Received 23 March 2023, accepted 16 April 2023, date of publication 21 April 2023, date of current version 27 April 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3269018

RESEARCH ARTICLE

Time-Based Moving Target Defense Using Bayesian Attack Graph Analysis

HYEJIN KIM[1], EUISEOK HWANG[1], (Senior Member, IEEE), DONGSEONG KIM[2], JIN-HEE CHO[3], (Senior Member, IEEE), TERRENCE J. MOORE[4], (Member, IEEE), FREDERICA F. NELSON[4], AND HYUK LIM[5], (Member, IEEE)

[1]Gwangju Institute of Science and Technology (GIST), Gwangju 61005, Republic of Korea
[2]The University of Queensland, Brisbane, QLD 4072, Australia
[3]Virginia Tech, Falls Church, VA 22043, USA
[4]U.S. Army Research Laboratory, Adelphi, MD 20783, USA
[5]Korea Institute of Energy Technology (KENTECH), Naju-si 58217, Republic of Korea

Corresponding author: Hyuk Lim (hlim@kentech.ac.kr)

This work was supported in part by the U.S. Army Combat Capabilities Development Command (CCDC) International Technology Center—Pacific (ITC-PAC) and the CCDC Army Research Laboratory (CCDC-ARL) under Grant FA5209-19-P-A056, and in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) funded by the Korean Government [Ministry of Science and ICT (MSIT)] through the Privacy Risk Analysis and Response Technology Development for Artificial Intelligence (AI) Systems under Grant 2021-0-00379.

ABSTRACT The moving target defense (MTD) is a proactive cybersecurity defense technique that constantly changes potentially vulnerable points to be attacked, to confuse the attackers, making it difficult for attackers to infer the system configuration and nullify reconnaissance activities to a victim system. We consider an MTD strategy for software-defined networking (SDN) environment where every SDN switch is controlled by a central SDN controller. As the MTD may incur excessive usage of the network/system resources for cybersecurity purposes, we propose to perform the MTD operations adaptively according to the security risk assessment based on a Bayesian attack graph (BAG) analysis. For accurate BAG analysis, we model random and weakest-first attack behaviors and incorporate the derived analytical models into the BAG analysis. Using the BAG analysis result, we formulate a knapsack problem to determine the optimal set of vulnerabilities to be reconfigured under a constraint of SDN reconfiguration overhead. The experiment results prove that the proposed MTD strategy outperforms the full MTD and random MTD counterparts in terms of the maximum/average of attack success probabilities and the number of SDN reconfiguration updates.

INDEX TERMS Moving target defense, Bayesian attack graph, software-defined networking.

I. INTRODUCTION

Cybersecurity threats can be raised from various security faults and vulnerabilities in operating systems (OSs), software/hardware, or network applications/services. In recent years, cybersecurity threats have rapidly evolved and advanced significantly to bypass or nullify conventional security defense systems, such as firewalls and intrusion detection systems (IDSs). For example, advanced persistent threat (APT) is one of the significant threats accessing a targeted network stealthily and performing malicious reconnaissance for a long time until completing their goals [1].

To counter those advanced attacks, proactive defense techniques have emerged as a promising direction of defense. Moving target defense (MTD) is one of them and achieves the proactive defense by constantly changing vulnerable attack surfaces (i.e., potentially vulnerable points to be attacked) to thwart malicious attacks, confuse the attackers, and invalidate the information collected by the attackers (e.g., attack paths) [2]. However, it is a non-trivial problem to efficiently deploy MTD while maintaining its defense effectiveness under resource-constrained environments. For example, a shuffling-based MTD performs changing Internet protocol

The associate editor coordinating the review of this manuscript and approving it for publication was Guangjie Han.

(IP) and medium access control (MAC) addresses or routing paths randomly, migrating VMs from one host to another, and randomizing the data [3]. However, MTD techniques may incur system reconfiguration overhead as well as performance degradation such as service latency to legitimate users. In addition, modern IP networks have the nature of high dynamics that can change network states and their vulnerabilities over time. Hence, it is highly challenging to dynamically and adaptively identify an optimal setting that can optimally deploy the MTD in terms of minimizing defense cost and risk and maximizing the defense effectiveness (e.g., minimizing attack success).

The common vulnerability scoring system (CVSS) has been widely used as a metric of security risk assessment [4]. The CVSS evaluates the vulnerabilities of individual nodes and provides scores on a scale of 10 by considering the basic, temporal, and environmental metrics [4]. The CVSS provides only the threat level of individual vulnerability, but it is also important to perform a network-wide threat analysis of a given network with various components because the vulnerabilities of individual network/system components could interact with each other and make the system more vulnerable [5]. Attack trees (ATs) and attack graphs (AGs) have been commonly used to represent the relationships between vulnerabilities in a given network [6]. One of the promising AGs is directed acyclic AGs with probability metrics, called the *Bayesian attack graph* (BAG), which has been used to analyze uncertain, stochastic attack behaviors. BAGs have been employed to identify potential attack paths and find the most vulnerable components or assets based on the probabilistic properties of BAG analysis results. The BAG analysis can be applied to identify potential attack paths based on the level of priority. These BAG analysis results make it possible to differentiate the use of defense resources in the MTD strategy by handling more vulnerable system components in the attack graph with high priority.

In this paper, we aim to develop a time-based MTD strategy that determines an optimal set of hosts whose addresses are to be shuffled to maximize the system's security level while limiting the usage of network resources for cybersecurity at a certain level not to degrade network service quality for legitimate users. To accurately assess the system's security risk, we model and analyze the attacker's behavior using the BAGs. We formulate the optimization problem for the MTD strategy as a knapsack problem, which considers the vulnerable levels of hosts identified by BAG analysis and the capacity of the SDN controller to ensure acceptable network flow. We solve the knapsack problem using dynamic programming and perform the experiment simulations of conducting MTD. To be specific, we make the following **key contributions** in this work:

- We model the attackers' behaviors for selecting the next victim in a situation where attackers have a list of visible victim candidates on the network, and derive a random attack model that selects a victim randomly among the visible candidates and a weakest-first attack model that attacks the weakest victim first.
- Based on the attack behavior models, we compute topology-aware attack success probabilities for vulnerabilities on the network and incorporate the results into the BAGs for accurate security risk assessment instead of using a simple CVSS-based attack success probability for individual vulnerability.
- We formulate a knapsack problem to determine an optimal set of vulnerabilities to be reconfigured by MTD. The optimization aims to proactively protect as many vulnerable hosts as possible only if the MTD overhead does not exceed a certain amount of resources allocated for cybersecurity.

The remainder of this paper is organized as follows. Section II briefly reviews the related works. In Sections III and IV, a BAG analysis with the attacker's behavior models is described and evaluated. Section V describes the proposed optimal MTD strategy, and in Section VI, the simulation results are presented. The conclusion is presented in Section VII.

II. RELATED WORK

A. ADDRESS SHUFFLING MTD

The MTD is a proactive defense technique, which constantly changes the attack surface so that attackers cannot figure out the network configuration. There are various types of MTD techniques based on how the configuration is changed in the network [2].

- Time-based MTD: The configuration is periodically changed over time. The interval of the MTD reconfiguration is a controllable/adjustable parameter that can be determined by the security operator. The MTD interval can be fixed regardless of the existence of attacks or varied depending on a threat alert level. In general, under a full MTD, the configurations of all the devices are randomized at every MTD interval. While this reconfiguration is being performed, existing traffic flows can be interfered with.
- Event-based MTD: When an attack is detected, the MTD operation is triggered to defend the system from the following potential attacks. When event-based MTD is triggered and performed, the information exposed to the attacker is invalidated.
- Hybrid MTD: Time-based and event-based policies can be combined in practice to balance the tradeoff between security performance and system reconfiguration overhead.

The MTD can shuffle various properties of networked system configuration such as source/destination addresses of IP datagrams, port numbers of TCP/UDP segments, MAC addresses of data-link layer frames, virtual machine IP addresses, and so on. Address shuffling, called host/address mutation, is one of the most popular MTD techniques, which maps and changes the IP addresses in use to other IP addresses within a usable address space [7]. The address shuffling-based MTD can be readily implemented using software-defined networking (SDN) technology, which decouples the data and control planes and supports the programmability of packet routing and manipulation through the OpenFlow (OF) protocol [8], [9]. Jafarian et al. proposed to randomly mutate (i.e. shuffle) host IP addresses to make the address information unpredictable [8]. They devised a weighted mutation method based on the idea that an already scanned IP address is less likely to be scanned again compared to those that are not scanned yet. They set the weight of the IP address as the number of times that the IP address has been scanned by other users including attackers. The SDN controller chooses an unused IP address randomly based on the weight (i.e. selection probability) and changes the real IP address of a host as the chosen virtual IP address. Using the Mininet emulator, they showed their weighted mutation restrained the propagation of scanning worms. As the centralized SDN controller can be attacked or face system faults, Narantuya et al. proposed to use multiple SDN controllers for the IP shuffling-based MTD [10]. They showed that multiple controllers could shorten the IP shuffling delay and achieve low attack success probabilities via the experiment. Recently, Kim et al. proposed a deep reinforcement learning (DRL)-based traffic inspection and MTD countermeasure [11]. They adopted the DRL for finding the optimal traffic inspection points on a target network. Once the traffic flows are sampled, they are forwarded to an IDS for inspection. According to the inspection analysis of the IDS, the MTD is performed for the weakest assets to proactively protect them from attacks.

B. ATTACK GRAPH (AG)

Attack representation models such as ATs and AGs have been proposed for the identification of network nodes or services that intruders can easily access and exploit (i.e. attack paths). In detail, ATs represent a series of attack processes consisting of intermediate nodes required to achieve the ultimate goal of an attack, providing advantages of finding the security hardening methods [12], [13]. In ATs, the root node represents the attack goal to be accomplished by an attacker, and leaf nodes are various attack points to be exploited to compromise the root. If the conditions of child nodes are satisfied, the parent node is compromised. Each node in ATs is limited to having only one parent node in the hierarchical structure. On the other hand, AGs represent a series of attack paths starting from a root node of an initial condition in the graph to other nodes with various attributes such as system vulnerabilities, access privilege, or system/network properties [14], [15]. Although AGs are useful for analyzing the security risk using the causality of exploiting vulnerabilities, it also has a limitation of scalability in terms of network size.

In the literature, there are several approaches to overcoming this scalability problem in AGs. In [16], Noel and Jajodia proposed hierarchical aggregation of AGs to achieve scalability. They defined a set of aggregation rules for each level of the hierarchy based on the attack subgraph connectedness. Using their framework for AGs, both the computational and cognitive aspects of scalability were improved. In [17], Ou et al. proposed new logical AGs, which illustrate the system configuration information and logical dependencies of various attacks. In the logical AGs, the nodes represent the logical statements and the edges represent the dependency relation. They demonstrated that the logical AGs could reduce the topology of AGs up to a polynomial size. In [18], Yoon et al. proposed a three-tier AG (TAG), which consists of a network connectivity layer, a remote vulnerability layer, and a local vulnerability layer. By separating the layers depending on network topology and vulnerability connectivity, the TAGs provide a low complexity when generating AGs. In addition, they proposed to eliminate the cycles in the TAG to obtain a directed acyclic graph structure using the cycle handling method and the d-separation algorithm. It is easier to find the attack paths in a directed acyclic graph than in a graph with cycles.

C. BAYESIAN ATTACK GRAPH (BAG)

BAGs are AGs with Bayesian networks, which are directed acyclic attack graphs with probabilities assigned to edges [19]. The BAGs can be used to prevent potential damages caused by attackers by exploiting the likelihood that attacks occur at each node [20], [21]. In [22], Frigault et al. defined probability-based metrics with the CVSS scores of vulnerabilities and the dependencies of the exploits and introduced AGs constructed with a Bayesian network. Because the BAGs are grounded in probabilistic analysis, the security countermeasure decisions based on BAG are more reasonable than those based on the administrator's experience. Poolsappasit et al. modeled and utilized the BAGs for security risk assessment and mitigation [20]. Using the BAGs, they identified weak nodes in the network system and computed the change of attack success probabilities after attack incidents. In addition, they formulated an optimization problem with BAG analysis and solved it with a genetic algorithm, providing security mitigation plans. An example of mitigation plans includes applying a firewall, establishing an access control policy, patching the services, and disabling services. Miehling et al. utilized the BAGs for determining optimal countermeasure actions [21]. They introduced the existing BAG model and focused on modeling the actions of both attackers and defenders. They assumed that attackers move randomly in the BAGs and that the defenders know only the current status of the attackers. These assumptions made it possible to model the attacker's behavior as a probabilistic spreading process and to formulate a security defense problem as a partially observable Markov decision process (POMDP). The authors solved the POMDP with dynamic programming and presented the optimal security countermeasures consequently.

TABLE 1. Comparative analysis of MTD strategies with AGs.

Approach	Security Assessment	MTD Type	MTD Strategy
HARM-based MTD [3]	Hierarchical attack model	Shuffling, diversity, redundancy	A fixed portion selection
ACD using BAG [25]	Bayesian attack graph	Time/event based defense action	Reinforcement learning
DSEOM [26]	Centrality of attack graph	Address shuffling, container migration	Knapsack selection with QoS constraint
Asset criticality-aware MTD [18]	Three-tier attack graph	Time-based address shuffling on SDN	Probabilistic selection
DIVERGENCE [11]	IDS alert messages	Time-based address shuffling on SDN	Probabilistic selection
Ours	Bayesian attack graph	Time-based address shuffling on SDN	Knapsack selection with SDN overhead constraint

D. MTD STRATEGIES WITH AG

The AGs or BAGs would be used for deploying security countermeasures such as MTD and adaptive cyber defense (ACD) [23] techniques efficiently and appropriately. In [3], Hong and Kim assessed the effectiveness of various MTD techniques such as address shuffling, diversity, and redundancy using a hierarchical attack representation model called HARM, which consisted of the AG in the upper layer and the AT in the lower layer [24]. For a given MTD deployment scenario, HARM was exploited to evaluate the level of network hardening based on attack paths and to compute the importance measure based on network centrality for risk assessment. In [25], Hu et al. considered a reimaging-based ACD that replaced the images known to be compromised with clean ones. Because the defender was limitedly accessible to a subset of machines and their statuses, it was modeled as a POMDP. The interaction between the network and the attacker was modeled as BAG. They proposed online reinforcement learning-based algorithms to find effective defense policies by solving the POMDP problem. In [26], Jin et al. proposed the MTD strategies using the AGs in the container-based cloud environment. They identified the nodes frequently belonging to the attack paths in the AGs and proposed deploying the MTD techniques to those nodes to improve the effectiveness of the MTD technique. In [18], Yoon et al. proposed to apply the MTD with different shuffling intervals according to the hosts' criticality. They developed an AG model and identified the critical hosts, which are likely highly exploitable, by analyzing the AG. They implemented the mechanism for the MTD shuffling for both MAC and IP addresses and conducted the experiments in a real SDN testbed. However, implementing MTDs on SDN in existing work incurs message exchange overheads on the control plane because the routing rules are exchanged between the controller and switches to update the flow tables in each switch. It is crucial to optimize the operations of MTDs on SDN by using the BAG analysis results under the consideration of SDN flow table updating overhead that may affect the quality of service for legitimate traffic flows. The comparative analysis of MTD strategies with AGs is summarized in Table 1.

III. MODELING BAYESIAN ATTACK GRAPHS

A. RISK ASSESSMENT USING BAG

For security risk assessment, AGs can be generated using various attack graph generation tools [27] such as MulVAL [28] and CyGraph [29]. We consider a BAG for security risk assessment. A BAG is represented by $G = (V, E)$ where $v \in V$ is the index for a vulnerability and $e \in E$ is the directed edge representing relationships between vulnerabilities. Fig. 1 shows simple examples of BAG. The BAG in Fig. 1(a) has five vertices, where $V = \{1, 2, 3, 4, 5\}$ and $E = \{1 \to 3, 1 \to 4, 1 \to 5, 2 \to 5\}$. In a BAG, each vulnerability has the state of X, which has the value of one if the vulnerability is compromised by an attacker and zero if otherwise. Let $\mathbf{X} = (X_1, \cdots, X_n)$ denote a set of random variables for the states, where X_i is the random variable for the state of the i-th vulnerability. We model the states of nodes as Bernoulli random variable; X_i is one with a probability of p_{ji} and zero with a probability of $(1 - p_{ji})$, where p_{ji} is the attack success probability for the i-th vulnerability with the j-th parent vulnerability and is given by

TABLE 2. The parameters used for BAG modeling.

Parameter	Definition
$G = (V, E)$	Bayesian attack graph
v_i	The i-th vulnerability in G
$\mathbf{pa}_i$	Set of parent nodes of v_i in G
child(i)	Set of child nodes of v_i in G
X_i	Bernoulli random variable for the state of v_i
p_{v_i}	Vulnerability-specific attack success probability of v_i
f_{v_i}	The expected number of failures before the first success for v_i
p_{ji}	Topology-aware attack success probability for v_i with $v_j \in \mathbf{pa}_i$
$\check{p}_{ji}$	The modeled p_{ji} for the weakest-first attack behavior
$\hat{p}_{ji}$	The modeled p_{ji} for the random attack behavior
P_{X_i}	Unconditional attack success probability for v_i in G
$\Pr(\mathbf{X})$	Joint attack success probability for all nodes in G

$$p_{ji} = \Pr(X_i = 1 | X_j = 1) \text{ for } \exists j \in \mathbf{pa}_i, \tag{1}$$

where $\mathbf{pa}_i$ is a set of parent nodes of the i-th vulnerability in the BAG. The probability that the i-th vulnerability is compromised by an attacker is given by the summation of the joint probability function with $X_i = 1$ as follows:

$$\mathrm{P}_{X_i} := \Pr(X_i = 1) = \sum_{\{\mathbf{X} | X_i = 1\}} \Pr(\mathbf{X}). \tag{2}$$

In Bayesian networks, the joint probability for a set of random variables $\mathbf{X}$ can be obtained from conditional probabilities using the chain rule as follows:

$$\Pr(\mathbf{X}) = \prod_{i=1}^{n} \Pr(X_i | \mathbf{pa}_i). \tag{3}$$

For simplicity, we assume that a child node can be compromised when any of its parent nodes are compromised in the

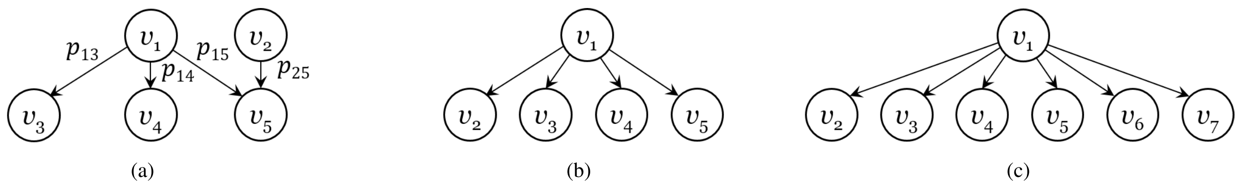


FIGURE 1. Examples of simple Bayesian attack graphs.

BAG. Under the assumption, $\Pr(X_i|\mathbf{pa}_i)$ in (3) can be written as follows:

$$\Pr(X_i = 1|X_j = 1, \forall j \in \mathbf{pa}_i) = 1 - \prod_{j\in\mathbf{pa}_i}(1 - p_{ji}). \quad (4)$$

Using (2) and (4), we can obtain P_X's for all the vulnerabilities in the BAG. The parameters used for BAG modeling are summarized in Table 2.

B. VULNERABILITY-SPECIFIC ATTACK SUCCESS PROBABILITIES

We consider how to obtain the attack success probabilities for the vulnerabilities in the BAG. Let p_{v_i} denote a vulnerability-specific attack success probability, which is solely dependent on the type of vulnerability. To perform risk assessment using BAG, the vulnerability-specific attack success probabilities of vulnerabilities in BAG should be estimated precisely [30]. Unfortunately, the estimation of attack success probabilities of vulnerabilities is challenging, and there is no exact way to determine the values yet. However, one may obtain the attack success probabilities using CVSS as follows:

- Frigault et al. [22] computed the attack success probability by dividing the CVSS's *Base Score* of the vulnerability by the maximum value of 10.
- Poolsappasit et al. [20] computed the attack success probability by dividing the *Exploitability* sub-score in the CVSS's *Base Score* by 10.
- Zhang et al. [31] computed the attack success probability with the exponential function, which is given by $p_{v_i} = 1 - e^{-(\text{exploitability score of } i\text{th vulnerability})}$ using *Exploitability* sub-score in the CVSS's *Base Score*.
- Homer et al. [32] computed the attack success probability using the values of the access complexity metric in the CVSS 2.0. The access complexity metric takes one of the low, medium, or high values according to the complexity of vulnerability exploits. Then, they mapped the values of the low, medium, and high to the attack success probabilities of 0.9, 0.6, and 0.2, respectively.

Alternatively, a measurement-based approach can be adopted [33]. Leversage et al. proposed using the concept of the number of attack trials until reaching attack success. The time-to-compromise (TTC) values can be empirically measured for each vulnerability and converted to the attack success probability for the corresponding vulnerability [33]. Most existing modeling methods estimate the attack success probability for a vulnerability simply by normalizing the exploitability of the vulnerability. Among the aforementioned approaches, we adopt the simplest one in [22], which uses the CVSS's Base Score. Thus, p_{v_i} is simply calculated as (CVSS's Base Score of v_i)/10.

Given a vulnerability-specific attack success probability, it is possible to compute the number of attempts for an attacker to compromise the vulnerability. Assume that an attack attempt for the i-th vulnerability is a Bernoulli trial with the success probability of p_{v_i} and the failure probability of $(1-p_{v_i})$. Using the geometric distribution, we can calculate the expected number of failures before the first success for the i-th vulnerability as follows:

$$f_{v_i} = \frac{1 - p_{v_i}}{p_{v_i}}. \quad (5)$$

For example, in Fig. 1(a), suppose that v_1 has been compromised. Then, the attacker starts to attack v_3, which has the vulnerability-specific attack success probability $p_{v_3} = p_{13}$ in Fig. 1(a). By (5), it is expected that the attacker experiences $(1 - p_{v_3})/p_{v_3}$ failures on average until it succeeds in compromising v_3. In other words, if the attacker has failed to attack f_{v_3} times, the next attack attempt is likely to succeed. In Fig. 1(a), v_1 has three reachable vulnerabilities, and it takes $(f_{v_3} + f_{v_4} + f_{v_5})$ for the attacker to compromise all the child nodes of v_1.

C. MODELING ATTACKER's BEHAVIORS

In general, AGs represent a series of attack paths from a root node to the child nodes. On an attack path, it is assumed that the attack is performed sequentially such that an attacker finds the next victim among the child nodes of a compromised parent node when it accomplishes an attack on the parent node. Under the assumption, we model the attack success probability in a generalized form of p_{ji} in (1), which is the attack success probability of the i-th vulnerability when the j-th vulnerability is its parent node in the BAG. Note that p_{ji} is dependent on both the i-th and j-th vulnerabilities. It is observed that p_{ji} can be affected by the degree of the j-th vulnerability in the BAG. For example, consider simple BAGs with the degree of four in Fig. 1(b) and that of six in Fig. 1(c). Once the attacker has exploited vulnerability v_1, the attacker selects one of the child nodes of v_1 as the next victim and starts to attempt to compromise it. If the degree of the parent node is higher, it takes a longer time on average until a specific vulnerability is attacked as the victim node because there are many other candidates to be selected as the

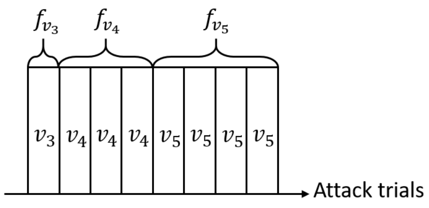


FIGURE 2. An example of attacker's behavior under the weakest-first attack model in the BAG of Fig. 1(a) in case of $f_{v_3} < f_{v_4} < f_{v_5}$.

next victim. In Fig. 1(b), as v_1 has 4 child nodes, there is a possibility that a child is not attacked until the other 3 nodes are compromised. In Fig. 1(c), as v_1 has 6 child nodes, it may happen that a child is attacked after the other 5 child nodes are compromised. It implies that the degree of a parent node affects the time for a child node to be scanned and attacked in the given topology. Hence, for computing the attack success probability of a child vulnerability, it is essential to consider how long it takes for an attacker to scan and start an attack on a child node in the given topology. However, existing literature does not consider topological properties such as the number of other vulnerabilities that are reachable from a vulnerability in BAG.

In addition to the degree of vulnerabilities, the scanning behavior of attackers affects the attack success probabilities of the vulnerabilities. In Fig. 1(b), suppose $p_{v_i} > p_{v_j}$ for all i and j such that $i < j$ without loss of generality. In this case, the attacker that has compromised v_1 may randomly select one of the child nodes of v_1 as the next victim. If the vulnerability-specific attack success probabilities are exposed to the attacker through a scanning attack, it may start to attack the weakest vulnerability with a higher vulnerability-specific attack success probability. In Fig. 1(b), the attacker in v_1 is likely to compromise v_2 first because p_{v_2} is higher than others. Therefore, for an accurate security risk assessment, it is necessary to consider the attacker's behavior and incorporate it into the BAG model.

We consider the most intuitive attacker behaviors of a weakest-first attack and a random attack. According to the cyber kill chain (CKC) [34], real-life attacks are performed in seven stages: reconnaissance, weaponization, delivery, exploitation, installation, command and control (C2), and actions on objectives. In the stage of reconnaissance, attackers scan the network and search for potential targets. Once they secure a list of potential victims, they can randomly choose one victim among the potential targets and move on to the next stage of weaponization. The random selection is modeled as **random attack behavior**. On the other hand, the attackers could exploit the vulnerability information of potential targets for selecting the next victim. The attackers may have various preferences for the victim selection. For example, if a specific malware tool is already ready in the weaponization stage, the attacker may choose a victim that is easily susceptible to the tool. In the CKC, after the reconnaissance step, the attackers have the vulnerabilities of potential targets and can decide to move on to the next stage for the most vulnerable target. This preference is modeled as **weakest-first attack behavior**. We derive the analytical models for the attack behaviors and use the results for BAG-based security assessment analysis.

1) WEAKEST-FIRST ATTACK BEHAVIOR

Attack scenarios where attackers scan a victim network and identify the vulnerabilities of the victim network are considered here. For the weakest-first attack behavior, the attackers attempt to compromise the most vulnerable node at first among the vulnerabilities of the victim network. Fig. 2 shows an example of the weakest-first attack behavior for the BAG in Fig. 1(a). In Fig. 2, each rectangle represents an attack trial on the labeled vulnerability. Suppose that v_3 is the most vulnerable and v_5 is the least vulnerable among the child nodes of v_1 in the BAG in Fig. 1(a), (i.e. $p_{v_3} > p_{v_4} > p_{v_5}$). Then, the attacker in the weakest-first attack model compromises the child nodes in the order of v_3, v_4, and v_5 as shown in Fig. 2. The number of trials for each vulnerability is computed by (5) for a given vulnerability-specific attack success probability. It confirms that the topology of BAGs affects the average time for attackers to compromise a specific node.

Proposition 1: Let $\tilde{p}_{ji}$ denote the topology-aware attack success probability for the i-th vulnerability with the j-th parent vulnerability. For the weakest-first attack behavior, $\tilde{p}_{ji}$ is given by

$$\tilde{p}_{ji} = \frac{1}{1 + \sum_{k \in child(j), p_{v_k} \geq p_{v_i}} f_{v_k}}, \tag{6}$$

where the child(j) represents the child nodes of the node v_j in the BAG.

Proof: For the weakest-first attack behavior, a child node v_i is attacked after the other child nodes with greater vulnerability-specific probabilities than v_i are compromised. As a result, the number of attack trials that the attacker should perform to compromise v_i is given by $\sum_{k \in \text{child}(j), p_{v_k} \geq p_{v_i}} f_{v_k}$. The equivalent attack success probability $\tilde{p}_{ji}$ is obtained by $\frac{1-\tilde{p}_{ji}}{\tilde{p}_{ji}} = \sum_{k \in \text{child}(j), p_{v_k} \geq p_{v_i}} f_{v_k}$ using (5), and it can be written in (6).

Suppose that the vulnerability-specific attack success probabilities for nodes v_3, v_4, and v_5 are 0.5, 0.25, and 0.2, respectively, in the example shown in Fig. 2. Then, the expected number of attack failures for each node can be calculated as $f_{v_3} = 1, f_{v_4} = 3$, and $f_{v_5} = 4$. Using (6), $\tilde{p}_{13}, \tilde{p}_{14}$, and $\tilde{p}_{15}$ become 0.5, 0.2, and 0.11, respectively. The topology-aware attack success probability of the most vulnerable node v_3 does not change, but that of the least vulnerable node v_5 becomes much smaller because the attacker is assumed to have the preference to attack the weakest node first under the weakest-first attack model. It implies that the model reflects the attacker's preference that selects the weakest vulnerability as its first victim. However, the attacker may not attempt to compromise all the vulnerabilities exposed to the attacker one by one. Once one or a few vulnerabilities are compromised,

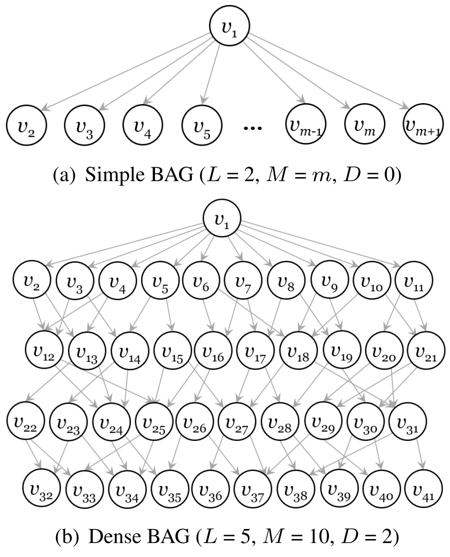

v_1
v_2 v_3 v_4 v_5 ... v_{m-1} v_m v_{m+1}
(a) Simple BAG (L = 2, M = m, D = 0)
v_1 … v_{41}
(b) Dense BAG (L = 5, M = 10, D = 2)

FIGURE 3. Topology of multi-tree structured BAGs, which have *L* layers, *M* nodes per layer, and *D* edges per node.

it can attempt to compromise the next vulnerabilities that are newly accessible through the compromised vulnerabilities. In this case, the analytic result is expected to reside between the result without attack behavior and that of the weakest-first attack behavior. In addition, as the attacker would keep searching every vulnerability until it succeeds in reaching its final target, the model is derived under the assumption that the attacker performs the searching of every vulnerability in the ascending order of compromisation difficulty.

2) RANDOM ATTACK BEHAVIOR

Under the random attack behavior, an attacker randomly selects a victim node and attempts to compromise it. We assume that this victim selection procedure is repeated until there are no more vulnerabilities left for an attack. The random attack model is also suitable in situations where the attackers cannot estimate the vulnerability-specific attack success probabilities of the candidate nodes. Consider the BAG in Fig. 1(a) under the assumption that an attacker has compromised the node v_1. It is observed that there are six random cases of the victim sequences, i.e. v_3–v_4–v_5, v_3–v_5–v_4, v_4–v_3–v_5, v_4–v_5–v_3, v_5–v_3–v_4, and v_5–v_4–v_3. The case occurs with the same possibility of 1/6.

Proposition 2: Under the random attack model, the topology-aware attack success probability $\hat{p}_{ji}$ is given by

$$\hat{p}_{ji} = \frac{1}{1 + f_{v_i} + \frac{1}{2}\sum_{k\in child(j),k\neq i} f_{v_k}}, \quad (7)$$

where the child(j) represents the child nodes of the node v_j in the BAG.

Proof: For the random attack behavior, a victim node is randomly selected among the child nodes of a parent node. For the i-th victim node, any sibling nodes can be randomly selected before the i-th node and after the i-th node with the same probability of 1/2. If a sibling node precedes the i-th node, the attack on the i-th node is postponed by the amount of time required to compromise the sibling node. As a result, the total expected time until the attacker starts to compromise the i-th node is given by $(\frac{1}{2}\sum_{k\in\text{child}(j),k\neq i} f_{v_k})$ on average. Using (5), the equivalent attack success probability $\hat{p}_{ji}$ is obtained by $\frac{1-\hat{p}_{ji}}{\hat{p}_{ji}} = (f_{v_i} + \frac{1}{2}\sum_{k\in\text{child}(j),k\neq i} f_{v_k})$, and it can be written in (7).

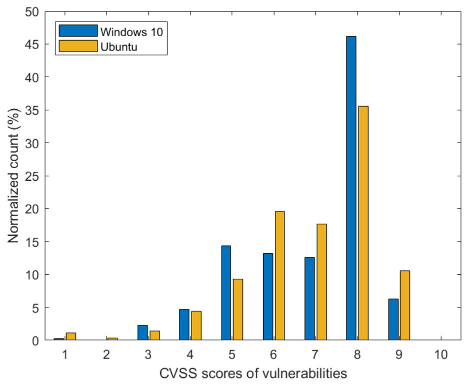


FIGURE 4. The distribution of the OS-related vulnerability.

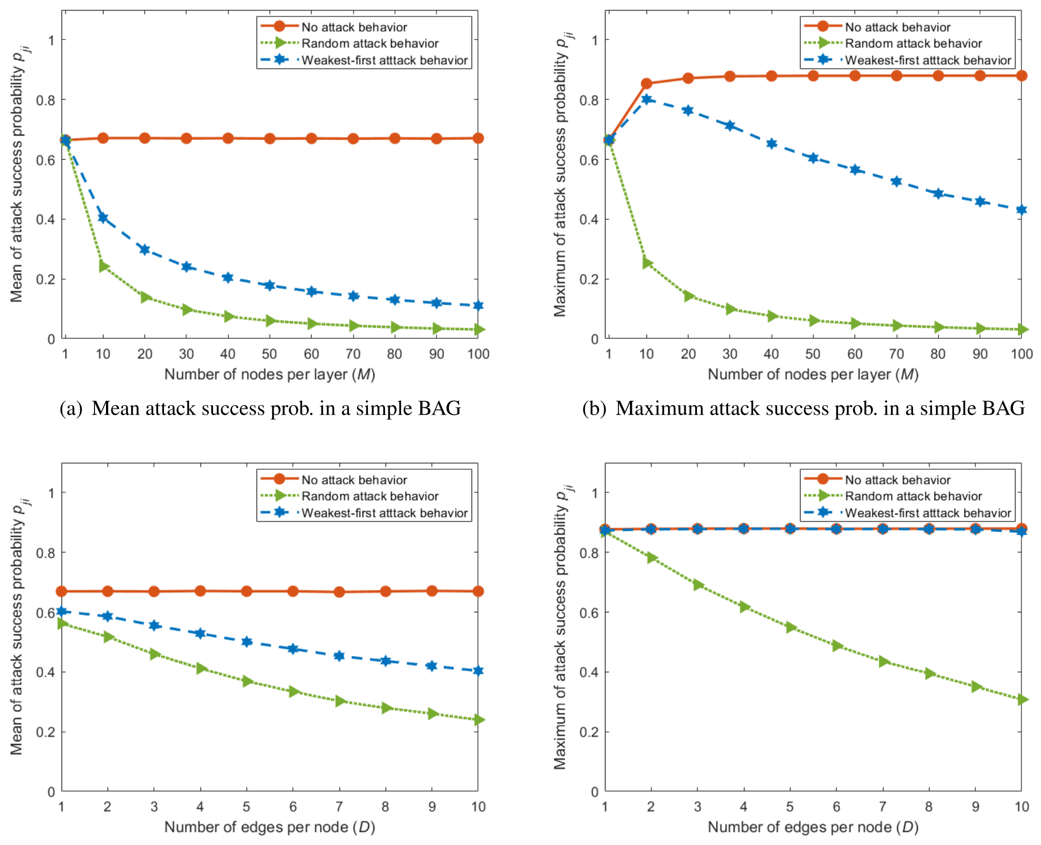
IV. ATTACK BEHAVIOR MODEL VALIDATION EXPERIMENT

A. SIMULATION ENVIRONMENT

To validate our attack models, we have performed the experimental simulations in simple and dense BAGs as shown in Fig. 3 and measured the mean and maximum values of unconditional attack success probabilities P_{X_i} for the nodes in BAGs. We compared the results for the case with no consideration of the attacker's behavior, the random attack behavior, and the weakest-first attack behavior while changing the number of nodes per layer M in the simple BAG of Fig. 3(a) and the number of edges per node D in the dense BAG of Fig. 3(b). In these BAGs, the root node v_1 is connected with every node in the second layer by default, and the prior probability $\Pr(X_1 = 1)$ for v_1 is given as 0.5. The probabilities p_{v_i} for the other vulnerabilities from v_2 to v_n follow a distribution obtained from a real-life dataset.

B. REAL-LIFE DATASET FOR VULNERABILITY-SPECIFIC ATTACK SUCCESS PROBABILITY

We exploit a CVSS score distribution of vulnerabilities related to Windows 10 and Ubuntu to determine the vulnerability-specific attack success probabilities. First, we searched the CVE IDs of the OS-related vulnerabilities in the CVE database. The CVE IDs were crawled from the CVE website, which is accessible at 'https://cve.mitre.org.' Specifically, we used the following URL for obtaining the vulnerabilities related to MS Windows 10: 'https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=windows+10.' Then, we obtained the score of vulnerabilities corresponding to these CVE IDs in the CVSS database, which is accessible at



(a) Mean attack success prob. in a simple BAG

(b) Maximum attack success prob. in a simple BAG

(c) Mean attack success prob. in a dense BAG

(d) Maximum success prob. in a dense BAG

FIGURE 5. Topology-aware attack success probability in the multi-tree structured BAG of Fig. 3(a) and (b).

'https://nvd.nist.gov.' For example, we obtained the vulnerability score for the CVE ID of CVE-2022-42973 at the following URL: 'https://nvd.nist.gov/vuln/detail/CVE-2022-42973.' For each score in the range of [1, 10], we counted the vulnerabilities with the corresponding score value in the list of CVE IDs and then normalized them by the total number of vulnerabilities. Fig. 4 shows that most vulnerabilities of both Windows 10 and Ubuntu are scored in the range of 5 to 9. Both cases have a similar distribution and it was observed that there was a peak at the score of eight in the distribution. Using the distribution obtained from the CVSS score, we generate the vulnerability-specific attack success probabilities randomly for constructing BAGs in the experiments.

C. MEAN AND MAXIMUM ATTACK SUCCESS PROBABILITIES

Fig. 5(a) and (b) show the mean and maximum attack success probabilities in the simple BAG in Fig. 3. In the simple topology, M varies from 1 to 100 while $L = 2$ and $D = 0$, and thus the total number of edges in the BAG is given by M. Here, M vulnerability-specific attack success probabilities follow the distribution for Ubuntu vulnerabilities obtained in Section IV-B. The results for the case with no consideration of the attacker's behavior (denoted by 'no attack behavior') are plotted in red color in Fig. 5(a) and (b). In Fig. 5(a), it was almost a constant of about 0.67 because the distribution for Ubuntu has a mean of about 6.7, and the vulnerability-specific attack success probabilities are obtained by a normalization factor of 1/10. In Fig. 5(b), it was about 0.67 for M=1 and 0.88 for the other M's. Note that in the case of M=1, the BAG has one edge, and thus the mean and maximum values are the same. For both random and weakest-first attack behaviors, the mean and maximum attack success probabilities decrease exponentially with respect to M. It is because each vulnerability node has more sibling nodes, and the attacker has more chances to attack the other vulnerabilities. The weakest-first attack behavior gives a larger mean and maximum of attack success probabilities than the random attack behavior. This result implies that the weakest-first attack strategy of attackers is more effective to compromise the victim network.

Fig. 5(c) and (d) show the mean and maximum attack success probabilities in the dense BAG in Fig. 3. In the dense topology, D varies from 1 to 10 while L=5 and M=10, and the total number of edges in the BAG is given by $M(1+D(L-2))$. In Fig. 5(c), the mean attack success probabilities for the random and weakest-first attack behaviors decrease gradually with respect to D. If each node has more edges, the attacker has more options to select the attack paths. As a result, each

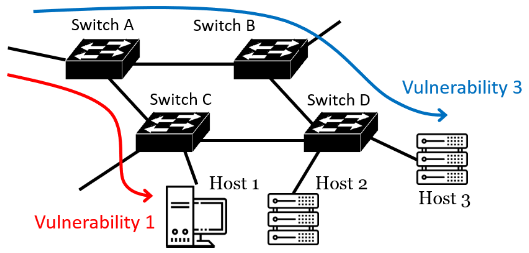


FIGURE 6. An example of the proposed time-based MTD system.

TABLE 3. The parameters used for the MTD algorithm and experiment.

Parameter	Definition
L	Number of layers in G
M	Number of nodes per layer in G
D	Number of edges per node in G
t_{MTD}	MTD time interval that the MTD operation is triggered
x_i	The binary decision value for the MTD operation for v_i
n	Total number of nodes in G
Q	Flow table updating capacity of the SDN controller within t_{MTD}
D_i	Flow table updating overhead for OF-enabled switches caused by v_i

vulnerability would have a smaller attack success probability. Fig. 5(d) shows the maximum attack success probabilities in the dense BAG. Interestingly, the weakest-first attack behavior gives the same value as that with the legend of 'no attack behavior'. It implies that the maximum values in both cases do not change. This is because, under the weakest-first behavior, the attacker starts to attack the weakest vulnerability first, and thus the weakest vulnerability would have the largest attack success probability. On the contrary, under the random attack behavior, the attack success probability with every node decreases as the number of edges increases. As a result, the average of the maximum attack success probability decreases with respect to D. In summary, the proposed random and weakest-first attack models were developed to describe the behaviors of attackers according to the attacker's strategies, and the models were validated by observing the trend of topology-aware attack success probabilities with respect to the BAG parameters in simple and dense BAG topologies.

V. PROPOSED MOVING TARGET DEFENSE SYSTEM

We propose a time-based MTD system in the SDN environment. In the SDN environment, the address shuffling-based MTD technique is conducted by exchanging flow table rules between the SDN controller and all switches in the network using OF protocol. The SDN controller can modify the IP address information of packets using the modification packet-handling action for the address shuffling and can change a packet forwarding path for the packets with the shuffled source/destination addresses using the forwarding action on the SDN flow tables. We formulate an optimization problem to determine a set of vulnerabilities for which the addresses are shuffled. Based on the BAG-based analysis, the optimization finds the solution to maximize the sum of the attack success probabilities for the selected vulnerabilities to focus on the protection of weak vulnerabilities with a high attack success probability. Because the MTD operations on SDN incur communication overhead between the SDN controller and OF-enabled switches on the control plane, and the flow tables for MTD may excessively occupy the memory space of the switches, it is desirable to impose a constraint on the number of the SDN flow table exchanges in the optimization. This optimization is in the form of a 0/1 knapsack problem and can be solved by using dynamic programming. Note that the knapsack problem is to determine a set of items to be included in a collection such that the total value of the selected items is maximized while satisfying the constraint of the collection weight. The set of items in the knapsack problem corresponds to the set of vulnerabilities to be reconfigured by the MTD operations, and the constraint of the collection weight corresponds to the SDN overhead constraint for flow table updating.

Fig. 6 shows a simple network in which the proposed MTD technique is deployed. The network consists of three hosts and four switches, and the red and blue lines represent the current network flows toward host 1 with vulnerability 1 and host 3 with vulnerability 3, respectively. For the given network, the BAG analysis is performed and P_{X_i}, which is the unconditional probability for an attacker to compromise the i-th vulnerability, is obtained for $i \in V$ of the network. Here, it is worth noting that the number of SDN flow table exchanges required to perform the address shuffling for the two vulnerabilities is different. To perform the MTD for vulnerabilities 1 and 3, the flow tables of two and three OF-enabled switches are to be updated by the SDN controller, respectively. While the traditional full MTD technique shuffles the addresses of all hosts in the network, the proposed MTD shuffles the addresses of a subset of hosts that are determined by solving an optimization problem under the consideration of an unconditional attack success probability of vulnerabilities and the SDN control overhead required to perform MTD for security purposes.

A. OPTIMIZATION FORMULATION

The proposed MTD system aims to apply address shuffling to as many vulnerable hosts as possible only if the MTD overhead does not exceed a certain amount of SDN resources allocated for cybersecurity. We formulate the following maximization optimization problem:

$$\begin{aligned} &\max \sum_{i=1}^{n} (\mathrm{P}_{X_i} \cdot x_i), \\ &\text{subject to} \sum_{i=1}^{n} D_i \cdot x_i \leq Q, \\ &\qquad x_i \in \{0, 1\}, 1 \leq i \leq n, \end{aligned} \tag{8}$$

where D_i is the flow table updating overhead for OF-enabled switches that the flows destined to a host with the i-th vulnerability go through, and Q is the flow table updating capacity of the SDN controller within an MTD time interval. Here,

Algorithm 1 Algorithm for Solving a 0/1 Knapsack Problem

```
Input: $n$, $Q$, $[P_{X_1}, \cdots, P_{X_n}]$, $[D_1, \cdots, D_n]$
Output: $[x_1, \cdots, x_n]$
1:  Make an array $m[0..n, 0..Q]$.
2:  for $i \leftarrow 1$ to $n$ do
3:      $x_i \leftarrow 0$
4:  end for
5:  for $i \leftarrow 0$ to $n$ do
6:      $m[i, 0] \leftarrow 0$
7:  end for
8:  for $j \leftarrow 0$ to $Q$ do
9:      $m[0, j] \leftarrow 0$
10: end for
11: for $i \leftarrow 1$ to $n$ do
12:     for $j \leftarrow 1$ to $Q$ do
13:         if $D_i > j$ then
14:         $m[i, j] \leftarrow m[i-1, j]$
15:         else
16:         $m[i, j] \leftarrow \max\{m[i-1, j], m[i-1, j-D_i] + P_{X_i}\}$
17:         end if
18:     end for
19: end for
20: Let $i = n$ and $j = Q$
21: while $i > 0$ and $j > 0$ do
22:     if $m[i, j] \neq m[i-1, j]$ then
23:         $x_i = 1$ {mark the $i$-th vulnerability}
24:         $j \leftarrow j - D_i$ and $i \leftarrow i - 1$
25:     else
26:         $i \leftarrow i - 1$
27:     end if
28: end while
```

D_i is measured by the number of OF messages to be sent to OF-enabled switches. For example, D_1=2 and D_3=3 in Fig. 6. Q is the maximum amount of resources reserved for the MTD operation, depending on the available resources of the SDN controller. In (8), P_{X_i} is the unconditional probability for an attacker to compromise the i-th vulnerability and is obtained by the BAG analysis under the assumption that the attackers use either the random or weakest-first attack strategy. The simulation parameters are listed in Table 3.

B. DYNAMIC PROGRAMMING FOR SOLVING THE PROBLEM

The optimization of (8) is a form of the 0/1 knapsack problem where items to be put in the knapsack correspond to vulnerabilities that require MTD. This optimization problem is solved using dynamic programming. Algorithm 1 shows the pseudocode of the dynamic programming for solving the 0/1 knapsack problem. The inputs of the algorithm are n, Q, P_{X_i}'s, and D_i's for $i \in V$. Here, n is the total number of vulnerabilities in the BAG, and Q is the maximum number of resources of the SDN controller for MTD. P_{X_i}'s are the unconditional probabilities of vulnerabilities for $i \in V$. D_i's are the amount

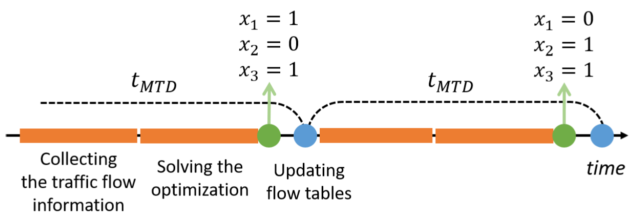


FIGURE 7. An example of the operation of the proposed MTD in the network of Fig. 6.

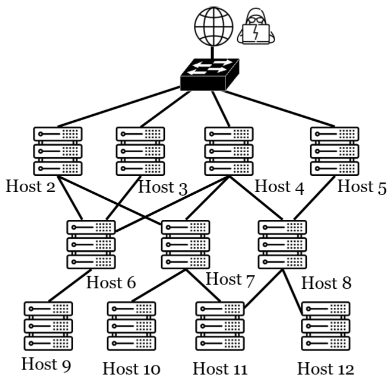
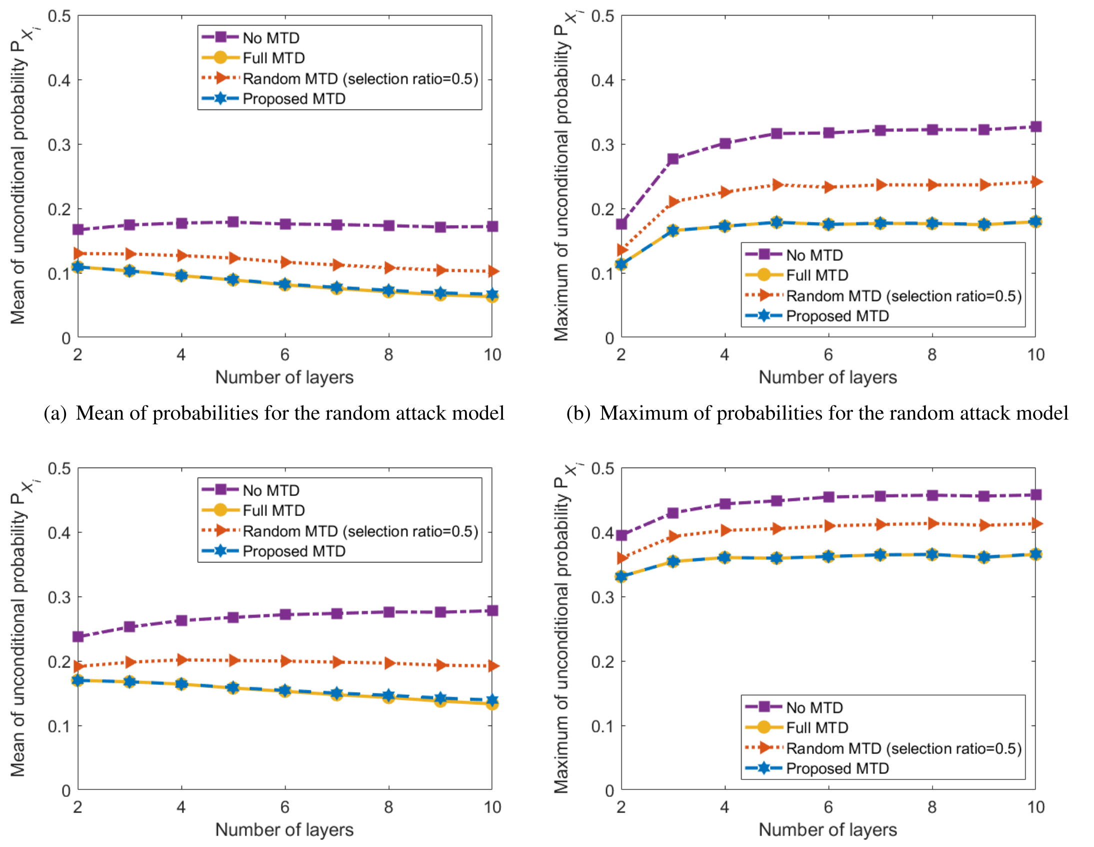


FIGURE 8. An example of network topology for MTD simulations ($L = 4$).

of overhead depending on the current network flow paths when applying the MTD to a host with the i-th vulnerability for $i \in V$. Note that P_{X_i} and D_i correspond to the value and weight of the i-th item in a knapsack problem, respectively. The array x_i's for $i \in V$ is the binary decision output vector of the algorithm. In lines 2-8, we initialize the elements of x_i's and m to zero. Then, we recursively calculate m by the dynamic programming of a bottom-up approach in lines 9-17. In lines 19-26, we inspect $m[i, j]$ to find and mark x_i's for the vulnerabilities that require MTD. According to x_i's, the SDN controller identifies the intermediate OF switches forwarding the network flows destined for the hosts with the marked vulnerabilities and sends flow table modification rules to the switches. In addition, the complexity of Algorithm 1 is given by $\mathcal{O}(n \cdot Q)$ because it needs $n \cdot Q$ iteration to update m. This is more feasible than the complexity $\mathcal{O}(2^n)$ of the brute force method for solving the 0/1 knapsack problem.

C. OPERATION OF THE PROPOSED MTD

Fig. 7 shows an operation example of the proposed time-based MTD in the network of Fig. 6. Here, t_{MTD} is the time interval of MTD and determines how frequently the MTD operation is performed. At every t_{MTD} time, the time-based MTD operation is triggered. If an attack threat level is high, the MTD interval needs to be set to a small value. However, a small interval value causes the MTD operations to be triggered too frequently. In this case, too much network resource might be abused for network reconfiguration, and thus the existing traffic flows can be interfered with.



(a) Mean of probabilities for the random attack model

(b) Maximum of probabilities for the random attack model

(c) Mean of probabilities for the weakest-first attack model

(d) Maximum of probabilities for the weakest-first attack model

FIGURE 9. Unconditional attack success probability P_{X_i}'s for the three MTD strategies.

In Fig. 7, the SDN controller sets up a timer for the MTD interval, and when the timer expires, it performs the operations for collecting the traffic flow information, solving the optimization, and updating flow tables over the OF protocol. Algorithm 1 gives a binary decision vector of $[x_1, \cdots, x_n]$. If $x_i = 1$, then the address of the host with the i-th vulnerability is changed randomly, and the SDN flow tables for the traffic flows destined to the i-th vulnerability are updated accordingly by the SDN controller when the MTD is triggered. The total number of flow table updates on the SDN is $\sum_{i=1}^{n} D_i \cdot x_i$. Thus, t_{MTD} should be set large enough to perform collecting the traffic flow information, solving the optimization, and updating the SDN flow tables. If there is any change in network and device configuration, vulnerability scanning and BAG-based risk assessment should be reperformed. Unlike existing MTD methods that change the configuration of all the hosts on the network, the proposed MTD selects a subset of hosts using the decision vector of (8) and changes their configuration. Because the number of hosts whose configurations are shuffled is smaller than the other MTDs, t_{MTD} can be set smaller.

As an example, in Fig. 7, the first binary decision vector is [1, 0, 1]. Then the addresses of Host 1 and 3 are updated, and the flow tables on Switch A and C for the flow destined for Host 1 and Switch A, B, and D for the flow destined for Host 3 are updated. For t_{MTD}, the flow information collecting and the optimization solving are performed. As the result, the decision vector is [0, 1, 1], and the address shuffling is performed again.

VI. PERFORMANCE EVALUATION

To evaluate the performance of the proposed MTD system, we performed the simulation for random network topologies with multiple layers. The random topology networks have multiple layers from 2 to 10, each layer with a random number of nodes between 3 and 10, and each node has a random number of edges between 1 and 5. Fig. 8 shows an example of the network topology generated by setting the number of layers as 4. We assume that each host has only one vulnerability related to Ubuntu OS and the attackers can access this network externally with a probability of 0.5. Note that the attack success probability of the top node in BAG is dependent on the security level and policy of the gateway for external network access. In addition, the attacker always starts to attack the victim nodes from the top layer of the network and can generate either one network flow passing through the shortest path or multiple flows passing through all possible paths. As a result, the number of flow tables D_i to be updated

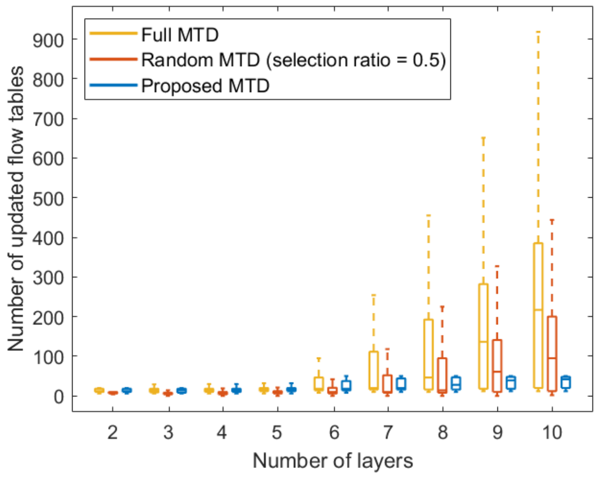


FIGURE 10. Overhead comparison with the number of updated flow tables.

for the i-th node is set as the random value between the hop count of the shortest path to the i-th node and the number of all nodes in the upper layer of the i-th node. The capacity Q of the SDN controller for the flow table updating is set as 50. We performed 1,000 simulations, and the average values for the overhead and performance measurements were reported for our MTD system. The performance of the proposed MTD is compared with that of a full MTD and a random MTD with a fixed selection ratio. The full MTD shuffles the addresses of all hosts in the network like traditional MTD methods. The random MTD with a fixed selection ratio shuffles the addresses of hosts randomly chosen with the ratio.

Fig. 9 shows the mean and maximum of the unconditional probabilities P_{X_i}'s for different MTD strategies under the assumption that the attackers use either random or weakest-first attack strategy. In Fig. 9(a) and (b), the full MTD gives a much lower mean and maximum of unconditional attack success probabilities than the original network without MTD. Under the MTD, the addresses of hosts change randomly, but the attackers cannot distinguish whether the hosts are new ones or the existing hosts just changed their addresses. As a result, the MTD makes the attackers believe that there exist more hosts with different addresses. This is how the MTD can protect the network proactively against cybersecurity threats. In Fig. 9(a), the mean of the probabilities decreases with respect to L because the number of nodes increases, and each node has a lower chance of being attacked. On the contrary, in Fig. 9(b), the maximum increases and levels off after L=5. If the shuffling rate is reduced by half, the mean and maximum of the probabilities increase accordingly. In comparison, the proposed MTD achieves the same mean and maximum results as that of the full MTD. Fig. 9(c) and (d) show the performance result under the weakest-first attack model. They show similar trends to those in Fig. 9(a) and (b), respectively. The only difference is that the magnitude of the mean and maximum results is larger than that in Fig. 9(a) and (b). It implies that the weakest-first attack strategy is more effective to compromise the vulnerabilities in the same victim network. Fig. 10 shows the box plot for the number of flow table updatings for conducting MTD techniques under the weakest-first attack behavior. It was observed that the MTD overhead for the random attack behavior, which is the number of flow table updatings, was almost the same as that for the weakest-first attack behavior. The full MTD has the largest MTD overheads in terms of the number of updated flow tables in the entire range of L because the full MTD changes the addresses of all nodes in the network. For the random MTD, the overheads are the smallest in the range of L from 2 to 5. However, when the number of layers exceeds 6, the proposed MTD achieves the smallest MTD overheads, which is lower than 50. Note that the SDN capacity is set as 50 in this simulation. These simulation results indicate that the proposed MTD has almost the same MTD performance as the full MTD while the MTD overhead is kept low.

VII. CONCLUSION

In this paper, we have proposed a time-based MTD strategy for the SDN environment that considers the security risk level and MTD performing overhead. The proposed MTD strategy shuffles the addresses of an optimal set of vulnerable hosts rather than the entire hosts on the network. To determine the set of vulnerable hosts, we have proposed a BAG-based risk analysis that considers the random and weakest-first attack behaviors and solved the 0/1 knapsack problem for selecting vulnerable hosts under an overhead constraint using dynamic programming. The simulation results indicated that the proposed BAG-based MTD strategy can achieve almost the same performance as the full MTD strategy while retaining low MTD overhead.

REFERENCES

[1] E. Cole, *Advanced Persistent Threat: Understanding the Danger and How to Protect Your Organization*. Waltham, MA, USA: Syngress, 2012.

[2] J.-H. Cho, D. P. Sharma, H. Alavizadeh, S. Yoon, N. Ben-Asher, T. J. Moore, D. S. Kim, H. Lim, and F. F. Nelson, "Toward proactive, adaptive defense: A survey on moving target defense," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 709–745, 1st Quart., 2020.

[3] J. B. Hong and D. S. Kim, "Assessing the effectiveness of moving target defenses using security models," *IEEE Trans. Depend. Secure Comput.*, vol. 13, no. 2, pp. 163–177, Mar. 2016.

[4] *Common Vulnerability Scoring System (CVSS)*. Accessed: Jan. 13, 2022. [Online]. Available: https://www.first.org/cvss/

[5] G.-Y. Shin, S.-S. Hong, J.-S. Lee, I.-S. Han, H.-K. Kim, and H.-R. Oh, "Network security node-edge scoring system using attack graph based on vulnerability correlation," *Appl. Sci.*, vol. 12, no. 14, p. 6852, Jul. 2022.

[6] J. B. Hong and D.-S. Kim, "HARMs: Hierarchical attack representation models for network security analysis," in *Proc. Austral. Inform. Secur. Manage. Conf.*, 2012, pp. 74–81.

[7] T. E. Carroll, M. Crouse, E. W. Fulp, and K. S. Berenhaut, "Analysis of network address shuffling as a moving target defense," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2014, pp. 701–706.

[8] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Openflow random host mutation: Transparent moving target defense using software defined networking," in *ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, 2012, pp. 127–132.

[9] D. P. Sharma, D. S. Kim, S. Yoon, H. Lim, J. Cho, and T. J. Moore, "FRVM: Flexible random virtual IP multiplexing in software-defined networks," in *Proc. IEEE Int. Conf. Trust, Secur. Privacy In Comput. Commun. (TrustCom)*, Aug. 2018, pp. 579–587.

[10] J. Narantuya, S. Yoon, H. Lim, J.-H. Cho, D. S. Kim, T. Moore, and F. Nelson, "SDN-based IP shuffling moving target defense with multiple SDN controllers," in *Proc. Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.-Supplemental volume (DSN-S)*, Jun. 2019, pp. 15–16.

[11] S. Kim, S. Yoon, J.-H. Cho, D. S. Kim, T. J. Moore, F. Free-Nelson, and H. Lim, "DIVERGENCE: Deep reinforcement learning-based adaptive traffic inspection and moving target defense countermeasure framework," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 4, pp. 4834–4846, Dec. 2022.

[12] S. Mauw and M. Oostdijk, "Foundations of attack trees," in *Proc. Int. Conf. Inf. Secur. Cryptol.*, 2005, pp. 186–198.

[13] R. Dewri, N. Poolsappasit, I. Ray, and D. Whitley, "Optimal security hardening using multi-objective optimization on attack tree models of networks," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, Oct. 2007, pp. 204–213.

[14] K. Ingols, R. Lippmann, and K. Piwowarski, "Practical attack graph generation for network defense," in *Proc. 22nd Annu. Comput. Secur. Appl. Conf. (ACSAC)*, Dec. 2006, pp. 121–130.

[15] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, "Automated generation and analysis of attack graphs," in *Proc. IEEE Symp. Secur. Privacy*, May 2002, pp. 273–284.

[16] S. Noel and S. Jajodia, "Managing attack graph complexity through visual hierarchical aggregation," in *Proc. ACM workshop Visualizat. Data Mining Comput. Secur.*, Oct. 2004, pp. 109–118.

[17] X. Ou, W. F. Boyer, and M. A. McQueen, "A scalable approach to attack graph generation," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, Oct. 2006, pp. 336–345.

[18] S. Yoon, J.-H. Cho, D. S. Kim, T. J. Moore, F. Free-Nelson, and H. Lim, "Attack graph-based moving target defense in software-defined networks," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 3, pp. 1653–1668, Sep. 2020.

[19] Y. Liu and H. Man, "Network vulnerability assessment using Bayesian networks," *Proc. SPIE*, vol. 5812, pp. 61–71, Mar. 2005.

[20] N. Poolsappasit, R. Dewri, and I. Ray, "Dynamic security risk management using Bayesian attack graphs," *IEEE Trans. Depend. Secure Comput.*, vol. 9, no. 1, pp. 61–74, Jan. 2012.

[21] E. Miehling, M. Rasouli, and D. Teneketzis, "Optimal defense policies for partially observable spreading processes on Bayesian attack graphs," in *Proc. 2nd ACM Workshop Moving Target Defense*, Oct. 2015, pp. 67–76.

[22] M. Frigault, L. Wang, A. Singhal, and S. Jajodia, "Measuring network security using dynamic Bayesian network," in *Proc. 4th ACM Workshop Quality Protection*, Oct. 2008, pp. 23–30.

[23] G. Cybenko, S. Jajodia, M. P. Wellman, and P. Liu, "Adversarial and uncertain reasoning for adaptive cyber defense: Building the scientific foundation," in *Information Systems Security*. Cham, Switzerland: Springer, 2014, pp. 1–8.

[24] J. B. Hong and D. S. Kim, "Performance analysis of scalable attack representation models," in *Proc. IFIP Int. Inform. Secur. Conf.*, 2013, pp. 330–343.

[25] Z. Hu, M. Zhu, and P. Liu, "Online algorithms for adaptive cyber defense on Bayesian attack graphs," in *Proc. Workshop Moving Target Defense*, Oct. 2017, pp. 99–109.

[26] H. Jin, Z. Li, D. Zou, and B. Yuan, "DSEOM: A framework for dynamic security evaluation and optimization of MTD in container-based cloud," *IEEE Trans. Depend. Secure Comput.*, vol. 18, pp. 1125–1136, 2019.

[27] D. Tayouri, N. Baum, A. Shabtai, and R. Puzis, "A survey of MulVAL extensions and their attack scenarios coverage," *IEEE Access*, vol. 11, pp. 27974–27991, 2023.

[28] X. Ou, S. Govindavajhala, and A. W. Appel, "Mulval: A logic-based network security analyzer," in *Proc. 14th Conf. USENIX Secur. Symp.*, 2005, pp. 113–128.

[29] S. Noel, E. Harley, K. Tam, M. Limiero, and M. Share, "CyGraph: Graph-based analytics and visualization for cybersecurity," in *Handbook of Statistics*, vol. 35, V. N. Gudivada, V. V. Raghavan, V. Govindaraju, and C. Rao, Eds. Amsterdam, The Netherlands: Elsevier, 2016, pp. 117–167.

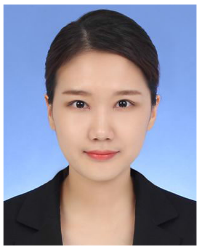
[30] P. Johnson, R. Lagerstrom, M. Ekstedt, and U. Franke, "Can the common vulnerability scoring system be trusted? A Bayesian analysis," *IEEE Trans. Depend. Secure Comput.*, vol. 15, no. 6, pp. 1002–1015, Nov. 2018.

[31] S. Zhang and S. Song, "A novel attack graph posterior inference model based on Bayesian network," *J. Inf. Secur.*, vol. 2, no. 1, pp. 8–27, 2011.

[32] J. Homer, S. Zhang, X. Ou, D. Schmidt, Y. Du, S. R. Rajagopalan, and A. Singhal, "Aggregating vulnerability metrics in enterprise networks using attack graphs," *J. Comput. Secur.*, vol. 21, no. 4, pp. 561–597, Sep. 2013.

[33] D. J. Leversage and E. J. Byres, "Estimating a system's mean time-to-compromise," *IEEE Secur. Privacy*, vol. 6, no. 1, pp. 52–60, Jan./Feb. 2008.

[34] *Cyber Kill Chain (CKC)*. Accessed: Mar. 22, 2023. [Online]. Available: https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html

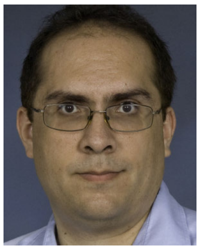


HYEJIN KIM received the B.S. degree in electronic engineering from Dong-A University, Busan, Republic of Korea, in 2017. She is currently pursuing the integrated M.S. and Ph.D. degrees with the AI Graduate School, Gwangju Institute of Science and Technology (GIST), Gwangju, Republic of Korea. Her research interests include cybersecurity and artificial intelligence.

EUISEOK HWANG (Senior Member, IEEE) received the B.S. and M.S. degrees from Seoul National University, in 1998 and 2000, respectively, and the M.S. and Ph.D. degrees in electrical and computer engineering from Carnegie Mellon University (CMU), Pittsburgh, PA, USA, in 2010 and 2011, respectively. He was with the Digital Media Research Center, Daewoo Electronics Company Ltd., South Korea, from 2000 to 2006, and the Data Controller Division, Channel Architecture Group, LSI (now Broadcom), San Jose, CA, USA, from 2011 to 2014. Since 2015, he has been an Assistant Professor/Associate Professor with the School of EECS, the AI Graduate School, and the School of Mechatronics, GIST. In 2021 and 2022, he was a Visiting Scholar with the Department of Computer Science and Engineering, University of Michigan, Ann Arbor. He is currently an Associate Professor with the School of Electrical Engineering and Computer Science (EECS), Gwangju Institute of Science and Technology (GIST), South Korea. His research interests include statistical signal processing, machine learning, and channel coding for data storage and communication systems focused on the physical layer and their emerging ICT applications, such as the Internet of Things and smart grids.

DONGSEONG KIM received the Ph.D. degree from Korea Aerospace University, in February 2008. From June 2008 to July 2011, he was a Postdoctoral Researcher with Duke University. He was a Senior Lecturer/Lecturer in cybersecurity with the University of Canterbury, from August 2011 to December 2018. He was a Visiting Scholar with the University of Maryland, College Park, in 2007. He has been an Associate Professor in cybersecurity with The University of Queensland, Australia, since January 2019. His research interests include automated cybersecurity modeling and analysis for the Internet of Things, cloud computing, and moving target defense. He was the General Co-Chair of ACISP2019 and the General Chair of IEEE PRDC 2017. He served as the Program Co-Chair for IEEE TrustCom2019, IEEE ICIOT2019, ATIS2017, GraMsec2015, and IEEE DASC2015, and a Program Committee Member for international conferences, including IFIP/IEEE DSN, ISSRE, SRDS, and ICC CISS, respectively.

JIN-HEE CHO (Senior Member, IEEE) received the M.S. and Ph.D. degrees in computer science from Virginia Tech, in 2004 and 2008, respectively. She has been an Associate Professor with the Department of Computer Science, Virginia Tech, since August 2018, and the Director of the Trustworthy Cyberspace Laboratory. Prior to joining Virginia Tech, she has been a Computer Scientist with the U.S. Army Research Laboratory (USARL), Adelphi, Maryland, since 2009. She has published over 120 peer-reviewed technical papers in leading journals and conferences in the areas of trust management, cybersecurity, metrics and measurements, network performance analysis, resource allocation, agent-based modeling, uncertainty reasoning and analysis, information fusion/credibility, and social network analysis. She is a member of the ACM. She received the best paper awards in IEEE TrustCom'2009, BRIMS'2013, IEEE GLOBECOM'2017, 2017 ARL's Publication Award, and IEEE CogSima 2018. She is a Winner of the 2015 IEEE Communications Society William R. Bennett Prize in the Field of Communications Networking. In 2016, she was selected for the 2013 Presidential Early Career Award for Scientists and Engineers (PECASE), which is the highest honor bestowed by the U.S. government on outstanding scientists and engineers in the early stages of their independent research careers.



TERRENCE J. MOORE (Member, IEEE) received the B.S. and M.A. degrees in mathematics from American University, in 1998 and 2000, respectively, and the Ph.D. degree in mathematics from the University of Maryland, College Park, in 2010. He is currently a Researcher with the U.S. Army Research Laboratory, Network Science Division. His research interests include sampling theory, constrained statistical inference, stochastic optimization, network security, geometric and topological applications in networks, and network science.

FREDERICA F. NELSON is currently a Researcher and the Program Lead of the U.S. Army Research Laboratory (ARL), Adelphi, MD, USA, where she leads research on machine learning and intrusion detection methods and techniques to promote cyber resilience and foster research on autonomous active cyber defense. She manages and negotiates the research and project agreements for ARL between the network security branch and academia or international organizations. She is also the Lead of the Robust Low-Level Cyber-Attack Resilience for Military Defense (ROLLCAGE) Program working in collaboration with the Army Tank Automotive Research, Development and Engineering Center (TARDEC), Office of Naval Research (ONR), and the Air force Research Laboratory (AFRL) to build a cohesive in-vehicular resilient system for defense against sophisticated enemy malware that strives to blend in with normal system activities. She has over 20 year's combined experience in cybersecurity research, software engineering, and program management within the DoD and other federal services to include the Federal Bureau of Investigation (FBI) and the Department of Justice. She has expertise in leading projects to success from conception to execution and delivery/transfer. She currently serves as the Chairperson of the International Science Technology (IST-163) Panel—NATO Science & Technology Organization (STO) on the topic of deep machine learning for military cyber defense. She is a participant in the Army Education Outreach Program as an Ambassador and a Virtual Judge for the e-Cybermission Program.

HYUK LIM (Member, IEEE) received the B.S., M.S., and Ph.D. degrees from the School of Electrical Engineering and Computer Science, Seoul National University, Seoul, Republic of Korea, in 1996, 1998, and 2003, respectively. From 2003 to 2006, he was a Postdoctoral Research Associate with the Department of Computer Science, University of Illinois at Urbana–Champaign, Champaign, IL, USA. From 2006 to 2021, he was a Professor with the AI Graduate School and jointly with the School of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology (GIST), Gwangju, Republic of Korea. He was the Dean of the School of Electrical Engineering and Computer Science and the Director of the GIST Institute for AI. In 2022, he joined the Korea Institute of Energy Technology (KENTECH), Naju-si, Republic of Korea, as a Full Professor. His research interests include artificial intelligence, cyber-security, big data privacy, software-defined networking, and wireless communication systems. He is also conducting active research on AI applications for cybersecurity, networks, and energy systems.