow table rule change as the countermeasure, our approach is to limit the attack path up to $h$-hops, where $h$ is the number of hops from the node with initial attack detection. For example, if we use 2-hop path disable, then the result is shown as in Figure 6(b). As a result, we are able to disable further attack paths of the attacker in a trade-off to some loss of SDN functionalities. In conclusion, this is to show that we can still maintain some functionalities of the SDN while disabling any potential ongoing attacks. We investigate how security is affected further in Section 5.

*3.4. Attack Response.* SDN can manipulate the flow of data plane using flow table. Therefore, when an attack occurs in the SDN environment, it is possible to block the attack path by modifying the flow table in addition to the response method (e.g., patching a vulnerability) used in the existing network. However, if the response is delayed, the attacker may succeed in exploiting the next target before the defense is implemented. We considered system loss and cost of action based on the relationship between the attacker's attack time and the defender's response time. For that, we assume the following. First, the attack detection (IDS) is complete and all nodes can be monitored at the same time. Second, all of network flows can be changed using the flow table. Third, the devices or software that make up the SDN are not changed. The attack time and response time that we use follow the following definition.

*Definition 2.* Attack time $t_A$ is defined as the time taken for an attacker to succeed in attacking the next host connected at the current location.

*Definition 3.* Response time is defined as $t_R = t_D + t_S + t_C$. Here, $t_D$ is defined as the time taken to detect an attacker's attack attempt on the host (attack detection time). $t_S$ is defined as the time taken to calculate the security model in real time or to retrieve it from the precomputed security model (security model calculation time). $t_C$ is defined as the time required to apply a countermeasure to one host (countermeasure time).

Given the above definition, an example of comparisons between attack time and response time can be expressed as follows.

*Example 1.* Figure 5(b) shows $t_R = t_A$. The loss node is $\text{VM}_2$, and the action node is $\text{SW}_1$. Assuming that both the cost of damage from the attack and the cost of the action are 100, the total cost is 200.

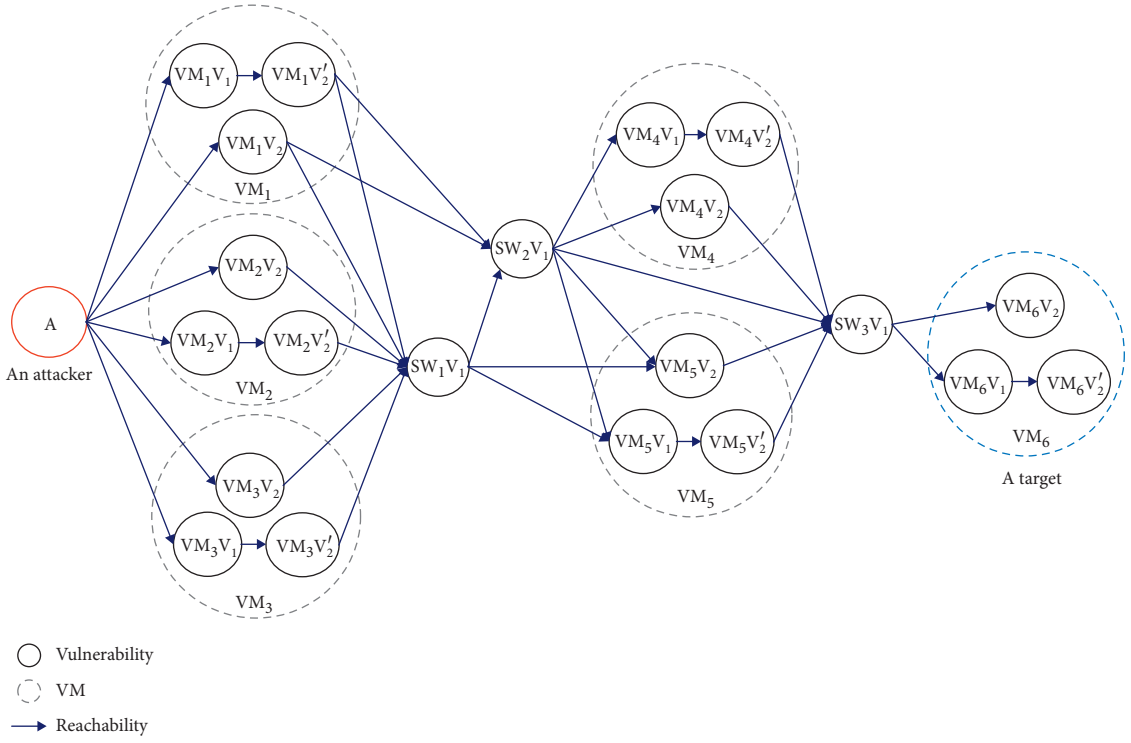


Figure 4: An AG of the SDN

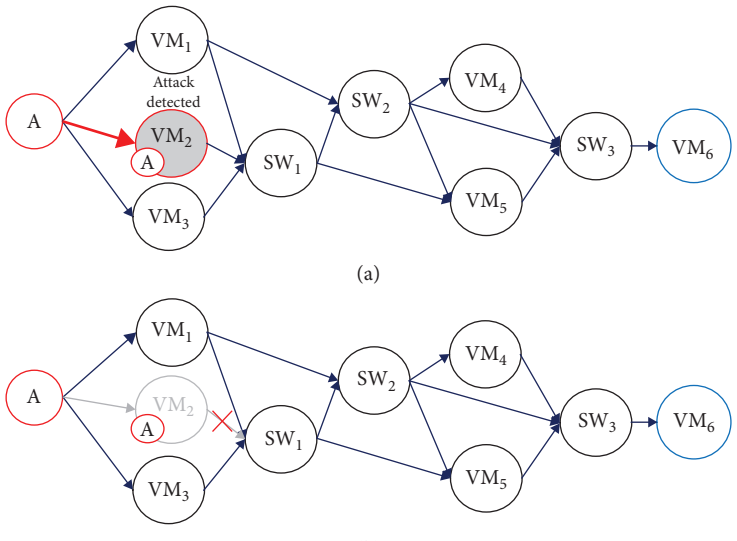


Figure 5: Continued.

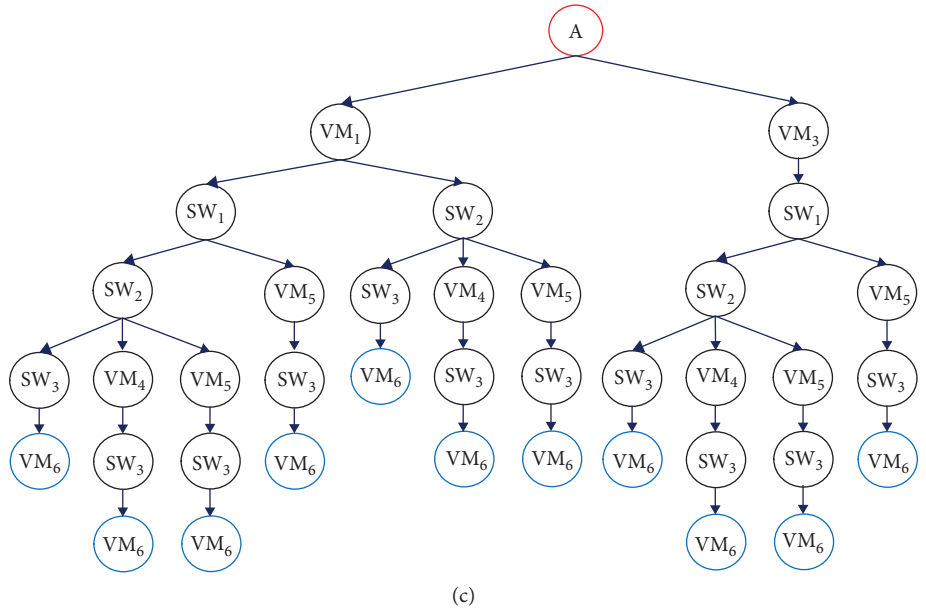


FIGURE 5: Attack detection and countermeasure without detection delay. (a) Attack detection at $VM_2$. (b) Applying flow table to block $VM_2$. (c) Full AG after countermeasure applied.

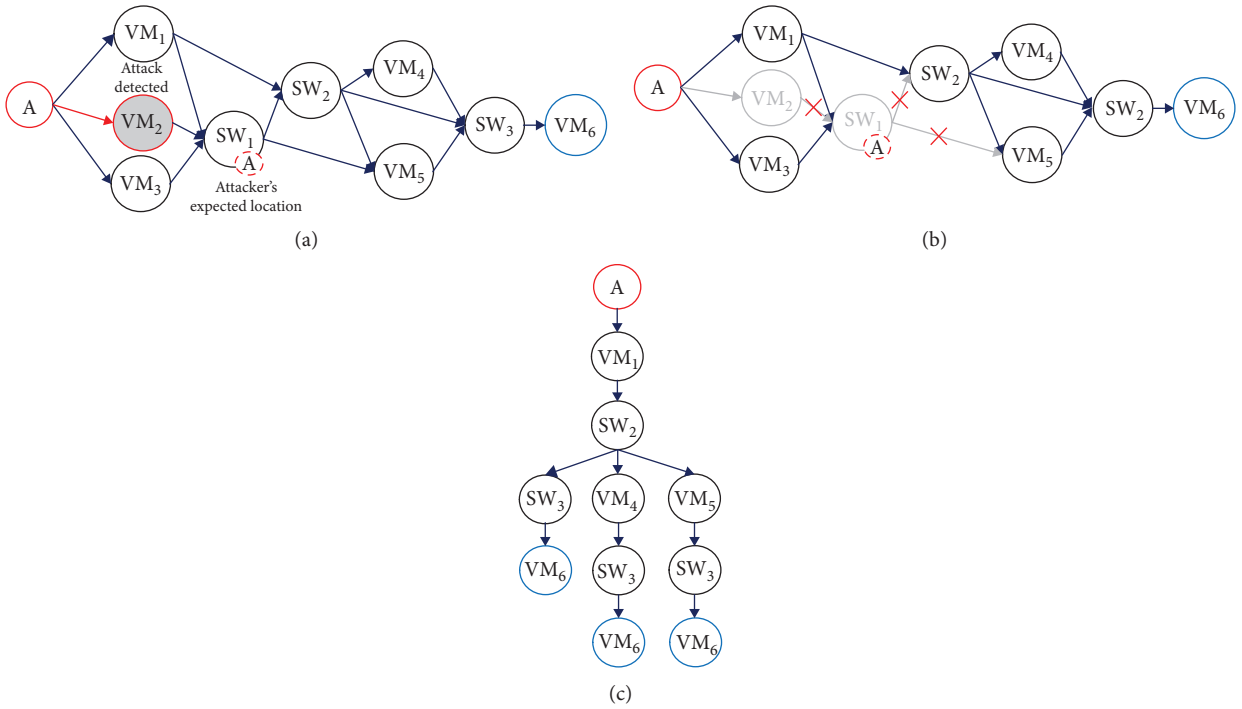


FIGURE 6: Attack detection and countermeasure with detection delay. (a) Attack detection at $VM_2$. (b) Applying flow table to block $VM_2$ and $SW_1$. (c) Full AG after countermeasure applied.

*Example 2.* Figure 6(b) shows $t_R = 2$ times $t_A$. The loss nodes are $VM_2$ and $SW_1$, and the action nodes are $SW_2$ and $SW_3$. Assuming that both the cost of damage from the attack and the cost of the action are 100, the total cost is 400.

If $t_R$ is less than $t_A$, the attack can take immediate action on the detected node. But in reality, this is not always true, and therefore the attacker has extra time to continue compromising nodes in the SDN. $k$ is used to determine the attacker's attack progress. It also indicates the number of possible SDN nodes that the attacker may have compromised (i.e., a predictive value to estimate the attacker's progress). Hence, the defender must take action on the nodes that is up to $k$ hops in distance when the condition of equation (6) is satisfied. For example, if $t_R$ and $t_A$ are the

same, then $k$ is 1. The defender can then take action on a node that is 1 hop away. Using Algorithm 1, the defender can select the node to take action.

$$k \times t_A \leq t_R < (k+1) \times t_A. \quad (6)$$

The countermeasures should be applied as soon as possible to minimize the loss of the entire system. If you cannot reduce $t_D$ and $t_C$ in $t_R$, you should reduce $t_S$. Rather than calculating the security model in a real time when an attack occurs, it is possible to reduce the $t_S$ by searching and applying a model that matches the current situation among the precomputed security models.

## 4. Precomputation and Attack Prediction for Security Assessment

This section introduces the precomputation of attack scenarios and attack scenario prediction by taking into account the delays in attack detections. We can precompute the attack scenarios in order to reduce the time taken to evaluate them. We also take into account the delays observed in attack detection mechanisms and propose an attack scenario prediction method to enhance the capabilities of SDN defense mechanisms. The generations of both the full graph and the HARM can be found in [17].

*4.1. Full Graph.* Assessing the security of SDN in real time faces a scalability problem using existing graphical security models as presented in Section 7. To address this problem, we precompute all possible attack scenarios using full AG. By precomputing all possible attack scenarios offline, we can reuse this information in real time when necessary. For precomputation of attack scenarios, we use a full AG, which represents all possible attack paths. Algorithm 2 is used to generate a full AG. The inputs required are the AG, attacker location, and the target node. Then, the algorithm searches for all possible attack paths of the given attack scenario. Given the attacker outside the SDN and the target node of $VM_6$, the full AG of the example SDN is shown in Figure 7. For simplicity, we only represented attack paths of VMs as the size of the full AG grows exponentially relative to the AG above.

*4.2. HARM.* The full AG above is used for fast real-time security assessment for particular attack scenarios. However, it is not scalable to enumerate all possible attack scenarios for a security overview of the SDN. Instead, we use the HARM [17] to assess the security of the SDN in a more scalable manner. The HARM models network nodes and their vulnerabilities onto multiple layers and utilizes the benefits of hierarchy to reduce the scalability complexity. We generate a 2-HARM (a two-layered HARM) of the example SDN, as shown in Figure 8. The formalism of the 2-HARM is as follows.

*Definition 4.* The two-layered HARM is defined as a 3-tuple $H = (U, L, M)$. Here, $U$ is the AG and $L$ is the ATs for $H$ and $V$, where $M$ is the mapping between the upper layer components and lower layer components. This mapping is described by $M: U \longrightarrow L$. Each host in the upper layer may have a corresponding AT in the lower layer.

The upper layer of the HARM uses the AG to represent the reachability between the nodes in the SDN (i.e., the VMs and the switches). Hence, we define $U$ as follows.

*Definition 5.* An AG in the upper layer of the HARM is defined as a 2-tuple $U = (N, E)$, where $N$ is a finite set of nodes in the SDN and $E \subseteq N \times N$ is a set of edges where a pair of nodes.

The lower layer of the HARM is a set of attack trees (ATs) [29], where each AT represents the vulnerability information of each upper layer node of the HARM (i.e., SDN nodes). We define each $L$ in the lower layer of the HARM as follows.

*Definition 6.* An AT in the lower layer of the HARM is defined as a 5-tuple $L = (A, B, c, g, \text{root})$, where $A$ is a finite set of vulnerabilities and $B$ is a set of gates which are the inner nodes of $L$. We require $A \cap B = \emptyset$ and root $\in A \cup B$. Function $c$: $B \longrightarrow P(A \cup B)$ describes the children of each inner node in at (we assume there are no cycles). Function $g$: $B \longrightarrow \{\text{AND}, \text{OR}\}$ describes the type of each gate. The

```
procedure RNS(AG_SDN, N_D, t_A, t_R)
  if t_A > t_R then
    Send N_D to Reconfiguration Module
  else
    for all E from N_D to N_i do
      t_R ⟵ t_R − t_A
      RNS(AG_SDN, N_i, t_A, t_R)
    end for
  end if
end procedure
```

Algorithm 1: Response node selection algorithm.

```
procedure fullAG(AG, N_cr, N_tg)
  Mark N_cr visited
  Stack.push(N_cr)
  if N_cr = N_tg then
    Return Stack
  else if Sizeof(E_Ncr) ≠ 0 then
    for i⟵0 to Sizeof(E_Ncr) in AG do
      N_next ⟵ destination of E_i^Ncr
      if N_next is uniquely aligned then
        fullAG (AG, N_next, N_tg)
      end if
    end for
  else
    Stack.clear
  end if
end procedure
```

Algorithm 2: Algorithm to generate a full AG.

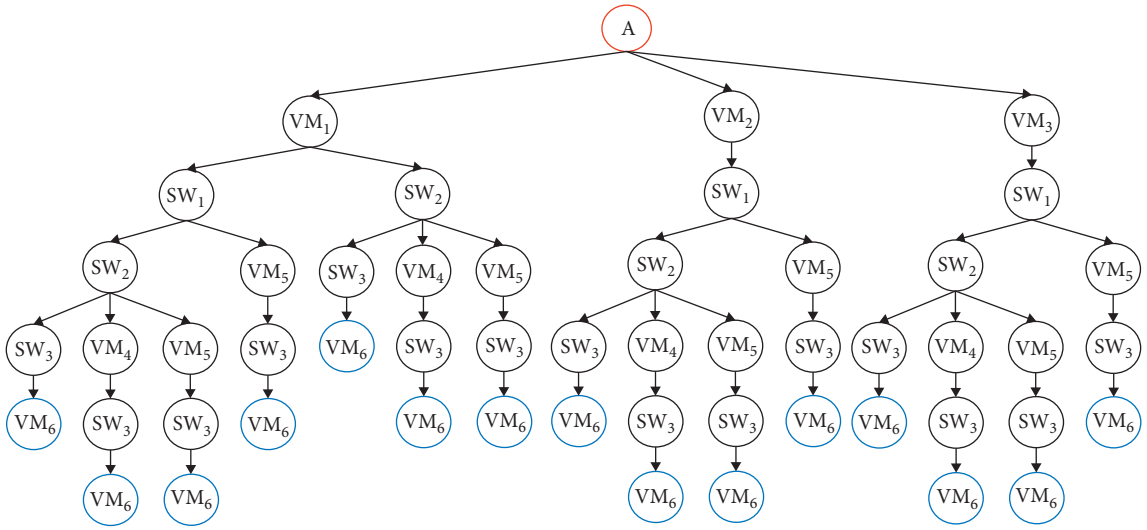


Figure 7: A full AG of the SDN

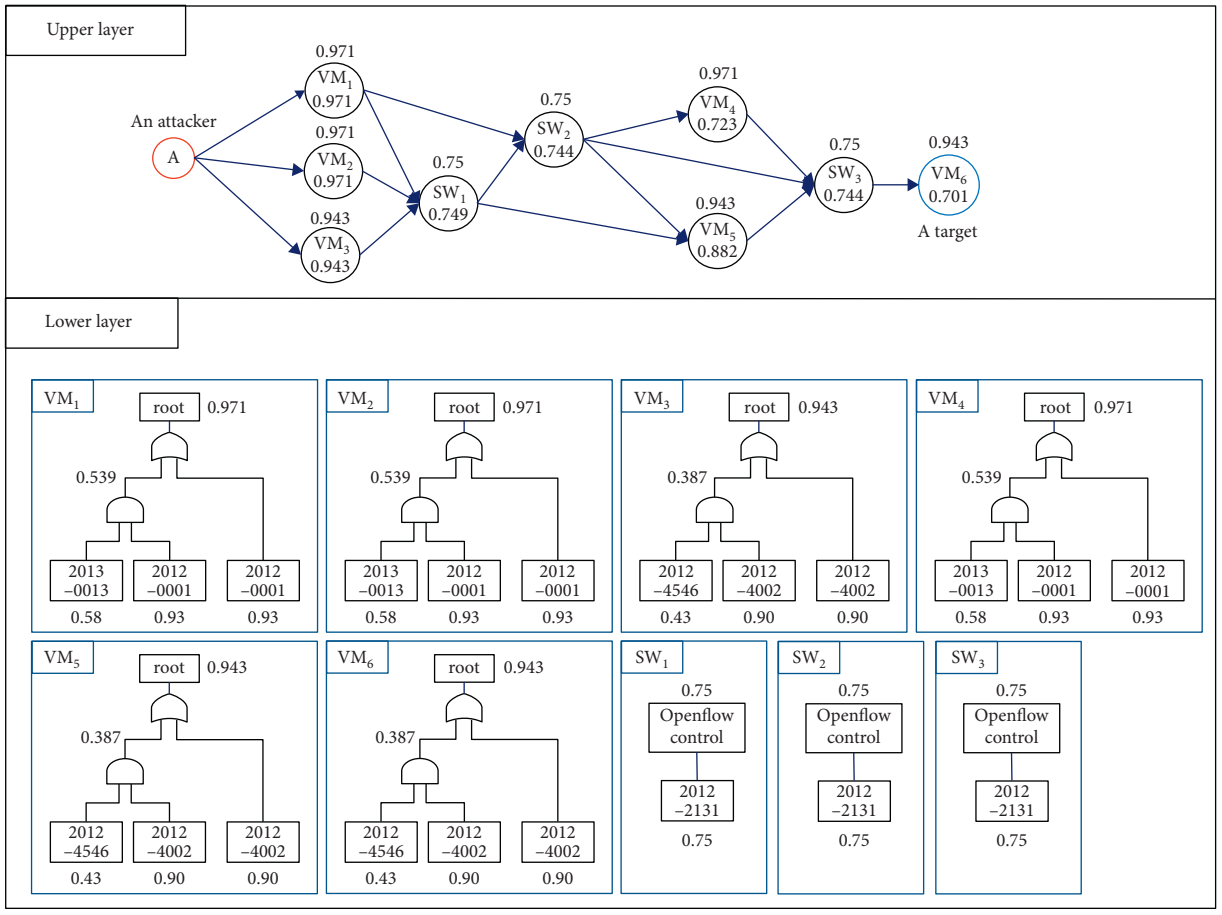


Figure 8: A HARM of the SDN in Figure 4.

representation of the attack tree $L^n$ associated to the host $n \in N$ is as follows:

$$L^n: A \subseteq n_{\text{vuls}}. \tag{7}$$

This means that the vulnerabilities of a node are combined using logical AND and OR gates.

Given the definitions above, the example SDN in the form of the HARM can be represented as follows.

*Example 3. The Upper and Lower Layer Mapping:* Figure 8 shows HARM of the SDN. The HARM for given SDN model is $H = (U, L, M)$, where $U$ and $L$ are the AG and the set of

ATs in the upper and the lower layer, and $M$: $U \longrightarrow L$ is a one-to-one mapping of the upper layer $U$ to the corresponding lower layer $L$.

*Example 4. The Upper Layer:* the AG shown in Figure 8 is a directed graph $AG_{SDN} = (N_{SDN}, E_{SDN})$, where $N_{SDN} = \{A, VM_1, VM_2, VM_3, VM_4, VM_5, VM_6, SW_1, SW_2, SW_3\}$ and $E_{SDN} = \{(A, VM_1), (A, VM_2), (A, VM_3), (VM_1, SW_1), (VM_2, SW_1), (VM_3, SW_1), (VM_1, SW_2), (SW_1, SW_2), (SW_1, VM_5), (SW_2, VM_4), (SW_2, VM_5), (SW_2, SW_3), (VM_4, SW_3), (VM_5, SW_3), (SW_3, VM_6)\}$.

*Example 5. The Lower Layer:* the ATs in the lower layer are shown in Figure 8. The set of conditions required to compromise $VM_1$ is given by $L^{VM_1} = (A^{VM_1}, B^{VM_1}, c^{VM_1}, g^{VM_1}, \text{root}^{VM_1})$, where $A^{VM_1} = \{WV_1, WV_2, WV_2'\}$ is a set of components which are the leaves (vulnerabilities), $B^{VM_1} = \{AND_1, OR_1\}$, $c^{AND_1} = \{WV_1, WV_2'\}$, $c^{OR_1} = \{AND_1, WV_2\}$, $g^{VM_1}(\text{root}_{VM_1}) = OR_1$, and $\text{root}^{VM_1} = \text{root}$, $\text{root} \in A^{VM_1} \cup B^{VM_1}$.

## 5. Result and Analysis

In this section, we investigate the effectiveness of using full AG for precomputation taking into account various security metrics. Regardless of which model we use, the security metric computed will be the same. Since both full AG and the HARM computes the same metric values, we do not explicitly present those results in this paper.

First, we look at changes in security metrics with and without deploying countermeasure, where we change the flow table rules to block attack paths up to three steps in Section 5.1. Then, we conduct simulations to investigate the performance difference of computing an AG used in the HARM to a full AG for precomputation in Section 5.3.

*5.1. Change in Security Metrics.* For this experiment, we use the example SDN as shown in Figure 3 as our experimental testbed. In this system, service is not available unless a packet is sent to the database. So, we assume that the network administrator cannot change the flow table rules of $SW_3$ and $VM_6$ due to system constraints (i.e., they need to be functional to continuously provide SDN service). To ensure the operability, we extend this assumption such that at least one connection path exists such that users' requests can be handled. Although modifying flows can affect the performance of the SDN, we only consider the minimal cost to enhance the security of SDN in this paper (i.e., the minimum number of flow changes for maximized security). For example, an alternative flow path can be used to continue delivering the service, but it may create a bottleneck effect if the traffic is not managed carefully. We will investigate the trade-off between enhancing security and degrading the network performance in our future work.

First, we investigate the change in security when predicting potential attack in 1-hop, and then we measure the change in the probability of attack success and the system risk. The result is shown in Figure 9, which shows that blocking 1-hop at $SW_1$ or $SW_2$ flows can minimize the probability and the risk than other nodes.

On the other hand, if the detection of an attack was delayed, we need to consider further steps in order to mitigate the attack. So, we also look at 2-hop flow blocking of nodes, where the combinations are shown in Table 3. The result is shown in Figure 10, which shows a similar result to the 1-hop blocking (i.e., the best practice is to block flow through $SW_1$ or $SW_2$). However, we observe that the importance of nodes for defense has changed (i.e., the priorities to secure SDN components can vary when the number of hops changes). For instance, blocking the flow through $VM_2$ and $SW_2$ can also achieve a similar effect, where $VM_2$ in the 1-hop analysis was significantly worse.

Lastly, we look at the 3-hop flow blocking. Table 4 shows the combinations of three nodes and their flows to be blocked. With the given attack scenario, we have 21 possible combinations of nodes out of the maximum number of 35. Figure 11 shows the result, where three conditions that include $SW_2$ minimized the probability of attack success and the system risk, but only one condition that includes $SW_1$. This indicates that we look into various attack paths as well as the importance of nodes. In conclusion, we observe that our proposed solution has identified $SW_2$ as the most important SDN component to secure. In general, the most vulnerable node or the node with many connections to other nodes in the network can be the most important node. Another method of analyzing the importance of nodes is the network centrality measure [30]. For the running example, it is obvious to pick it up easily by inspection, but when the SDN becomes larger and more complex, this can be done easily using the proposed solution, whereas it would be near impossible and impractical by human efforts.

*5.2. Numerical Sensitivity Analysis.* The slower the response to an attack, the more attackers can attack the node. This results in more loss to the system. We conducted an experiment to compare the losses incurred in the system with the costs required to take action in response time. Since loss cost and cost of action cannot be defined objectively, the sensitivity analysis methodology was applied. In this experiment, we calculated loss and response costs based on detection time and attack time when an attacker successfully attacked $VM_2$.

In the first experiment, we applied a sensitivity analysis to the loss cost. The corresponding cost was fixed at 100 and the loss cost increased from 0 to 500. In each case, the total cost of ownership was calculated. Figure 12(a) shows the experimental result. As the response time is slower than the attack time, the total cost is higher.

Second, we applied a sensitivity analysis to the response costs. The loss cost was fixed at 100 and the corresponding cost was increased from 0 to 500. And, as in the previous experiment, we calculated the total cost for each case. Experimental results show that the total cost of ownership varies depending on the situation, such as Figure 12(b). If a defender defends a node that is far from the compromised

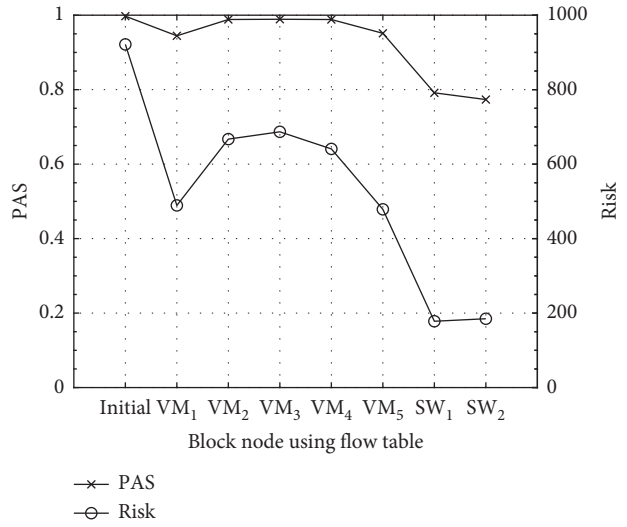


Figure 9: Block one node vs. security metrics.

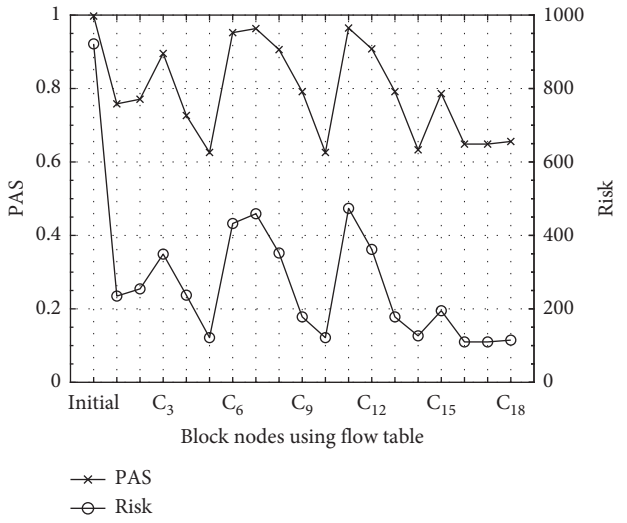


Figure 10: Block two nodes vs. security metrics.

Table 3: A set of two-node block conditions.

| ID | Nodes |
|---|---|
| $C_1$ | $VM_1VM_2$ |
| $C_2$ | $VM_1VM_3$ |
| $C_3$ | $VM_1VM_4$ |
| $C_4$ | $VM_1VM_5$ |
| $C_5$ | $VM_1SW_2$ |
| $C_6$ | $VM_2VM_3$ |
| $C_7$ | $VM_2VM_4$ |
| $C_8$ | $VM_2VM_5$ |
| $C_9$ | $VM_2SW_1$ |
| $C_{10}$ | $VM_2SW_2$ |
| $C_{11}$ | $VM_3VM_4$ |
| $C_{12}$ | $VM_3VM_5$ |
| $C_{13}$ | $VM_3SW_1$ |
| $C_{14}$ | $VM_3SW_2$ |
| $C_{15}$ | $VM_4VM_5$ |
| $C_{16}$ | $VM_4SW_1$ |
| $C_{17}$ | $VM_4SW_2$ |
| $C_{18}$ | $VM_5SW_1$ |

Table 4: A set of three-node block conditions.

| ID | Nodes |
|---|---|
| $C_1$ | $VM_1VM_2VM_4$ |
| $C_2$ | $VM_1VM_2VM_5$ |
| $C_3$ | $VM_1VM_2SW_2$ |
| $C_4$ | $VM_1VM_3VM_4$ |
| $C_5$ | $VM_1VM_3VM_5$ |
| $C_6$ | $VM_1VM_3SW_2$ |
| $C_7$ | $VM_1VM_4VM_5$ |
| $C_8$ | $VM_1VM_4SW_2$ |
| $C_9$ | $VM_2VM_3VM_4$ |
| $C_{10}$ | $VM_2VM_3VM_5$ |
| $C_{11}$ | $VM_2VM_3SW_1$ |
| $C_{12}$ | $VM_2VM_3SW_2$ |
| $C_{13}$ | $VM_2VM_4VM_5$ |
| $C_{14}$ | $VM_2VM_4SW_1$ |
| $C_{15}$ | $VM_2VM_4SW_2$ |
| $C_{16}$ | $VM_2VM_5SW_1$ |
| $C_{17}$ | $VM_3VM_4VM_5$ |
| $C_{18}$ | $VM_3VM_4SW_1$ |
| $C_{19}$ | $VM_3VM_4SW_2$ |
| $C_{20}$ | $VM_3VM_5SW_1$ |
| $C_{21}$ | $VM_4VM_5SW_1$ |

node, loss cost may occur at a node with a relatively short distance. However, if the cost of response is greater than the cost of loss, taking action on multiple nodes significantly increases the total cost of ownership. In this case, taking action on one node that is farther away, even if the loss is considered, may be a way to save the total cost of ownership.

*5.3. Simulation.* To investigate the performance of precomputing the full AG in comparison to the AG, we simulate the generation and evaluation time via simulations. The precomputation of the full AG is important as it reduces the security evaluation time for real-time mitigation, while it is also used for attack prediction. As increasing the number of nodes put both AG and full AG in an exponential time complexity [16], we focus on generation and evaluation when certain node flows are blocked as shown in Table 5.

The comparison results are shown in Figure 13; it shows that the full AG outperforms the AG in terms of evaluation time for all the conditions. This indicates that real-time security assessment for a large-sized SDN (or any other general networks) using AG may not be feasible [17] and there is an efficiency of precomputing all possible attack paths using the full AG. And it is more efficient to utilize more scalable security models such as HARM.

## 6. Discussion and Limitations

*6.1. Scalability.* The framework provides an approach to assessing the security of SDN and applying countermeasures to the system using a security model for real-time

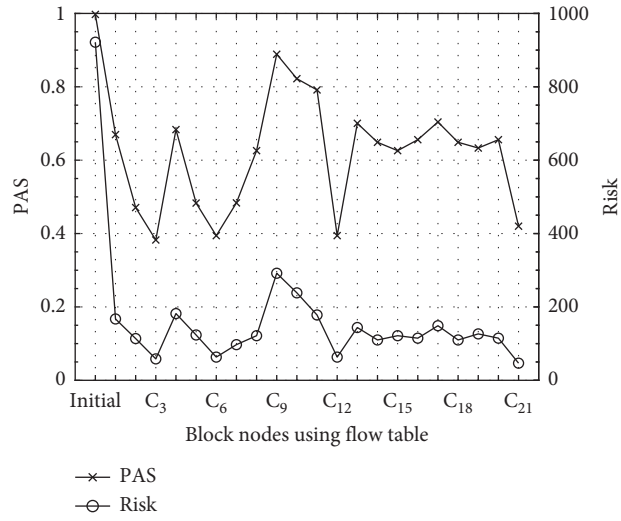


Figure 11: Block three nodes vs. security metrics.

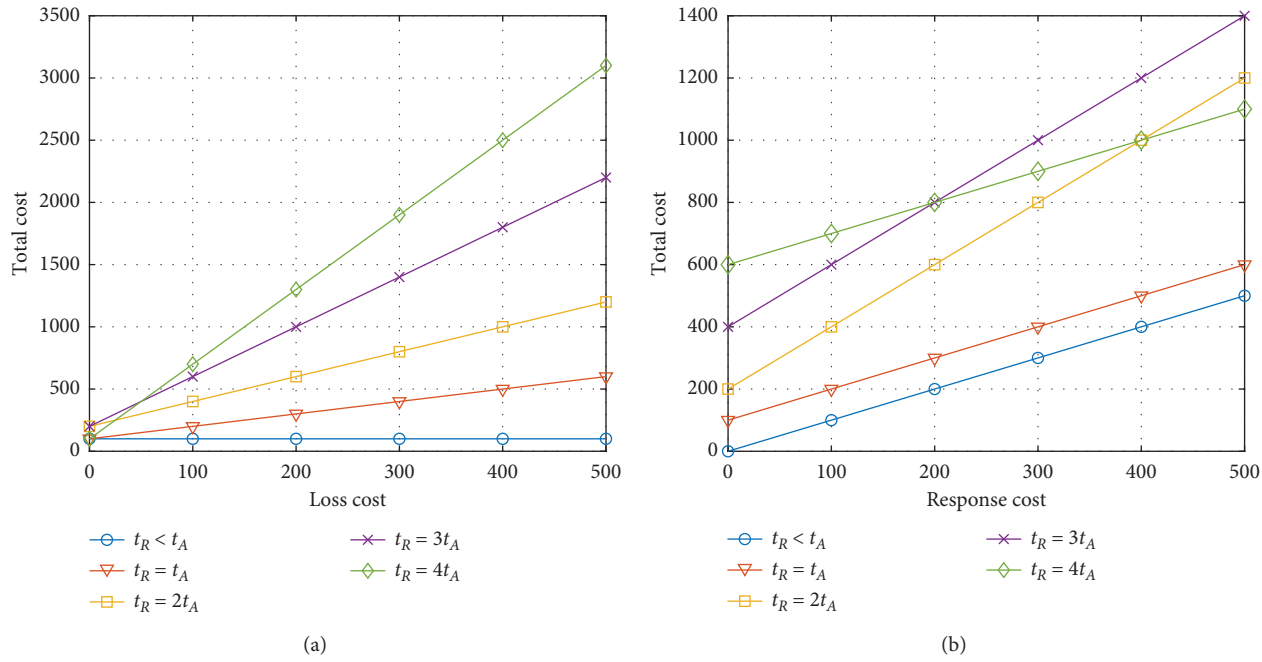


Figure 12: Cost sensitivity analysis. (a) Loss cost vs. total cost. (b) Response cost vs. total cost.

intrusion responses. However, the security model has scalability issues. In our future work, we will consider improving the performance of security modeling and analysis for the SDN, as we face an exponential time complexity when the number of nodes in the SDN increases.

*6.2. SDN Attack Surface.* Furthermore, we use network devices that exist in the data plane for security modeling. However, SDN has a variety of components and threat vectors in addition to the data plane. Accordingly, we will incorporate the control plane and the SDN controller in the model in order to assess the security posture of the whole life-cycle of the SDN. In addition, the network may normally have an internal attacker. But, we only used scenarios in which attacker would always break in from the outside. We can deal with internal attacker in our future work.

Table 5: A set of conditions that include specific node(s).

| ID | Nodes |
|---|---|
| $C_1$ | $VM_1$ |
| $C_2$ | $SW_2$ |
| $C_3$ | $VM_1SW_2$ |
| $C_4$ | $VM_4SW_1$ |
| $C_5$ | $VM_2VM_4SW_2$ |
| $C_6$ | $VM_3VM_5SW_1$ |

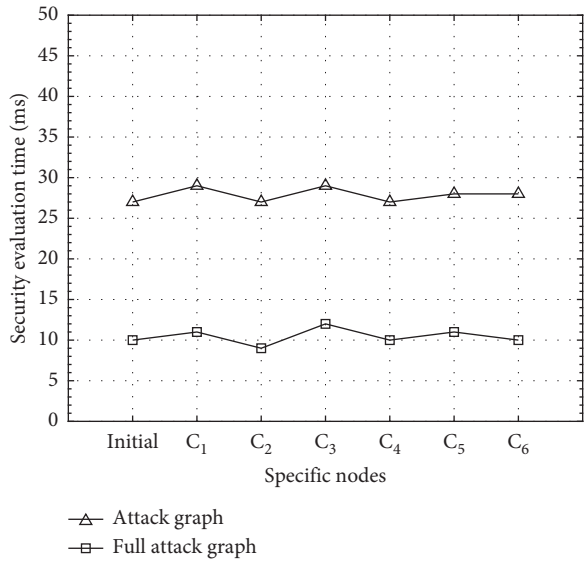


Figure 13: Specific node(s) vs. security evaluation time.

*6.3. Evaluation Correctness of Different Vulnerabilities.* In this paper, we removed the assumption that an IDS may not work in real time, and also the detection rate may not be 100%. This is more realistic as an attack which was detected at the entry point but progressed to install a backdoor on an user's computer may not be mitigated effectively by only enhancing the entry point vulnerability. As such, we go beyond securing the point of detection and evaluate the possible extensions of the damage. However, we did not explicitly evaluated the effects of different vulnerabilities present, where some system settings may be mitigated well by securing the detection point (e.g., security by design). As we only focused on the usefulness of precomputation, we will investigate the impact of secure design in the context of our work in the future.

*6.4. Multiple Attackers.* In this paper, we focused only on a single attacker version, but in reality, there can be multiple attackers trying to exploit the SDN in various entry routes. However, because we already precomputed all possible attack paths and the IDSes are working in real time, our attack response module only has to make sure that different attackers are differentiated when formulating countermeasures. In this way, our proposed solution can mitigate multiple attackers even exploiting different attack surfaces. On the other hand, we only handled targeted attacks. Therefore, volume-based attacks such as DDoS are not appropriately addressed by our solution. We will investigate the mitigation schemes for volume-based attacks in the SDN in our future work.

*6.5. Network Topology Changes.* In this paper, we changed the network flow to respond to the attack when it occurred. However, the network topology of an SDN can be dynamic. Therefore, when the model changes, it is necessary to redo the precomputation of the changing model. This could be a similar solution to the MTD in response to an attack [5]. In our future work, we will investigate the application of precomputation to changing models.

## 7. Related Work

*7.1. General SDN Security.* Various security issues related to the SDN have been studied previously [1, 31–34]. Kreutz et al. [6] presented new threat vectors of the SDN that were not present in traditional networks. They also provided some potential solutions to mitigate their identified threats. However, they do not specify the means to evaluate those threats. We aim to provide solutions to some of these problems in this paper. Shin et al. [35] presented a framework named *FRESCO*, which is to enhance the security of the OpenFlow protocol for the SDN, allowing them to detect and mitigate attacks. This work shows that SDN components such as SDN switches are prone to cyber attacks. Porras et al. [36] presented SE-Floodlight, an extension to a widely used OpenFlow Floodlight Controller, to provide additional security features to protect the control plane of the SDN. Our paper aims to leverage these technologies and provide a comprehensive security analysis of the SDN.

*7.2. Intrusion Detection in SDN.* An intrusion is defined as a successfully carried out attack, including any malicious behavior in the system. Intrusion detection is an activity to detect such intrusions. It is ideal to detect all the intrusion with 100% accuracy, but in practice, this is infeasible. Because intrusion detection is not perfect, there is always false alarm such as true-positive and false-positive rates. Several types of research have been conducted on attack detection in the SDN environment. Dhawan et al. [25] proposed SPHINX, a framework to detect attacks on network topology and data plane forwarding. Braga et al. [26] proposed a lightweight method for detecting DDoS attacks based on traffic flow capabilities. Giotis et al. [27] presented how to apply SDN for distributed denial of service (DDoS) mitigation by using OpenFlow protocol as a means to enhance the legacy Remote Triggered Black-Hole (RTBH). And in [28], they performed anomaly detection and mitigation in the SDN architecture through an efficient and scalable mechanism. Although not perfect, we can leverage these techniques to detect intrusions in the SDN, which can be used to formulate optimal countermeasures using the HARM.

*7.3. Security Modeling for SDN.* For security modeling and analysis, Chung et al. [37] presented NICE, a network intrusion detection and countermeasure selection framework. The core of this framework is using an AG to evaluate the security. However, there are limitations of using an AG due to its scalability issues. Similarly, many of the existing graphical security models, as described in [10, 11], suffer from the scalability and adaptability problems [5]. Typically, the scalability problem arises when these models have real-time constraints. In order to address this issue, our approach is to precompute attack scenarios in advance and use them

where necessary. Hence, we propose to use a full AG, as presented in [18], to generate all possible attack paths in the precomputation. Moreover, we also use the HARM [17] that supports scalable and adaptable security modeling and analysis. By precomputing all possible attack paths, we can quickly evaluate the security posture of the SDN when an intrusion is detected and formulate effective countermeasures in real time.

## 8. Conclusion

SDN provides functionalities that can dynamically control the network flow, enabling a more robust and economical way of managing network communications. However, it also introduced new vulnerabilities and attack vectors that were not present previously. As a result, many have proposed security solutions to strengthen the SDN against cyber attacks. Nevertheless, there is still a lack of security modeling and analysis for SDN, which enables a system administrator to have a systematic security overview of the SDN.

In this paper, we proposed a security modeling and analysis framework with precomputation for the SDN to formulate countermeasures in real time. The precomputation method was used for the AG, full AG, and the HARM to generate attack scenarios of the current SDN, which can then be used in conjunction with the IDS and correlate with the precomputed attack scenarios. Further, the precomputed models can be used for predicting potential attacks in case the detection mechanisms are delayed. To verify, we carried out experimental analysis on the SDN testbed and simulations, which showed that our proposed approaches would be practical and effective in the SDN to defend against an ongoing attack in real time.

## Data Availability

The vulnerability data used to support the findings of this study are included within the article.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## References

[1] G. Stabler, A. Rosen, S. Goasguen, and K.-C. Wang, "Elastic ip and security groups implementation using openflow," in *Proceedings of the 6th International Workshop on Virtualization Technologies in Distributed Computing Date, ser. VTDC '12*, pp. 53–60, ACM, New York, NY, USA, 2012.

[2] D. Kreutz, F. M. V. Ramos, P. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: a comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, p. 63, 2015.

[3] M. Tariq, B. Koldehofe, S. Bhowmik, and K. Rothermel, "PLEROMA: A SDN-based high performance publish/subscribe middleware," in *Proceedings of the 15th International Middleware Conference (Middleware 2014)*, pp. 217–228, ACM, New York, NY, USA, 2014.

[4] J. Jafarian, E. Al-Shaer, and Q. Duan, "Openflow random host mutation: transparent moving target defense using software defined networking," in *Proc. of the 1st Workshop on Hot Topics in Software Defined Networks (HotSDN 2012)*, pp. 127–132, ACM, New York, NY, USA, 2012.

[5] J. B. Hong and D. S. Kim, "Assessing the effectiveness of moving target defenses using security models," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 2, pp. 163–177, 2016.

[6] D. Kreutz, F. M. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, ser. HotSDN '13*, pp. 55–60, ACM, New York, NY, USA, 2013.

[7] S. Lee, C. Yoon, C. Lee, S. Shin, V. Yegneswaran, and P. A. Porras, "Delta: a security assessment framework for software-defined networks," in *Proceedings of the 2017 Network and Distributed System Security Symposium*, San Diego, CA, USA, March 2017.

[8] S. Lee, J. Kim, S. Shin, P. Porras, and V. Yegneswaran, "Athena: A framework for scalable anomaly detection in software-defined networks," in *Proceedings of the 2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 249–260, Denver, CO, USA, June 2017.

[9] S. Nanda, F. Zafari, C. DeCusatis, E. Wedaa, and B. Yang, "Predicting network attack patterns in sdn using machine learning approach," in *Proceedings of the 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 167–172, Palo Alto, CA, USA, November 2016.

[10] B. Kordy, L. Piètre-Cambacédès, and P. Schweitzer, "DAG-based attack and defense modeling: don't miss the forest for the attack trees," *Computer Science Review*, vol. 13-14, pp. 1–38, 2014.

[11] J. B. Hong, D. S. Kim, C.-J. Chung, and D. Huang, "A survey on the usability and practical applications of graphical security models," *Computer Science Review*, vol. 26, pp. 1–16, 2017.

[12] H. Xu, J. Su, X. Zong, and L. Yan, "Attack identification for software-defined networking based on attack trees and extension innovation methods," in *Proceedings of the 2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 1, pp. 485–489, Bucharest, Romania, September 2017.

[13] L. Yao, P. Dong, T. Zheng, H. Zhang, X. Du, and M. Guizani, "Network security analyzing and modeling based on petri net and attack tree for sdn," in *Proceedings of the 2016 International Conference on Computing, Networking and Communications (ICNC)*, pp. 1–5, Kauai, HI, USA, February 2016.

[14] A. Roy, D. Kim, and K. Trivedi, "Scalable optimal countermeasure selection using implicit enumeration on attack countermeasure trees," in *Proceedings of the 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012)*, pp. 1–12, IEEE Computer Society, Los Alamitos, CA, USA, June 2012.

[15] J. Hong and D. Kim, "Performance analysis of scalable attack representation models," in *Security and Privacy Protection in Information Processing Systems (SEC 2013)*, L. Janczewski, H. Wolfe, and S. Shenoi, Eds., vol. 405, pp. 330–343, Springer, Berlin, Germany, 2013.

[16] R. Lippmann and K. Ingols, "An Annotated Review of Past Papers on Attack Graphs," Technical report ESC-TR-2005-054, MIT Lincoln Laboratory, Lexington, MA, USA, 2005.

[17] J. B. Hong and D. S. Kim, "Towards scalable security analysis using multi-layered security models," *Journal of Network and Computer Applications*, vol. 75, pp. 156–168, 2016.

[18] K. Ingols, R. Lippmann, and K. Piwowarski, "Practical attack graph generation for network defense," in *Proceedings of the 22nd Annual Computer Security Applications Conference (ACSAC 2006)*, pp. 121–130, Miami Beach, FL, USA, December 2006.

[19] Z. Shu, J. Wan, D. Li, J. Lin, A. V. Vasilakos, and M. Imran, "Security in software-defined networking: threats and countermeasures," *Mobile Networks and Applications*, vol. 21, no. 5, pp. 764–776, 2016.

[20] M. Albanese, S. Jajodia, and S. Noel, "Time-efficient and cost-effective network hardening using attack graphs," in *Proceedings of the 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012)*, IEEE Computer Society, Los Alamitos, CA, USA, June 2012.

[21] J. Beale, R. Deraison, H. Meer, R. Temmingh, and C. Walt, *The NESSUS project*, Syngress Publishing, Burlington, MA, USA, 2002, http://www.nessus.org.

[22] O. Developers, "The open vulnerability assessment system (openvas)," 2012.

[23] J. Hong and D. Kim, "Scalable security model generation and analysis using k-importance measures," in *Security and Privacy in Communication Networks (SecureComm 2013)*, T. Zia, A. Zomaya, V. Varadharajan, and M. Mao, Eds., vol. 127, pp. 270–287, Springer, Berlin, Germany, 2013.

[24] M. Schiffman, G. Eschelbeck, D. Ahmad, A. Wright, and S. Romanosky, *CVSS: A Common vulnerability scoring system*, National Infrastructure Advisory Council (NIAC), Google Scholar, 2004.

[25] M. Dhawan, R. Poddar, K. Mahajan, and V. Mann, "Sphinx: detecting security attacks in software-defined networks," in *Proceedings of the 2015 Network and Distributed System Security Symposium*, vol. 15, pp. 8–11, San Diego, CA, USA, February 2015.

[26] R. Braga, E. M. M. Braga, and A. Passito, "Passito, "Lightweight ddos flooding attack detection using nox/openflow," in *Proceedings of the 2010 IEEE 35th Conference on Local Computer Networks, ser. LCN '10*, pp. 408–415, IEEE Computer Society, Washington, DC, USA, October 2010.

[27] K. Giotis, G. Androulidakis, and V. Maglaris, "Leveraging sdn for efficient anomaly detection and mitigation on legacy networks," in *Proceedings of the 2014 Third European Workshop on Software Defined Networks, ser. EWSDN '14*, pp. 85–90, IEEE Computer Society, Washington, DC, USA, September 2014.

[28] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments," *Computer Networks*, vol. 62, pp. 122–136, 2014.

[29] B. Schneier, *Secrets and Lies: Digital Security in a Networked World*, John Wiley & Sons, Hoboken, NJ, USA, 2000.

[30] J. Hong and D. Kim, "Scalable security analysis in hierarchical attack representation model using centrality measures," in *Proceedings of the 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSNW 2013)*, pp. 1–8, Budapest, Hungary, June 2013.

[31] N. Gude, T. Koponen, J. Pettit et al., "Nox: towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.

[32] M. Casado, T. Garfinkel, A. Akella et al., "A protection architecture for enterprise networks," in *Proceedings of the 15th Conference on USENIX Security Symposium-Volume 15, ser. USENIX-SS'06*, USENIX Association, Berkeley, CA, USA, July 2006.

[33] J. Matias, J. Garay, A. Mendiola, N. Toledo, and E. Jacob, "Flownac: flow-based network access control," in *Proceedings of the 2014 Third European Workshop on Software Defined Networks, ser. EWSDN '14*, pp. 79–84, IEEE Computer Society, Washington, DC, USA, September 2014.

[34] J. B. Guang Yao and P. Xiao, "Source address validation solution with openflow/nox architecture," in *Proceedings of the 2011 19th IEEE International Conference on Network Protocols*, pp. 7–12, Vancouver, Canada, October 2011.

[35] S. Shin, P. A. Porras, V. Yegneswaran et al., "Modular composable security services for software-defined networks," in *Proceedings of the 20th Annual Network & Distributed System Security Symposium*, The Internet Society, San Diego, CA, USA, 2013.

[36] P. Porras, S. Cheung, M. Fong, K. Skinner, and V. Yegneswaran, "Securing the software-defined network control layer," in *Proceedings of the 2015 Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, USA, February 2015.

[37] 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*, vol. 10, no. 4, pp. 198–211, 2013.