DIVERGENCE: Deep Reinforcement Learning-Based Adaptive Traffic Inspection and Moving Target Defense Countermeasure Framework

Sunghwan Kim, *Member, IEEE*, Seunghyun Yoon, *Member, IEEE*, Jin-Hee Cho, *Senior Member, IEEE*, Dong Seong Kim, *Senior Member, IEEE*, Terrence J. Moore, *Member, IEEE*, Frederica Free-Nelson, and Hyuk Lim, *Member, IEEE*

***Abstract*—Reinforcement learning (RL) is a promising approach for intelligent agents to protect a given system under highly hostile environments. RL allows the agent to adaptively make sequential defense decisions based on the perceived current state of system security aiming to achieve the maximum defense performance in terms of fast, efficient, and automated detection, threat analysis, and response to the threat. In this paper, we propose a deep reinforcement learning (DRL)-based adaptive traffic inspection and moving target defense countermeasure framework, called 'DIVERGENCE,' for building a secure networked system. The DIVERGENCE provides two main security services: (1) a DRL-based network traffic inspection mechanism to achieve scalable and intensive network traffic visibility for rapid threat detection; and (2) an address shuffling-based moving target defense (MTD) technique to defend against threats as a proactive intrusion prevention mechanism. Through extensive simulations and experiments, we demonstrate that the DIVERGENCE successfully caught malicious traffic flows while significantly reducing the vulnerability of the network through MTD.**

***Index Terms*—Traffic inspection, moving target defense, deep reinforcement learning, software-defined networking.**

Manuscript received 12 April 2021; revised 16 September 2021 and 7 December 2021; accepted 21 December 2021. Date of publication 3 January 2022; date of current version 31 January 2023. This material is based upon work supported by the International Technology Center Pacific (ITC-PAC) under Contract No. FA520920C0022, and the research was partly supported by the Army Research Office under Grant Contract Numbers W91NF-20-2-014 and NSF Grant 2107450. The associate editor coordinating the review of this article and approving it for publication was A. Dhamdhere. *(Corresponding author: Hyuk Lim.)*

Sunghwan Kim is with the Samsung Research, Samsung Electronics, Seoul 06765, South Korea (e-mail: sh001.kim@samsung.com).

Seunghyun Yoon is with the Korea National Engineering Technology Center, Korea Institute of Industrial Technology, Cheonan 15588, Gyeonggi, South Korea (e-mail: syoon@kitech.re.kr).

Jin-Hee Cho is with the Department of Computer Science, Virginia Tech, Falls Church, VA 22043 USA (e-mail: jicho@vt.edu).

Dong Seong Kim is with the School of Information Technology and Electrical Engineering, University of Queensland, Brisbane, QLD 4072, Australia (e-mail: dan.kim@uq.edu.au).

Terrence J. Moore and Frederica Free-Nelson are with the Network Science Division, U.S. Army Research Laboratory, Adelphi, MD 20783 USA (e-mail: terrence.j.moore.civ@mail.mil; frederica.f.nelson.civ@mail.mil).

Hyuk Lim is with the AI Graduate School, Gwangju Institute of Science and Technology, Gwangju 61005, South Korea (e-mail: hlim@gist.ac.kr).

Digital Object Identifier 10.1109/TNSM.2021.3139928

I. Introduction

THE NEED for automated defense mechanisms has been significantly grown due to a large volume of Internet users and network traffic. This naturally leads to the critical role of artificial intelligence (AI), which allows a system to make autonomous defense decisions dealing with various attacks, such as network scanning attacks, denial-of-service (DoS) attacks, and malware attacks. Deep reinforcement learning (DRL), which combines reinforcement learning (RL) with deep neural networks, has a great potential in solving automated defense decision problems under time-varying environments where there is unknown future information. In recent years, DRL has been successfully adopted for network operation and management automation, such as routing optimization and resource allocation [1], [2], [3].

An intrusion detection system (IDS) is a well-known mechanism to defend against attacks by inspecting network traffic and providing alerts for detected attacks. Based on the inspection of network traffics by the IDS, a network can be protected from malicious activities. For IDSs, traffic inspection resource allocation is an important issue for inspecting network traffic because it is impossible to inspect huge amounts of network traffic with a single IDS with a limited processing capacity. In the literature, various traffic inspection resource allocation approaches for IDSs have been proposed [4], [5]. However, deterministic approaches for IDS resource management do not perform well because the amount of network traffic from the large volume of devices significantly increases, and malicious attack patterns become more complicated and intelligent in modern Internet environments. The resource allocation for traffic inspection should be efficient because it is not feasible to inspect every traffic flows on the entire network using a small number of IDSs with a limited processing capacity for traffic inspection. If the traffic flows sent to an IDS exceed the processing capacity of the IDS, packets would be discarded at the IDS without any inspection. Therefore, it is important to allocate the inspection capability to more vulnerable devices or suspicious traffic flows, especially on the real-time traffic inspection.

For the autonomous defense against malicious attacks, suspicious network traffic flows should be captured and re-routed to the IDS for traffic inspection. To this end, we leverage

the advanced features of software-defined networking (SDN) technology providing high flexibility and programmability. In conventional networks, a routing decision (i.e., packet forwarding) is made at each switch, which often causes a lack of controls over switches, leading to running a system in a less optimized manner. In an SDN environment, an SDN controller takes control and can effectively run all the packet forwarding operations in a centralized manner by decoupling network control and data planes. The use of SDN technology becomes popular in developing various network management applications and cybersecurity applications in networked systems. For example, the SDN controller allows a switch to rewrite packet headers (i.e., changing IP address) and steer traffic toward specific network devices (i.e., IDS) by simply updating the forwarding table of the switch via an OpenFlow protocol [6]. Using the SDN technology, various defense countermeasures, such as simple malicious traffic blocking and moving target defense (MTD), can be easily implemented and automated [7]. As a proactive defense countermeasure, MTD randomly changes network configuration to cause confusion or uncertainty for attackers. For example, random network address shuffling-based MTD can effectively nullify reconnaissance attacks and prevent the attacker from sending malicious traffics to target devices.

For real world applications such as network environment management and operation, RL algorithms such as Q-learning encounter complexity and scalability problems due to the large space of states and actions in a Markov decision process (MDP), which consists of state space, action space, transition probability, and reward function. To solve the scalability problems caused by value-based learning such as Q-learning, RL has been combined with deep learning in a DRL. The objective of the DRL is to learn an optimal policy by utilizing a multi-layer perceptron (MLP)-based non-linear function approximator, which represents the probability distribution of a DRL agent's action strategies in order to maximize the expected long-term reward. DQN, a typical DRL algorithm, is widely used to solve the complex state space problem in Q-learning [8]; however, since DQN has a discrete action space, it is difficult to apply in environments for which the number of actions increases or continuous action control is required.

To solve this problem, a deep deterministic policy gradient (DDPG) using an actor-critic method based on a deterministic policy gradient (DPG) algorithm was proposed. The actor-critic method comprises a critic model that updates an action-value function to maximize long-term rewards and an actor model that determines policies according to the critic model. The DPG finds the optimal deterministic policy using a policy gradient method that optimizes the parameterized policy for non-linear function approximators such as MLP via the gradient ascent algorithm. The DDPG is a model-free, off-policy, and actor-critic algorithm to solve MDP that has tremendous state and action spaces [9], [10].

In this paper, we present a DRL-based adaptive traffic inspection and MTD countermeasure framework, the so-called DIVERGENCE (deep reinforcement learning-based adaptive traffic inspection and moving target defense countermeasure framework). The DIVERGENCE aims to effectively decide a resource allocation policy for traffic inspection and MTD countermeasure in order to capture more malicious flows for multiple IDSs while reducing the vulnerability of devices on an SDN-enabled network using a DDPG algorithm. The DIVERGENCE framework includes two parts, the resource allocation for traffic inspection and the IP shuffling-based MTD for countermeasure. By transforming the network flow information into specially designed groups, the traffic inspection part can map the huge state space to the resource allocation for each group state. By inspecting the network flow states with a trained DRL agent, the MTD countermeasure can find the most vulnerable group on the network that should get the most attention for the IP address shuffling-based MTD.

The key contributions of this paper are summarized as follows.

- We formulate a resource allocation problem for traffic inspection and MTD countermeasure on the SDN-enabled network as a MDP, which considers the uncertainty of malicious flow occurrence by maximizing the long-term reward function to enhance the traffic inspection capability.
- We adopt the DDPG algorithm to automatically allocate resources for adaptive traffic inspection and an IP address shuffling-based MTD to enhance the security of the devices on the network.
- This DDPG-based approach gradually learns a better traffic inspection policy under uncertainty in the future network flows to capture more malicious flows and enhance the security of the network.

II. Related Work

A. Traffic Inspection

To inspect and detect network-based threats, such as network scanning and DoS attacks, it is necessary to obtain traffic visibility through network traffic monitoring. In traditional networks, a traffic monitoring agent provided by the network device vendor can be installed at a specific switch, and thus only limited flow information can be collected [11], [12]. Thanks to the SDN's flexibility, scalability, and programmability, the SDN technology has been widely used to develop more efficient network traffic monitoring schemes with higher accuracy and lower overhead. Ha *et al.* formulated an optimization problem to obtain optimal traffic sampling rates of switches for IDS traffic inspection on an SDN-capable network [4]. They attempted to minimize the malicious traffic sampling failure rate by allocating higher sampling rates to the flows that are identified to be more suspicious. Yoon *et al.* proposed a monitoring point and flow capture rate decision method that uses a flow-based centrality measure and a traffic matrix for scalable network traffic monitoring in SDN-enabled networks [13]. Wang *et al.* presented time-based fine-grained flow monitoring architecture in the SDN-enabled network [5]. They analyzed the spatial-temporal factors of SDN switches to decide the monitoring point and duration.

Recently, learning-based approaches such as RL and DRL have been applied to the traffic inspection problems because it is important to efficiently exploit the traffic

inspection resources and SDN capability for security purposes. Deng *et al.* proposed a deep Q-network-based traffic monitoring framework to capture more short-life time flows without redundancy [14]. Castillo *et al.* proposed IPro, an RL-based traffic monitoring architecture, which focuses on solving the problem of control plane overhead and extra CPU usage of the SDN controller [3]. Phan *et al.* proposed DeepMatch, a flow matching framework to provide a fine-grained flow measurement in SDN using deep dueling neural networks [15]. DeepMatch maximizes flow granularity by learning a flow rule updating strategy through the DRL algorithm while protecting SDN-enabled switches from a flow-table overflow. Kim *et al.* proposed a DDPG-based traffic sampling algorithm on an SDN-capable network equipped with multiple traffic analyzers. They proposed system can learn a sampling resource allocation policy for multiple traffic analyzers by taking the inspection results of previously traffic flow samples as the reward of the learning system [16].

B. Moving Target Defense

MTD is one of the promising cybersecurity mechanisms to proactively thwart potential attackers as an intrusion prevention mechanism. The key underlying idea of MTD is to increase the uncertainty of the system by changing attack surfaces (i.e., system or network configurations) with the aim of invalidating the attacker's malicious activities. Cho *et al.* provided a comprehensive survey on various aspects of MTD including design principles, classifications, key methodologies and algorithms, and evaluation metrics in [7].

MTD is designed to increase uncertainty and/or confusion to attackers attempting to penetrate into a system by identifying vulnerabilities of the target system. The main functions of MTD are to change the attack surfaces (i.e., system or network configurations), consequently invalidating the intelligence gathered by the attackers and wasting their resources and time [17]. The MTD techniques can be categorized in terms of shuffling, diversity, and redundancy [7]. Suffling-based MTD is the most commonly deployed MTD that randomizes system and networking configurations, which includes IP shuffling [18], [19], [20], [21], [22], packet header randomization [23], port hopping [24], virtual machine or proxy migration [25], [26], software reconfiguration [27], [28], or service replacement [29]. Recently, some studies [30], [31] have introduced network topology shuffling-based MTD techniques. Diversity-based MTD provides the capability to deploy different implementations of the same functionalities or services such as software stack diversity for enhancing network resilience and service provisions [32], code diversity for providing functionally-equivalent variants [33], network diversity for using the diversity of network configurations [34], data diversity for generating N-variants framework [35], and programming language diversity for avoiding code injection attacks [36]. Redundancy-based MTD improves system reliability by creating multiple replicas of network components such as redundancy of Web servers [37] or network sessions in cyber-physical systems [38]. Redundancy can be used in conjunction with the shuffling [39] or diversity techniques [40] to implement the MTD.

Nowadays, some recent studies verify that the use of SDN functionalities, such as packet header management, dynamic IP/MAC address management, and routing path modification, can enable the execution of MTD operations to further enhance system security [20], [41]. Using the SDN technology, various defense countermeasures, such as simple malicious traffic blocking and rerouting, can be easily implemented and automated [7]. As a proactive defense countermeasure, the SDN technology provides all the programmable functionalities needed to implement MTD that randomly change network configuration to cause confusion or uncertainty for attackers. For example, random network address IP/MAC address shuffling-based MTD can be readily implemented using the standard SDN OpenFlow protocol to nullify reconnaissance attacks and prevent the attacker from sending malicious traffics to target devices. Recently, several MTD technologies using SDN were presented such as network topology shuffling-based MTD [31], an OpenFlow-based random host mutation, called as OF-RHM, architecture [41], and port-hopping MTD [20], [42].

One of the IP shuffle-based MTD implementations using SDN is to allow only an SDN controller to know the real IP addresses of hosts, while the hosts on the network use only virtual IP addresses for communicating with each other [43]. Suppose a source host sends a packet to a destination host. The SDN controller securely maintains a mapping of real-virtual addresses in its database. As the mapping is shuffled randomly, each host would have a different virtual IP address over time. When the source host requests the IP address of the destination using DNS (Domain Name System) query, the SDN controller generates the DNS reply with the virtual IP address of the destination rather than the real IP address and delivers the DNS reply to the source host using OpenFlow Packet-Out message. As the result, the source host can use the virtual IP address for the destination host. When a new packet arrived at an SDN-enabled switch, the SDN controller looks up the virtual addresses for the source host and installs the appropriate flow rules on the SDN-enabled switches. The first switch converts the real source IP address to a virtual IP address in the packet header using the flow rule sent by the SDN controller. Then, the packet is forwarded to the destination with the virtual IP address. On the switch directly connected to the destination host, the virtual IP address of the destination host in a packet header is converted to its real IP address. The source and destination hosts as well as the relay nodes do not know the real IP address of each other, but they can communicate with each other on the SDN network. Performing the MTD on the SDN networks incurs the overhead of the computing resource of the SDN controllers, traffic congestion on the data plane of the SDN network, the breakdown of data flows in transit, or quality-of-service (QoS) degradation of ongoing services, especially in the network layer MTD strategy [44]. Therefore, it is essential to develop an MTD strategy that can allocate more MTD resources to vulnerable devices and suspicious traffic flows. Our proposed method

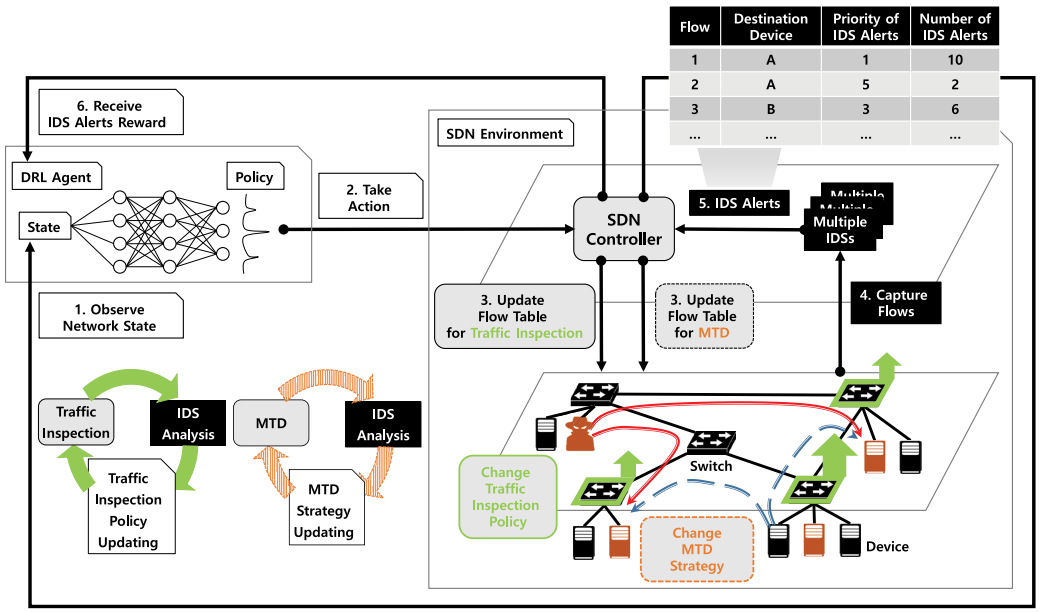


Fig. 1. DIVERGENCE framework on an SDN-enabled network.

can differentiate the MTD resource usage for the traffic flows depending on the suspiciousness level reported by the traffic inspectors.

III. Proposed DIVERGENCE Framework

We propose an SDN-based intelligent cybersecurity framework, which consists of a DRL agent, an SDN controller, SDN-enabled switches, and multiple IDSs as shown in Figure 1. In the proposed framework, the DRL agent on the SDN controller learns an optimal traffic inspection resource allocation policy under the uncertainty of malicious flow occurrence and performs MTD according to traffic inspection results reported from multiple IDSs. The DRL has a great potential for solving the automated resource allocation problem under time-varying environments with uncertainty of future network flows. The proposed framework includes three processes:

- *Monitoring:* The DRL agent on the SDN controller observes network state information and takes action according to the action-selection policy. The SDN controller updates the switch's flow table based on actions determined by the DRL agent to capture network traffic.
- *Inspection:* Multiple IDSs receive and analyze the traffic captured by the switches and report the inspection results containing information related to IDS alerts to the SDN controller's DRL agent as a reward.
- *Reconfiguration:* The SDN controller adjusts the traffic inspection resource of each switch and the IP address of each device according to the traffic inspection and MTD resource allocation policy determined by the DRL agent.

The entire process is conducted periodically over time and the DRL agent gradually learns a better resource allocation policy to maximize IDS alerts while reducing the vulnerability of the network.

A. Threat Model

We consider network scanning and DoS attacks, which are performed at multiple stages. We assume that the attack has the following attack behaviors:

- *Identification and analysis of currently active devices in the network:* Active devices in the network can be identified by the attacker's scanning activities. Attackers leverage network scanning tools to identify active devices in the network. For example, an attacker can use the Nmap [45] tool to know which devices are currently running on the target network.
- *DoS attack to a target device:* Depending on an attack goal, an attacker decides how to attack the device based on the scanning results. Attackers may perform DoS attacks to invalidate the running services on the target device. DoS attacks make the system's resources insufficient, preventing its use for its intended purpose. It is assumed that these malicious activities can be identified by the IDS.

B. Proposed Markov Decision Process Model in Reinforcement Learning

We assume that there are I devices to be protected, J SDN-enabled switches, and K IDSs in the network. The devices, switches, and IDSs are denoted by $\boldsymbol{v} = [v_1, v_2, \ldots, v_I]^T$, $\boldsymbol{o} = [o_1, o_2, \ldots, o_J]^T$, and $\boldsymbol{d} = [d_1, d_2, \ldots, d_K]^T$, respectively. The SDN controller periodically allocates resources for traffic inspection and MTD operations. Let t denote the resource allocation period that represents a time step in MDP. The processing capacity of IDS d_k is denoted by c_k (packets/t) and the total capacity of IDSs can then be calculated as follows:

$$\sum_k c_k = C. \tag{1}$$

Let $\boldsymbol{F_{tot}}$ be the set of total flows in the network during t and the element f_m be defined as:

$$f_m = \{(src_m, dst_m) | src_m \in \boldsymbol{v} \text{ and } dst_m \in \boldsymbol{v}\}, \quad (2)$$

where src_m and dst_m represent the source and destination IP address, respectively. The data rate of flow f_m during t is denoted by λ_m (packets/t). We obtain network environment information through Openflow protocol so we can model the network using MDP to adopt DRL algorithm.

Let $\boldsymbol{g} = [g_1, g_2, \ldots, g_N]^T$ denote a set of groups used in resource allocation for which $|\boldsymbol{F_{tot}}|$ flows are clustered into N groups. Note that $|\boldsymbol{F_{tot}}|$ is changed dynamically at each resource allocation period t, and N is determined by a hash function $H(\cdot)$. The key of $H(\cdot)$ is the destination IP address of each flow, and the hash value is a sub-net of them. For example, $H(\cdot)$ maps flows with the destination IP addresses ranging from 10.0.1.1 to 10.0.1.255 into a single hash value of 10.0.1.0 and flows with the destination IP addresses from 192.168.1.1 to 192.168.1.255 into 192.168.1.0 depending on their sub-net IP addresses. Therefore, we can manage traffic inspection resources in the unit of the sub-net group.

Given $\hat{\boldsymbol{\lambda}} = [\hat{\lambda}_1, \hat{\lambda}_2, \ldots, \hat{\lambda}_N]^T$ denoting a set of the current total data rate of flows in each group at time step t, the element $\hat{\lambda}_n$ can be calculated by:

$$\hat{\lambda}_n = \sum_{m \in g_n} \lambda_m. \quad (3)$$

We define a traffic inspection resource allocation of each group vector, $\boldsymbol{x} = [x_1, x_2, \ldots, x_N]^T$, where the element $x_n \in (0, 1)$. Let $\boldsymbol{y} = [y_1, y_2, \ldots, y_N]^T$ be the set of allocated traffic inspection resources for each group g_n depending on the $\boldsymbol{x}$, the element y_n can be given by:

$$y_n = x_n \cdot \hat{\lambda}_n. \quad (4)$$

Because the total amount of traffic inspection resource should not exceed the total processing capacity of IDSs, the relationship between C and $\boldsymbol{y}$ is given by:

$$\sum_n y_n \leq \min\{C,\ C_{max}\}, \quad (5)$$

where C_{max} is a tunable parameter to adjust the amount of network resource to be used for the proposed method. If C is too large to be supported by the SDN network without scarifying normal data delivery, the resource should be reduced appropriately by setting smaller C_{max}. Here, we assume that $C \ll C_{max}$ for simplicity. Let $\bar{\boldsymbol{y}} = [\bar{y}_1, \bar{y}_2, \ldots, \bar{y}_N]^T$ be the set of normalized allocated traffic inspection resource, the element $\bar{y}_n$ can be obtained by:

$$\bar{y}_n = \frac{y_n}{\sum_n y_n} \cdot C. \quad (6)$$

After assigning traffic inspection resources to each group, we should distribute a certain amount of resources to each flow in each group because traffic inspection is conducted in the unit of flow at the IDS. The inspection result for each flow f_m is defined by:

$$w_m = \frac{N^A_{f_m}}{P^A_{f_m}}, \quad (7)$$

where $N^A_{f_m}$ is the number of alerts for f_m and $P^A_{f_m}$ is the priority of alert for f_m. Note that the number of alerts is estimated based on one alert per one malicious packet while the priority of alert is scaled from 1 to 5 where a smaller number represents high priority meaning a higher risk alert. One of the popular IDSs, Snort, which is an open-source IDS software that inspects network traffic by matching it with pre-defined security rule sets, generates an alert message consisting of a priority, flow information with IP address, port number, and alert name. In order to increase traffic inspection performance, multiple IDSs can be deployed. We utilize the weighted max-min fair share (WMMFS) mechanism to inspect each flow fairly in each group by allocating normalized traffic inspection resource $\bar{y_n}$ depending on the weight for inspection result w_m. The WNNFS is widely used for network resource allocation, flow routing, and load balancing schemes. For example, the WMMMFS is applied to a pricing scheme for networks that use priorities to provide differentiated quality of service [46], [47], [48]. The WMMFS is the ideal fair distribution of a shared scarce resource with the consideration of weights. We define the WMMFS for our proposed DIVERGENCE by:

- Traffic inspection resources are allocated in the order of increasing demand depending on the data rate of flows, which are normalized by the weighted summation of IDS alerts.
- No flow obtains a resource share that exceeds its demand.
- Flows with unsatisfied demands obtain resource shares in proportion to their weights.

In the WMMFS, given the normalized resource capacity, $\bar{y}_n$, to capture the maximum flows, flow $f_{m \in g_n}$ is guaranteed to have the minimum resource defined by:

$$\bar{y}_{min} = \max\left(\frac{1}{|g_n|} \cdot \bar{y}_n,\ \frac{w_m}{\sum_{m \in g_n} w_m} \cdot \bar{y}_n\right). \quad (8)$$

If some flows need less than what they are entitled to, then other flows can receive more than $\bar{y}_{min}$.

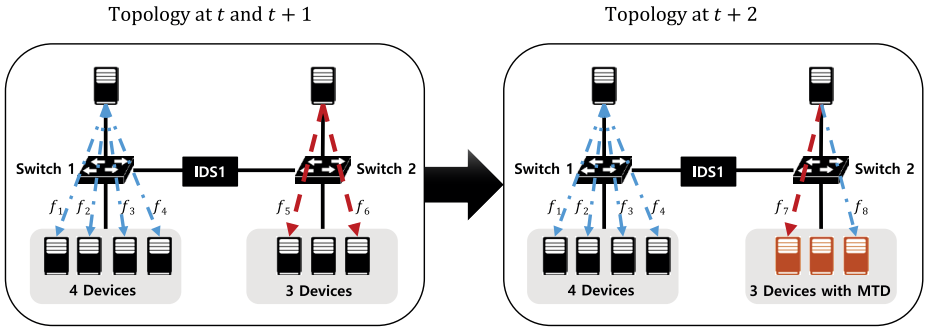
After inspecting flows by allocated resource $\bar{y}$ in the WMMFS, the DIVERGENCE performs an IP address shuffling-based MTD countermeasure for security enhancement of each device in each group. We define a probability of IP address shuffling for sub-net group g_n during time step t as:

$$\mathbb{P}_{g_n}(t) = \frac{\sum_{m \in g_n} w_m}{\sum_n \sum_{m \in g_n} w_m}. \quad (9)$$

The above equation means that the probability of changing IP addresses of devices belonging to the sub-net group at each time step is proportional to the IDS inspection result.

State space: The state space of the DIVERGENCE consists of the current network traffic state and allocated resources for the traffic inspection state. Let s_t denote state at time step t, represented by the current total data rate of flows in each group vector and the normalized allocated traffic inspection resource for each group vector as:

$$s_t = [\hat{\boldsymbol{\lambda}},\ \bar{\boldsymbol{y}}\,]. \quad (10)$$



- Capacity of IDS: $C = 50$ (packets/sec, pps)
- Data rate of normal flows: f_1, f_2, f_3, f_4, and f_8= 25 pps
- Data rate of malicious flows: f_5, f_6, and $f_7 = 25$ pps
- $dst_1, dst_2, dst_3, dst_4$: 10.0.0.1 ~ 10.0.0.4
- IP addresses of dst_5 and dst_6 at t and t+1: 172.26.0.1 and 172.26.0.3
- IP addresses of dst_7 and dst_8 at t+2: 192.168.0.1 and 192.168.0.3
- Groups at t and $t+1$: $g_1 = \{f_1, f_2, f_3, f_4\}$ and $g_2 = \{f_5, f_6\}$
- Groups at $t+2$: $g_1 = \{f_1, f_2, f_3, f_4\}$ and $g_2 = \{f_7, f_8\}$

Fig. 2. Simple network topology.

TABLE I
AN EXAMPLE OF STATE, ACTION, AND REWARD TABLE

Time	State	Action	Reward
t	$[(\hat{\lambda}_1 = 100, \hat{\lambda}_2 = 50), (\bar{y}_1 = 0, \bar{y}_2 = 0)]$	$[x_1 = 1, x_2 = 0.5]$	$R(s_t, a_t) = 0$
$t+1$	$[(\hat{\lambda}_1 = 100, \hat{\lambda}_2 = 50), (\bar{y}_1 = 40, \bar{y}_2 = 10)]$	$[x_1 = 0.5, x_2 = 1]$	$R(s_{t+1}, a_{t+1}) = 7.5$
$t+2$	$[(\hat{\lambda}_1 = 100, \hat{\lambda}_2 = 50), (\bar{y}_1 = 25, \bar{y}_2 = 25)]$	$[x_1 = 0.1, x_2 = 0.6]$	$R(s_{t+2}, a_{t+2}) = 12.5$

The state s_t changes in accordance with a given traffic inspection resource allocation policy and the uncertainty of the network flows.

Action space: The action space of the DIVERGENCE is an action that determines the traffic inspection resource allocation for each group. An action at time step t is given by:

$$a_t = [\boldsymbol{x}]. \tag{11}$$

An action a_t indicates that changing traffic inspection resource allocation for each group at time step t. When the agent is in the state s_t, it will decide which group to change resource allocation depending on $\boldsymbol{x} \in [0, 1]$, and then the allocated resource of each group $\bar{\boldsymbol{y}}$ and the weight for inspection result w_m for each flow will be changed. The probability that the DIVERGENCE undergoes a transition from state s_t to state s_{t+1} when action a_t is taken can be represented as $\mathbb{P}_{sa}{:}(s_t, a_t) \rightarrow s_{a_t}$. Since the traffic inspection resource of each group can be changed dynamically using an SDN controller, $\mathbb{P}_{sa}$ can be determined using the probability of selecting action a_t at state s_t defined by policy π.

Reward: The comprehensive reward of the DIVERGENCE should be designed to achieve the adaptive flow inspection to capture more malicious flows under the uncertainty of the malicious flow occurrence. To achieve more intensive malicious flow inspection, we define flow inspection reward as follows:

$$r_t = \sum_n \sum_{m \in g_n} w_m. \tag{12}$$

Let R denote the reward function that returns a cost value indicating whether the proposed framework utilizes IDSs to inspect more malicious flows. When the action a_t is taken in the state s_t, the reward function is defined by:

$$R(s_t, a_t) = r_t. \tag{13}$$

This reward function is used to measure the performance of the proposed framework.

Figure 2 shows a simple network topology, and Table I shows a list of corresponding state, action, and reward. For example, group g_1 and g_2 have 4 and 2 flows, respectively, and the alert priority for malicious flows $P^A_{f_5}$ and $P^A_{f_7}$ are set to 1, and $P^A_{f_6}$ is set to 2, respectively. We assumed that a_t is set to [1, 0.5] initially. At time $t+1$, the DRL agent allocates resource 10 for each flow in g_1 and 5 for each flow in g_2 using (8) and WMMFS strategy. Note that $\bar{y}_n$ for state s_{t+1} is calculated using (4) and (6) depending on the initial action a_t. After traffic inspection at the IDS depending on the allocated resource, the reward is calculated as 7.5 according to (7) and (12) because w_1, w_2, w_3, and w_4 are zero, and w_5 and w_6 are 2.5 and 5, respectively. Then, the DIVERGENCE decides to conduct MTD for all devices in g_2 using (9). Similarly, at $t+2$, when taking action a_{t+1}, the state is changed to s_{t+2} and the reward increases to 12.5 according to (12) because the traffic inspection resource 12.5 is allocated to the new malicious flow f_7.

C. DDPG-Based Resource Allocation for Traffic Inspection and MTD Operations

To account for the impact of the current action on future rewards, we define the total expected discount reward according to the π policy, which defines the probability of selecting

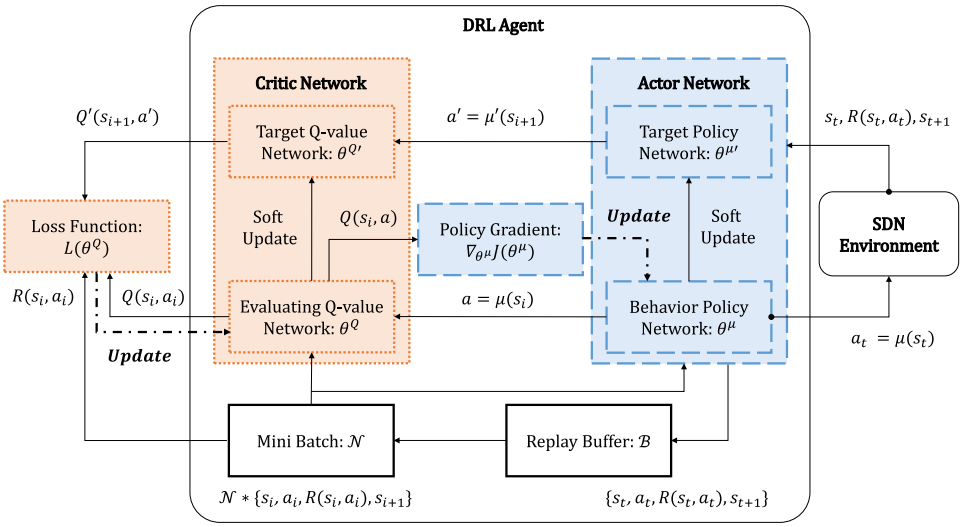


Fig. 3. Block diagram for DDPG training process.

action a_t at state s_t, as follows:

$$R_t^\pi = R(s_t, a_t) + \sum_{i=1}^{\infty} \gamma^i \cdot R(s_{t+i}, a_{t+i}). \tag{14}$$

The goal of the DRL agent is to identify the optimal action selection policy that increases the probability of capturing malicious flows under the uncertainty of malicious flow occurrence. To achieve this objective, we consider the action-value function that returns the total expected discounted reward when action a_t is taken in state s_t following policy π as:

$$\begin{aligned} Q(s_t, a_t) &= \mathbb{E}\{R_t^\pi\} \\ &= \mathbb{E}\left\{ R(s_t, a_t) + \sum_i \gamma^i \cdot R(s_{t+i}, a_{t+i}) \right\}. \end{aligned} \tag{15}$$

To approximate the optimal action-value function, Q^* is defined as:

$$\begin{aligned} Q^*(s, a) = \mathbb{E}\Bigg\{ & R(s_t, a_t) \\ & + \sum_i \gamma^i \max_{a_{t+i}} R(s_{t+i}, a_{t+i})|_{s=s_t, a=a_t} \Bigg\}. \end{aligned} \tag{16}$$

In continuous action space as modeled in Section III-B, it is difficult to discriminate the optimal action from the action-value function because the agent is required to explore every Q-value, i.e., $\mu(s_t) = \text{argmax}_{a_t} Q(s_t, a_t)$, and since complexity grows exponentially according to the size of the state and action spaces [49].

To determine the action-selection policy that increases the total expected discounted reward defined by (14), we consider a DDPG algorithm, which has model-free, off-policy, and actor-critic characteristics [9]. To deal with the continuous action space in DRL, the DDPG algorithm tries to approximate the optimal action-value function defined by (16) using two neural networks as shown in Figure 3. The first neural network is used for approximating the action-value function defined by (15), termed a critic-network. Its inputs are action and observation, and the output is the value of the action-value function, i.e., $Q(s_t, a_t)$. The second neural network is an actor-network that approximates behavior policy. Its input is observation and its output is action-value, i.e., $\mu(s_t)$. Let $Q(s, a|\theta^Q)$ and $\mu(s|\theta^\mu)$ denote the critic-network and actor-network, respectively. The network functions for the critic and actor target network are thus represented as Q' and μ', respectively.

The DDPG uses θ^Q for the critic-network and θ^μ for the actor-network to parameterize non-linear function approximators. θ^Q and θ^μ are updated according to loss function and policy gradient, respectively. The weight of critic-network θ^Q is optimized by minimizing loss as follows:

$$L(\theta^Q) = \frac{1}{\mathcal{N}} \sum_i \left(y_i - Q\left(s_i, a_i|\theta^Q\right) \right)^2, \tag{17}$$

where $y_i = R(s_i, a_i) + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$ and $\mathcal{N}$ is the number of transitions sampled from a replay buffer $\mathcal{B}$. The DDPG optimizes θ^μ by updating $J(\theta^\mu)$, which is the objective function of policy gradient method in the direction of $\nabla_{\theta^\mu} J(\theta^\mu)$ defined as follows:

$$\begin{aligned} \nabla_{\theta^\mu} J(\theta^\mu) \approx \frac{1}{\mathcal{N}} \sum_i & \nabla_a Q\left(s, a|\theta^Q\right)|_{s=s_i, a=\mu(s_i)} \\ & \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s=s_i}. \end{aligned} \tag{18}$$

To update the target networks gradually, the agent updates $\theta^{Q'}$ and $\theta^{\mu'}$ according to $\theta^{Q'} = \tau_c \theta^Q + (1 - \tau_c)\theta^{Q'}$, and $\theta^{\mu'} = \tau_a \theta^\mu + (1 - \tau_a)\theta^{\mu'}$, where τ_c and τ_a are positive small numbers between 0 and 1 used to learn rates corresponding to the critic-network and actor-network, respectively.

The proposed DDPG-based resource allocation policy update solution is described in Algorithm 1. The critic-network, actor-network, and target networks are initialized with experience replay buffer and initial observation in lines 1 – 4. Overall, the algorithm consists of a loop for time step *t*. Lines 5 – 13 show the loop for each time step *t*. This loop is repeated infinitely. One action a_t is selected by following the actor-network policy with adaptive noise process $\mathcal{P}$ in line 6. To assure exploration efficiency, the algorithm adds adaptive noise to the parameters of the neural network policy, not

Algorithm 1 The DDPG-Based Resource Allocation Algorithm

1: Initialize critic-network $Q(s,a|\theta^Q)$ and actor-network $\mu(s|\theta^\mu)$ with randomly generated weight θ^Q and θ^μ
2: Set target networks Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$ and $\theta^{\mu'} \leftarrow \theta^\mu$
3: Initialize experience replay buffer $\mathcal{B}$
4: Set initial state s_0 according to the initial resource allocation policy
5: **for** each time step **do**
6: Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{P}$
7: Take action a_t, observe $R(s_t,a_t)$, s_{t+1}, and store transition $\{s_t, a_t, R(s_t,a_t), s_{t+1}\}$ in $\mathcal{B}$
8: Perform the IP address shuffling-based MTD with a probability in (9) for security enhancement
9: Randomly sample a batch of $\mathcal{N}$ transitions $\{s_i, a_i, R(s_i,a_i), s_{i+1}\}$ from $\mathcal{B}$
10: Set $y_i = R(s_i,a_i) + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^Q)$
11: Update critic θ^Q and actor θ^μ in (17) and (18)
12: Update the targets softly to $\theta^{Q'}$ and $\theta^{\mu'}$
13: **end for**

the action space [50]. In line 7, the reward $R(s_t,a_t)$ and the next state s_{t+1} are observed by taking the selected action a_t, and the observed transition is stored as replay buffer $\mathcal{B}$. In line 8, the IP shuffling-based MTD countermeasure according to the probability in (9) for security enhancement of devices in the network. In lines 9 – 12, Based on the randomly sampled mini batch from $\mathcal{B}$, θ^Q and θ^μ are updated according to (17) and (18), and targets $\theta^{Q'}$ and $\theta^{\mu'}$ are updated softly.

IV. Performance Evaluation

A. Simulation Results

In order to evaluate the performance of the proposed resource allocation algorithm based on DDPG, we have conducted simulations using the networkX [51] for network topology generation and Stable-Baselines [52] for DDPG algorithm implementation. Simulations were performed on two types of network topologies as shown in Figure 4.

- *Fat-tree data center:* This topology shows the representative topology of a data center network environment. The fat-tree topology has the same bandwidth on each link using the same commodity switch [53].
- *Internet autonomous system (AS):* This topology is a random graph-based topology similar to the Internet network environment. The Internet AS topology has three tiers for nodes i.e., tier-1, mid-level, customer or content provider [54].

We compared the performance of the proposed DDPG-based method with that of two non-learning-based methods.

- *Uniform:* This method distributes traffic inspection resources to every switch equally.
- *Rate-Proportional:* In this method, each flow gets a traffic inspection resource proportional to its data rate.
- *Centrality:* This method allocates traffic inspection resources in each switch depending on the betweenness centrality (BC) value of the switch [13]. The BC value for a switch is given by the total number of flows that pass through the switch. The sampling rate of a switch is proportional to the BC value of the switch.

All methods conduct an alert-proportional MTD countermeasure for each device with a probability, which is the ratio of the number of alerts for a victim device to the total number of alerts for all devices depending on the traffic inspection method. We compare two security performance metrics as follows.

- *Capture-failure rate:* This metric represents the average probability of capturing a packet belonging to the malicious flow. The capture-failure rate of a malicious flow is calculated as the number of non-captured malicious packets over the total number of packets for the malicious flow.
- *Degree of security vulnerability:* This metric indicates the percentage of the number of malicious flows successfully sent to the target device. If all malicious flows are successfully transmitted to their target devices, the degree of security vulnerability is 1.

We assumed the traffic inspection and MTD countermeasure policy was updated every second ($t = 1$ second) and the processing capacity of each IDS was set to 1 Gbps. To account for the uncertainty in the malicious activity, 2% of total flows are randomly selected as malicious flows, and the malicious flows decide whether they perform an attack at each time step with a probability of 0.2. Malicious flows are either IP scanning attacks with alert *'priority 4'* or packet flooding DoS attacks with alert *'priority 2'* against devices in a group of victims. It was assumed that whenever a packet of a malicious flow is captured by IDS, a single IDS alert is generated for the flow. In the fat-tree data center topology, the maximum number of flows $|\boldsymbol{F_{tot}}|$ was set to 10,000, and the average data rate of flows was set to 20 Mbps. In the Internet AS topology, $|\boldsymbol{F_{tot}}|$ was set to 40,000, and the average data rate of flows was set to 20 Mbps. The hyper-parameter values for the DDPG algorithm and the other parameter values for the simulation were set as shown in Table II and Table III, respectively. We set policy object that implements actor and critic, using a MLP (two layers of 64 nodes in each), with layer normalisation.

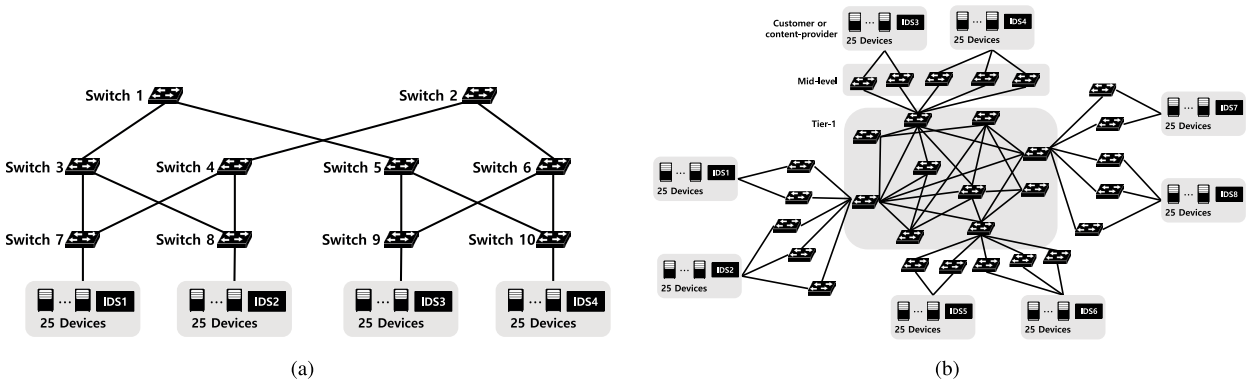
Figure 5 shows that the proposed method maintains lower capture-failure rate for malicious flows than the other methods on both network topologies. In the simulations, if most attacking flows go through a set of switches with high BC

TABLE II
Hyper-Parameter Values Used for the DDPG Algorithm

Hyper-parameter	Value
Discount factor γ	0.99
Replay buffer $\mathcal{B}$	50,000
Mini batch N	128
Critic learning rate τ_c	0.001
Actor learning rate τ_a	0.0001

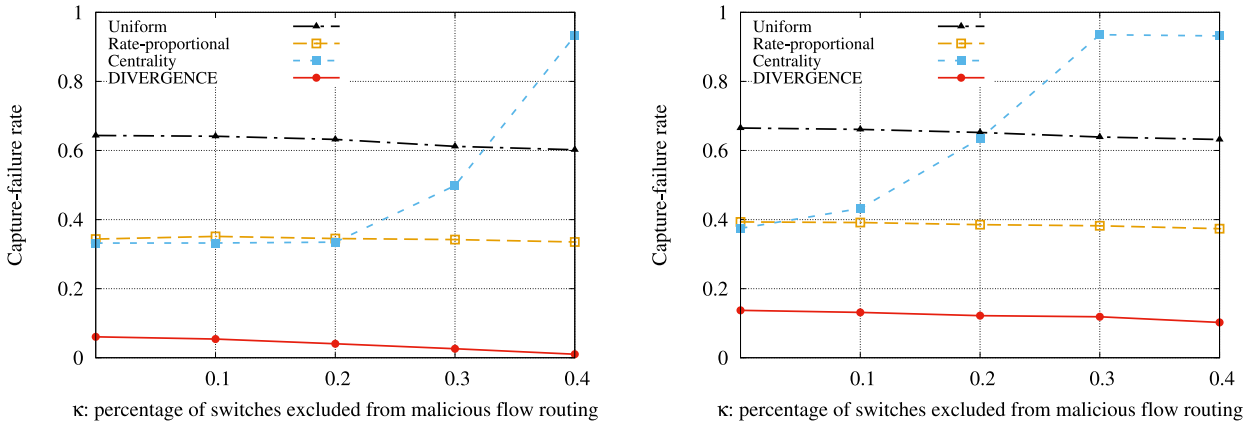
TABLE III
Parameter Values Used for Network Topologies

Parameter	Fat-tree	Internet AS	Testbed
Number of devices I	100	200	100
Number of switches J	10	30	10
Number of IDSs K	4	8	4
Total capacity of IDSs C	4 Gbps	8 Gbps	1 Gbps
Number of Flows $\lvert\boldsymbol{F_{tot}}\rvert$	10,000	40,000	10,250
Number of groups N	4	8	4
Time step t	1 sec	1 sec	10 sec



(a) (b)

Fig. 4. Network topologies for simulation: a) fat-tree data center; b) Internet AS.



(a) (b)

Fig. 5. Capture failure rate of malicious flows with respect to κ: a) fat-tree data center; b) Internet AS.

value, then the traffic inspection problem becomes straightforward because the switches with high BC would be the best place where the traffic flows are sampled. However, if this sampling strategy is known to the attackers, they can simple detour such switches. Here, we introduce a parameter $\kappa \in [0, 1]$ to consider more realistic pattern of attack paths in the topology. κ is defined by a percentage of dominant switches (with high BC value) to be excluded from routing paths of malicious traffic flows in the simulations. For example, if κ is set at 0.1, any switches with a top 10% BC value are excluded from the path when routing malicious flows. As κ increases, less malicious flows pass through switches with high BC values. As expected, the capture-failure rate for the centrality-based method increases rapidly as κ increases in Fig. 5. Even in these cases, the proposed method can capture malicious flows that do not pass through a switch with a high BC value.

Figure 6 represents the degree of a security vulnerability with respect to the number of time steps on both network topologies. As shown in the figures, the proposed algorithm based on DDPG decreases the security vulnerability more than the other methods as the number of time steps increases. The reason is that the proposed method allocates more inspection resource to the victim groups in which more alarms for malicious flows are reported and changes more frequently the IP addresses of the victim devices in the group.

B. Testbed Experiment Results

For verification and empirical evaluation of the proposed algorithm, we have constructed an SDN testbed and implemented the proposed DRL-based traffic inspection and MTD countermeasure framework on the testbed. Figure 7 represents the architecture of the SDN testbed and the graphical user interface (GUI) for the network topology. The testbed consists of a DRL agent, an SDN controller, Snort IDSs, 10 HP 2920 SDN-capable switches, Open vSwitch with Docker containers for end nodes, and a traffic generator. We implemented the DRL agent on a workstation with NVIDIA TITAN Volta GPU using OpenAI framework and the SDN controller on a workstation using OpenDaylight. For the DRL agent, we used Stable-Baselines [52], a fork of OpenAI-Baselines with additional RL algorithms such as DQN and DDPG. For multiple IDSs, we utilized Snort for malicious flow detection on a workstation. Snort is a lightweight IDS for UNIX and MS Windows operating systems. Specifically, with Snort, we can monitor network traffic in real-time to detect dangerous payloads or suspicious anomalies. For the MTD countermeasure, we adopted the virtual IP shuffling-based MTD that only the SDN controller knows the real IP addresses of hosts, while other hosts in the network only use virtual addresses to communicate with each other. Specifically, the MTD countermeasure on the SDN controller deals with the following

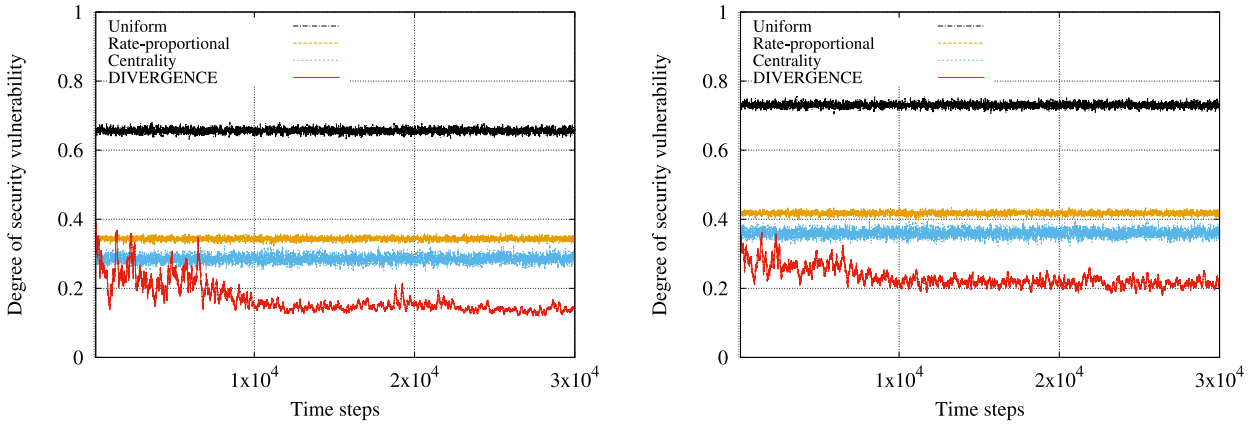


Fig. 6. Degree of security vulnerability with respect to time steps: a) fat-tree; b) Internet AS.

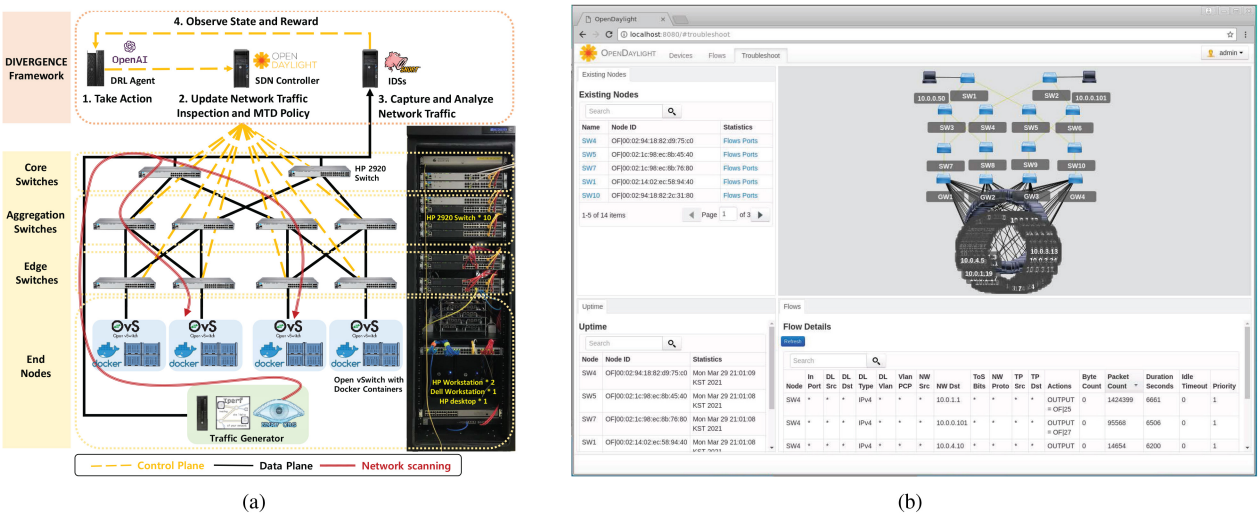


Fig. 7. SDN testbed network topology: a) architecture; b) GUI on OpenDaylight.

tasks: (1) making shuffling decisions of each host; (2) selecting virtual IP addresses that do not overlap (3) mapping from real IP address to virtual IP address during communication, and (4) ensuring communication for normal flows by allowing connection with the previous IP for a certain period of time excluding malicious flows [43]. We constructed a fat-tree network topology with 10 HP 2920 SDN-capable switches and 100 Docker containers. For traffic generation, we used Iperf for normal TCP or UDP flows, and Nmap for a network scanning attack on a desktop. The generated traffic samples including the normal TCP/UDP flows and Nmap network scanning attack flows were saved as a pcap dataset and can be replayed for comparison purposes.

The SDN testbed operates in the following order. First, depending on the traffic inspection policy generated by the DRL agent, the OpenDaylight SDN controller updates the traffic inspection policy and captures packets from each switch to the IDS server. And, the IDS server with four Snort processes analyzes the captured packets from the switches. We used Snort to detect malicious flows from captured packets and generate IDS alerts. Then, the IDS analysis results (i.e., IDS alerts) are reported to the DRL agent for calculating the rewards. Finally, the SDN controller conducts the virtual IP shuffling-based MTD for each group according to the IDS analysis results.

The hyper-parameter values for the DDPG algorithm and the other parameter values for the testbed topology configuration were summarized in Table II and Table III, respectively. We set the policy object for actor and critic using an MLP (two layers of 64 nodes in each) with layer normalisation. The traffic sampling resource allocation policy was updated every 10 seconds ($t = 10$ seconds). If the time step t is large (e.g., $t = 100$ seconds), the IDS can process more traffic samples, and the number of IDS alerts would be less fluctuating on average. This statistical stability is helpful to achieve the convergence of learning. However, it slows down the learning process because one action change is performed per each time step. The total processing capacity of IDSs was set to 1 Gbps. The maximum number of flows $|\boldsymbol{F_{tot}}|$ generated by the traffic generator was set to 10,250 (10,000 normal flows and 250 malicious flows). The average data rate of flows was set to 1 Mbps for normal flows and 740 bps for malicious flows, respectively. To account for the uncertainty in the malicious activity, Nmap on the traffic generator generated a total of

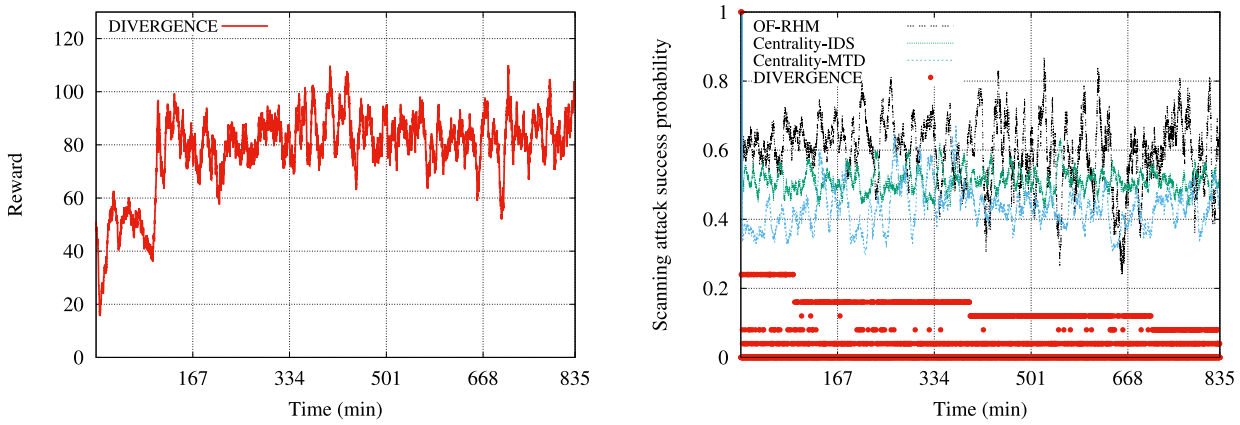


Fig. 8. SDN testbed experiment results for network scanning attack: a) reward; b) attack success probability result.

250 scanning malicious flows for two randomly-selected victim groups. Each malicious flow performs an attack at each time step with 50% probability. In the testbed configuration, if a packet of a malicious flow was captured by Snort, one IDS alert was generated for that flow. The proposed method was compared with the centrality method, which allocates traffic inspection resources to each switch according to the BC value and changes the IP address of the device with the probability of the number of alerts for that device out of the total number of alerts.

We compared the performance of the proposed DIVERGENCE with the other methods:

- *OF-RHM:* This method performs random host mutation, which randomly changes IP addresses of devices in the network with the fixed mutation rate of 0.01 [41].
- *Centrality-IDS:* This method performs the BC value-based traffic inspection but only uses the IDS. This method does not disturb the attacker through the MTD and simply drops the flows for which the alarm occurred.
- *Centrality-MTD:* This method allocates traffic inspection resources in each switch depending on the BC value of the switch, and performs MTD using (9) according to the traffic inspection result.

Figure 8(a) shows the rewards of the agent with respect to the time elapse. Each point is a measured reward at a traffic inspection resource allocation period ($t = 10$ seconds). The DRL agent is more likely to behave randomly at the start of training. However, the reward increases and levels off as the training time increases. The result shows that the agent can capture more malicious flows as the training time increases because the reward indicates how many malicious flows are captured and analyzed at IDSs.

Figure 8(b) shows the network scanning attack success probability with respect to time elapse. As shown in the figure, the proposed method achieves better attack prevention performance than the centrality-MTD method as time goes by because it can prevent more malicious flows. The OF-RHM method only performs MTD, so there is no visibility of IDS alerts, and the central-IDS method only drops flows that raise IDS alerts but do not change the IP address of the victim device. For the few minutes at the beginning, all of the scanning attacks succeed, but the scanning attack success probability abruptly decreases after the procedures of adaptive traffic inspection and MTD countermeasure.

V. Conclusion and Future Work

In this paper, we have proposed a DRL-based automated cybersecurity framework, DIVERGENCE, for allocating traffic inspection and MTD resources. The DIVERGENCE framework includes two parts, the resource allocation for traffic inspection and the IP shuffling-based MTD for countermeasure. By transforming the network flow information into specially designed groups, the traffic inspection part can map the huge state space to the resource allocation for each group state. By inspecting the network flow states with a trained DRL agent, the MTD countermeasure can find the most vulnerable group on the network that should get the most attention for the IP address shuffling-based MTD. Simulation and experiment results showed that our proposed method outperforms the other methods for capturing more malicious flows and reducing security vulnerabilities of the system.

For future work, we will focus on how to expedite the retraining of the DIVERGENCE model to fit dynamic changes of network and attack patterns. Analyzing the overhead caused by traffic inspection and MTDs and incorporating them into the DIVERGENCE model are also challenging issues. Finally, we will enhance the adaptation capability of DIVERGENCE for resource allocation problems in situations where IDSs have the heterogeneous capability and more sophisticated attack detection is required.

Acknowledgment

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of CCDC ITC-PAC, CCDC-ARL, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES

[1] T. A. Q. Pham, Y. Hadjadj-Aoul, and A. Outtagarts, "Deep reinforcement learning based QoS-aware routing in knowledge-defined networking," in *Proc. Int. Conf. Heterogeneous Netw. Qual. Rel. Security Robustness*, 2018, pp. 14–26.

[2] C. Yu, J. Lan, Z. Guo, and Y. Hu, "DROM: Optimizing the routing in software-defined networks with deep reinforcement learning," *IEEE Access*, vol. 6, pp. 64533–64539, 2018.

[3] E. F. Castillo, L. Z. Granville, A. Ordonez, and O. M. C. Rendon, "IPro: An approach for intelligent SDN monitoring," *Elsevier Comput. Netw.*, vol. 170, Apr. 2020, Art. no. 107108.

[4] T. Ha *et al.*, "Suspicious traffic sampling for intrusion detection in software-defined networks," *Elsevier Comput. Netw.*, vol. 109, pp. 172–182, Nov. 2016.

[5] X. Wang, X. Li, S. Pack, Z. Han, and V. C. Leung, "STCS: Spatial-temporal collaborative sampling in flow-aware software defined networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 999–1013, Jun. 2020.

[6] W. Queiroz, M. A. Capretz, and M. Dantas, "An approach for SDN traffic monitoring based on big data techniques," *J. Netw. Comput. Appl.*, vol. 131, pp. 28–39, Apr. 2019.

[7] J.-H. Cho *et al.*, "Toward proactive, adaptive defense: A survey on moving target defense," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 709–745, 1st Quart., 2020.

[8] V. Mnih *et al.*, "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.

[9] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. I-387-ŰI-395.

[10] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[11] R. Sommer and A. Feldmann, "NetFlow: Information loss or win?" in *Proc. ACM SIGCOMM Workshop Internet Meas.*, 2002, pp. 173–174.

[12] M. Wang, B. Li, and Z. Li, "sFlow: Towards resource-efficient and agile service federation in service overlay networks," in *Proc. IEEE Distrib. Comput. Syst.*, 2004, pp. 628–635.

[13] S. Yoon, T. Ha, S. Kim, and H. Lim, "Scalable traffic sampling using centrality measure on software-defined networks," *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 43–49, Jul. 2017.

[14] J. Deng, H. Cai, and X. Wang, "Improved flow awareness by intelligent collaborative sampling in software defined networks," in *Proc. Int. Conf. 5G Future Wireless Netw.*, 2019, pp. 182–194.

[15] T. V. Phan, T. G. Nguyen, and T. Bauschert, "DeepMatch: Fine-grained traffic flow measurement in SDN with deep dueling neural networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2056–2075, Jul. 2021. [Online]. Available: https://doi.org/10.1109/JSAC.2020.3041406

[16] S. Kim, S. Yoon, and H. Lim, "Deep reinforcement learning-based traffic sampling for multiple traffic analyzers on software-defined networks," *IEEE Access*, vol. 9, pp. 47815–47827, 2021.

[17] J.-H. Cho and N. Ben-Asher, "Cyber defense in breadth: Modeling and analysis of integrated defense systems," *J. Defense Model. Simulat.*, vol. 15, no. 2, pp. 147–160, 2018.

[18] S. Antonatos, P. Akritidis, E. P. Markatos, and K. G. Anagnostakis, "Defending against hitlist worms using network address space randomization," *Elsevier Comput. Netw.*, vol. 51, no. 12, pp. 3471–3490, 2007.

[19] 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)*, 2014, pp. 701–706.

[20] P. Kampanakis, H. Perros, and T. Beyene, "SDN-based solutions for moving target defense network protection," in *Proc. IEEE Int. Symp. World Wireless Mobile Multimedia Netw.*, 2014, pp. 1–6.

[21] D. C. MacFarland and C. A. Shue, "The SDN shuffle: Creating a moving-target defense using host-based software-defined networking," in *Proc. ACM Workshop Moving Target Defense*, 2015, pp. 37–41.

[22] D. P. Sharma, D. S. Kim, S. Yoon, H. Lim, J.-H. Cho, and T. J. Moore, "FRVM: Flexible random virtual IP multiplexing in software-defined networks," in *Proc. IEEE Int. Conf. Trust Security Privacy Comput. Commun. IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, 2018, pp. 579–587.

[23] Y. Wang, Q. Chen, J. Yi, and J. Guo, "U-Tri: Unlinkability through random identifier for SDN network," in *Proc. ACM Workshop Moving Target Defense*, 2017, pp. 3–15.

[24] Y.-B. Luo, B.-S. Wang, and G.-L. Cai, "Effectiveness of port hopping as a moving target defense," in *Proc. IEEE Int. Conf. Security Technol.*, 2014, pp. 7–10.

[25] W. Peng, F. Li, C.-T. Huang, and X. Zou, "A moving-target defense strategy for cloud-based services with heterogeneous and dynamic attack surfaces," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2014, pp. 804–809.

[26] Y. Zhang, M. Li, K. Bai, M. Yu, and W. Zang, "Incentive compatible moving target defense against VM-colocation attacks in clouds," in *Proc. IFIP Intl. Inf. Security Conf.*, 2012, pp. 388–399.

[27] V. Casola, A. De Benedictis, and M. Albanese, "A moving target defense approach for protecting resource-constrained distributed devices," in *Proc. IEEE Intl. Conf. Inf. Reuse Integr. (IRI)*, 2013, pp. 22–29.

[28] S. Vikram, C. Yang, and G. Gu, "NOMAD: Towards non-intrusive moving-target defense against Web bots," in *Proc. IEEE Conf. Commun. Netw. Security (CNS)*, 2013, pp. 55–63.

[29] S. Sengupta, A. Chowdhary, D. Huang, and S. Kambhampati, "Moving target defense for the placement of intrusion detection systems in the cloud," in *Proc. Int. Conf. Decis. Game Theory Security*, 2018, pp. 326–345.

[30] S. Achleitner, T. F. La Porta, P. McDaniel, S. Sugrim, S. V. Krishnamurthy, and R. Chadha, "Deceiving network reconnaissance using SDN-based virtual topologies," *IEEE Trans. Netw. Service Manage.*, vol. 14, no. 4, pp. 1098–1112, Dec. 2017.

[31] J. B. Hong, S. Yoon, H. Lim, and D. S. Kim, "Optimal network reconfiguration for software defined networks using shuffle-based online MTD," in *Proc. IEEE Symp. Rel. Distrib. Syst. (SRDS)*, 2017, pp. 234–243.

[32] Y. Huang and A. K. Ghosh, "Introducing diversity and uncertainty to create moving attack surfaces for Web services," in *Moving Target Defense*. Springer, 2011, pp. 131–151.

[33] M. Azab, R. Hassan, and M. Eltoweissy, "ChameleonSoft: A moving target defense system," in *Proc. IEEE Int. Conf. Collaborative Comput. Netw. Appl. Worksharing (CollaborateCom)*, 2011, pp. 241–250.

[34] R. Zhuang, S. Zhang, S. A. DeLoach, X. Ou, and A. Singhal, "Simulation-based approaches to studying effectiveness of moving-target network defense," in *Proc. Nat. Symp. Moving Target Res.*, vol. 246, 2012, pp. 1–12.

[35] A. Nguyen-Tuong, D. Evans, J. C. Knight, B. Cox, and J. W. Davidson, "Security through redundant data diversity," in *Proc. IEEE Int. Conf. Dependable Syst. Netw. FTCS DCC (DSN)*, 2008, pp. 187–196.

[36] M. Taguinod, A. Doupé, Z. Zhao, and G.-J. Ahn, "Toward a moving target defense for Web applications," in *Proc. IEEE Int. Conf. Inf. Reuse Integr.*, 2015, pp. 510–517.

[37] E. Yuan, S. Malek, B. Schmerl, D. Garlan, and J. Gennari, "Architecture-based self-protecting software systems," in *Proc. ACM Conf. Qual. Softw. Archit.*, 2013, pp. 33–42.

[38] Y. Li, R. Dai, and J. Zhang, "Morphing communications of cyber-physical systems towards moving-target defense," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2014, pp. 592–598.

[39] H. Alavizadeh, D. S. Kim, J. B. Hong, and J. Jang-Jaccard, "Effective security analysis for combinations of MTD techniques on cloud computing (short paper)," in *Proc. Int. Conf. Inf. Security Pract. Exp.*, 2017, pp. 539–548.

[40] A. Gorbenko, V. Kharchenko, and A. Romanovsky, "Using inherent service redundancy and diversity to ensure Web services dependability," in *Methods, Models and Tools for Fault Tolerance*. Springer, 2009, pp. 324–341.

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

[42] A. Chowdhary, S. Pisharody, and D. Huang, "SDN based scalable MTD solution in cloud network," in *Proc. ACM Workshop Moving Target Defense*, 2016, pp. 27–36.

[43] 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 Manag.*, vol. 17, no. 3, pp. 1653–1668, Sep. 2020.

[44] D. S. Kim, M. Kim, J.-H. Cho, H. Lim, T. J. Moore, and F. F. Nelson, "Design and performance analysis of software defined networking based Web services adopting moving target defense," in *Proc. IEEE-IFIP Int. Conf. Dependable Syst. Netw. Suppl. Vol. (DSN-S)*, 2020, pp. 43–44.

[45] G. F. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Sunnyvale, CA, USA: Insecure, 2008. [Online]. Available: https://nmap.org/

[46] P. Marbach, "Priority service and max-min fairness," in *Proc. IEEE Comput. Commun. Soc.*, vol. 1, 2002, pp. 266–275.

[47] M. Allalouf and Y. Shavitt, "Maximum flow routing with weighted max-min fairness," in *Quality of Service in the Emerging Networking Panorama*. Springer, 2004, pp. 278–287.

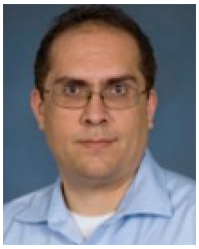
[48] D. Chen and V. Kuehn, "Weighted max-min fairness oriented load-balancing and clustering for multicast cache-enabled F-RAN," in *Proc. IEEE Int. Symp. Turbo Codes Iterative Inf. Process. (ISTC)*, 2016, pp. 395–399.
[49] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, vol. 135. Cambridge, MA, USA: MIT Press, 1998.
[50] M. Plappert *et al.*, "Parameter space noise for exploration," 2017, *arXiv:1706.01905*.
[51] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proc. Python Sci. Conf. (SciPy)*, 2008, pp. 11–15.
[52] A. Hill *et al.* "Stable Baselines." 2018. [Online]. Available: https://github.com/hill-a/stable-baselines
[53] Y. Li and D. Pan, "OpenFlow based load balancing for fat-tree networks with multipath support," in *Proc. Int. Conf. Commun. (ICC)*, Budapest, Hungary, 2013, pp. 1–5.
[54] A. Elmokashfi, A. Kvalbein, and C. Dovrolis, "On the scalability of BGP: The role of topology growth," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 8, pp. 1250–1261, Oct. 2010.

Sunghwan Kim (Member, IEEE) received the B.S. degree in computer science and engineering from Dongguk University, Seoul, Republic of Korea, in 2015, and the M.S. and Ph.D. degrees in electrical engineering and computer science from Gwangju Institute of Science and Technology, Gwangju, Republic of Korea, in 2021. He has been a Staff Engineer with Samsung Research, Samsung Electronics, Seoul, since September 2021. His research interests include communications system, software-defined networking, artificial intelligence, in-network computing, and moving target defense.

Seunghyun Yoon (Member, IEEE) received the B.S. degree in computer science from Handong Global University, Pohang, Republic of Korea, in 2016, and the Ph.D. degree in electrical engineering and computer science from Gwangju Institute of Science and Technology, Gwangju, Republic of Korea, in 2021. He has been a Senior Researcher with the Korea National Engineering Technology Center, Korea Institute of Industrial Technology since April 2021. From August 2019 to February 2020, he was a Visiting Scholar with the Department of Computer Science, Virginia Tech. His research interests include software-defined networking, network resource allocation and optimization, deep reinforcement learning, cybersecurity for various network domains, and moving target defense.

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, where he has been an Associate Professor with the Department of Computer Science since August 2018 and the Director of the Trustworthy Cyberspace Lab. Prior to joining the Virginia Tech, she worked as a Computer Scientist with the U.S. Army Research Laboratory, Adelphi, MD, USA, since 2009. He 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 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, he was selected for the 2013 Presidential Early Career Award for Scientists and Engineers, 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. She is a member of ACM.

Dong Seong Kim (Senior Member, IEEE) received the Ph.D. degree from Korea Aerospace University in February 2008. He has been an Associate Professor of Cybersecurity with The University of Queensland, Australia, since January 2019. He was a Senior Lecturer/Lecturer of 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. From June 2008 to July 2011, he was a Postdoctoral Researcher with Duke University. His research interests are in 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 a Program Co-Chair of IEEE TrustCom2019, IEEE ICIOT2019, ATIS2017, GraMsec2015, IEEE DASC2015 and Program Committee Member of international conferences, including IFIP/IEEE DSN, ISSRE, SRDS, and ICC CISS, respectively.



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 Network Science Division, U.S. Army Research Laboratory. His research interests include sampling theory, constrained statistical inference, stochastic optimization, network security, geometric and topological applications in networks, and network science.

Frederica Free-Nelson is a Researcher and the Program Lead with 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 the lead for the Robust low-level cyber-attack resilience for Military Defense (ROLLCAGE) program working in collaboration with Army Tank Automotive Research, Development and Engineering Center, Office of Naval Research, and Air force Research Laboratory 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 years' combined experience in Cybersecurity Research, Software Engineering, and Program Management within the DoD and other federal services to include the Federal Bureau of Investigation 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 to the International Science Technology (IST-163) Panel – NATO Science & Technology Organization 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 eCybermission 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. He is currently a Full Professor with the AI Graduate School and jointly with the School of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology, Gwangju, Republic of Korea. His research interests include wired and wireless networks, cyber-security, and artificial intelligence.