SCEMA: an SDN-oriented Cost-effective Edge-based MTD Approach

Amir Javadpour, Forough Ja'fari, Tarik Taleb, Mohammad Shojafar and Bin Yang

***Abstract*—Protecting large-scale networks, especially Software-Defined Networks (SDNs), against distributed attacks in a cost-effective manner plays a prominent role in cybersecurity. One of the pervasive approaches to plug security holes and prevent vulnerabilities from being exploited is Moving Target Defense (MTD), which can be efficiently implemented in SDN as it needs comprehensive and proactive network monitoring. The critical key in MTD is to shuffle the least number of hosts with an acceptable security impact and keep the shuffling frequency low. In this paper, we have proposed an SDN-oriented Cost-effective Edge-based MTD Approach (SCEMA) to mitigate Distributed Denial of Service (DDoS) attacks at a lower cost by shuffling an optimized set of hosts that have the highest number of connections to the critical servers. These connections are named *edges* from a graph-theoretical point of view. We have proposed a three-layer mathematical model for the network that can easily calculate the attack cost. We have also designed a system based on SCEMA and simulated it in Mininet. The results show that SCEMA has lower complexity than the previous related MTD field with acceptable performance.**

***Index Terms*—Software-defined networking (SDN), Moving Target Defense (MTD), Distributed Denial of Service (DDoS), Cost-effective, Edge-based Shuffling, Low complexity.**

I. Introduction

SOFTWARE Defined Networks (SDNs) are an evolving trend in computer network technology that effectively improves many network services such as management and monitoring, virtualization, distribution and integration. In SDNs, controlling the network traffic is assigned to a logically centralized component called a *controller*. The controller can create appropriate policies and set related rules on the switches to forward network traffic [1, 2]. However, SDNs are facing different security challenges, among which are Distributed Denial of Service (DDoS) attacks. DDoS attacks are sophisticated and deleterious cyber threats categorized as powerful large-scale distributed attacks [3]. They are becoming bigger and more common for extortion and malicious activities. AWS [4] reported that a DDoS attack was observed in 2020, which was 44% larger than the previously detected ones. Akamai [5] also reported that more than 3000 distinct DDoS attacks were observed only in the gaming industry in a year. These threat reports emphasize an essential need to perform security countermeasures against DDoS attacks.

Moving Target Defense (MTD) is one of the strategies to protect valuable assets from being compromised by DDoS. MTD intends to confuse the adversary by changing the attack space (e.g. by shuffling network addresses) and aims to invalidate the information gathered during network reconnaissance [6]. The advantages of MTD compared to other security mechanisms are (1) their scalability, (2) almost removing the need for threat detection, and (3) frustrating the adversary. Developing a network that can change its configuration and implement MTD methods is challenging. However, as SDN provides a dynamic manageable framework, it is a deserving environment for implementing dynamic security mechanisms [7] such as MTD approaches.

There is a trade-off between implementing a defensive approach and its cost. In some cases, the cost of improving security is too high, which dissuades the network admin from implementing security strategies. An ideal MTD approach keeps the number of reconfigurations and the algorithm complexity low while bringing an acceptable security level [8]. To the best of our knowledge, the execution time of all the previous MTD approaches grows as the network gets larger. This shortcoming was a motivation for us to work on simpler algorithms that can reduce both the number of reconfigurations and the complexity.

We proposed an MTD shuffling algorithm that finds the lowest-cost hosts to compromise and then shuffles them. The feature that helps us find low-cost hosts is the number of connections between the host and the critical servers. Since the connections are modeled with edges in a graph, we call the connections between the hosts and the servers *edges*. Shuffling these important hosts takes lower cost and brings a higher effect. Our proposed method is an SDN-oriented Cost-effective Edge-based MTD Approach, and we call it SCEMA. We have also designed a system that implements SCEMA. The main contributions of this paper are as follows.

1) Introducing a revised model for the networks under DDoS attacks. This model has three layers and uses Petri nets better to show the different states of the critical servers. It also contains mathematical relations

This research work is partially supported by the European Unions Horizon 2020 Research and Innovation Program through the Inspire5GPlus project under Grant Agreement No. 871808, the Academy of Finland 6Genesis project under Grant No. 318927, and the Academy of Finland IDEA-MILL project under Grant No. 352428. **Amir Javadpour** is with the Faculty of Information Technology and Electrical Engineering, University of Oulu (e-mail: a.javadpour87@gmail.com). **Forough Ja'fari** is with the Department of Computer Engineering, Sharif University of Technology, Tehran, Iran (e-mail: azadeh.mth@gmail.com). **Tarik Taleb** is with the Faculty of Information Technology and Electrical Engineering, Oulu University, Oulu, 90570 Finland, and the Department of Computer and Information Security, Sejong University, Seoul, 05006 South Korea (e-mail: talebtarik@gmail.com). **Mohammad Shojafar** is with the 5GIC & 6GIC, Institute for Communication Systems (ICS), University of Surrey, Guildford, GU27XH, United Kingdom (e-mail: m.shojafar@surrey.ac.uk). **Bin Yang** is with the School of Computer and Information Engineering, Chuzhou University, Anhui, 239000 China, and the MOSA!C Lab, 02150 Espoo, Finland (e-mail: yangbinchi@gmail.com).

Corresponding authors: Amir Javadpour and Bin Yang

for computing the attack's cost. In this model, we can effectively defend against the attacks considering the lowest attack cost.
2) Proposing a low-complexity shuffling method, SCEMA, considers the number of connections between the hosts and the servers (i.e., edges) as the main feature of importance. By shuffling the hosts with the highest number of edges, we can reduce the shuffling frequency while keeping the security level high.
3) Theoretically proving that SCEMA can achieve a higher or equal level of security compared with related MTD methods in specific networks.
4) Proposing a system that implements SCEMA and simulating it using Mininet. Two types of scanning methods, sequential and uniform random, are considered in the simulations. We also present the experimental results that show the effectiveness of SCEMA.
5) Presenting related metrics for measuring design goal achievement and comparing SCEMA with the related MTD approaches. The algorithm complexity is the metric for measuring MTD cost, and the adversary's success rate and the rate of the compromised server are the metrics for measuring security level.

The remainder of this paper is as follows. section II reviews the previous works in the field of deploying MTD methods in SDN. In section III the threat model, the adversary's behavior, and the goal of an MTD approach to preventing it are presented. section IV is concerned with the network model and its mathematical representation. section V explains the details of the proposed method and section VI proposes a system architecture that indicates how to implement our proposed method in an SDN environment. section VII represents the numerical results of simulating the proposed system. And, finally, section VIII gives the conclusion.

II. Related Work

In this section, we briefly describe the previous works about using MTD methods in SDN to mitigate cybersecurity attacks. The summary of these works is shown in Table I.

Rawski et al. [9] provided a platform for implementing MTD methods in SDNs. Topology mutation is the MTD technique used in this paper. Steinberger et al. [10] also implemented MTD methods in a collaborative SDN environment that reduces the success rate of a DDoS attack. Luo et al. [11] proposed a combined method of MTD and honeypot to improve network security against DDoS attack. Dynamic virtual IP addresses are assigned to the devices. Macwan and Lung [12] also used virtual IP addresses to hide the real ones through an MTD approach.

Aydeger et al. [13] introduced an optimal MTD strategy to mitigate DDoS attacks. The MTD strategy is modeled as a signaling game. Zhou et al. [14] also proposed a signaling game for defending DDoS attacks in a cost-effective way. Game theory is also used by Zhou et al. [15] and the MTD approach is modeled as a trilateral game. To solve the trade-off problem between MTD cost and its effectiveness, Markov decision processes are employed for adopting the optimal MTD algorithm, which is called TGCESA (Trilateral Game Cost-Effective Shuffling Algorithm).

Narantuya et al. [16] used multiple controllers to improve both the security and the performance of an MTD approach. Each host has several random virtual IP addresses which are altered over time. Karim et al. [17] proposed a random route mutation method to distribute a flow between different paths and make it complicated to find which hosts are in a specific path. Liu et al. [18] proposed a hopping strategy in which the switches change the ports of the packets to confuse the adversary. Chowdhary et al. [19] also employed a port hopping MTD strategy to mitigate multi-stage attacks. The ports of the virtual machines with the highest level of vulnerabilities are changed. Shi et al. [20] proposed a flexible MTD method in which the obfuscation level is variable. Some decoy servers are placed in the network to delay the attacks, and they are all obfuscated using mutation. Debroy et al. [21] proposed a frequency minimization MTD approach to defense cloud-based applications in SDN against DDoS attacks.

Hyder and Ismail [22] used port and IP shuffling to improve the security of SDN. Medina-López et al. [23] used MTD approaches, by which when the messages are exchanged between hosts, their IP address is changed. So, the intermediary hosts are unaware of the real address. Chang et al. [24] proposed a cost-effective MTD method in SDN which randomizes the IP addresses and synchronizes different MTD phases. Chowdhary et al. [25] used an SDN controller to mitigate cloud network attacks through network reconfiguration. The attack graph of network vulnerabilities plays the main role in analyzing the network security level.

A three-tier model called TAG is proposed by Yoon et al. [26], which is used to reduce MTD cost in SDN by finding an optimal set of hosts for shuffling. A greedy Backward Attack Path (BAP) prediction algorithm is proposed in this work to find optimal hosts to shuffle. In BAP, k most vulnerable attack paths from the adversary to the critical servers are selected, and the hosts in these paths are shuffled. The vulnerability of each path is calculated using attack graphs.

Only a few works have considered both MTD cost and DDoS attacks. These works have some limitations, such as being appropriate for only cloud networks with virtual machines and high complexity in game theory and hash-based approaches that may cause delay and processing overhead on the controller. So, we decided to improve one of the mentioned cost-effective works that do not consider DDoS attacks but is capable of being extended and improved. The algorithm and network model proposed by Yoon et al. [26] (BAP and TAG) can potentially be improved for mitigating DDoS attacks with an optimal MTD approach.

III. Threat Model

In a general computer network, there are multiple hosts and critical servers, and the hosts communicate with the servers to use their services. We have considered that the adversary's target is running a DDoS attack on all the critical servers, which are more than one. Our defined threat model assumes

TABLE I
THE SUMMARY OF RELATED WORK

Reference	Main Contribution	Controller	Evaluation Metrics	Cost	DDoS
[9]	Implementing MTD in SDN	FloodLight	Not mentioned	✗	✗
[10]	Proposing a low-cost MTD method in SDN	ONOS	Attack success rate	✗	✓
[11]	Combining MTD and honeypots	RYU	Delay	✗	✓
[12]	Showing the performance of MTD in SDN	RYU	Not mentioned	✗	✗
[13]	Modeling MTD as a signaling game	FloodLight	Cost, packet loss	✓	✓
[14]	Defending DDoS attacks with a signaling game	Ryu	Survival rate, packet loss, delay	✓	✓
TGCESA [15]	Modeling MTD as a novel trilateral game	OpenDayLight	Delay, packet loss, controller load	✓	✓
[16]	Using multiple controllers to improve MTD	ONOS	Delay, attack probability	✗	✗
[17]	Proposing a new path-changing MTD method	Not mentioned	Not mentioned	✗	✗
[18]	Synchronizing hopping policies	POX	Response time, service rate	✗	✓
[19]	Implementing MTD on a real world SDN	OpenDayLight	Threat score, risk value	✓	✗
[20]	Combining MTD and decoys in SDN	FloodLight	Delay, information disclosure	✗	✗
[21]	Minimizing MTD shuffling frequencies in SDN	POX	Packet loss, attack success rate	✓	✓
[22]	Using shadow controllers to protect SDN	ONOS	Defender's success rate	✓	✗
[23]	Detecting Malicious nodes by MTD	RYU	Detection probability	✓	✗
[24]	Synchronizing MTD phases	Not mentioned	Latency, cost	✓	✓
[25]	Using attack graph to for reconfiguration	OpenDayLight	Attack graph cost	✓	✗
BAP [26]	Proposing an attack graph model to reduce MTD cost	ONOS	Delay, complexity, attack success rate	✓	✗
SCEMA	Proposing a low-cost MTD method considering the number of connections	POX	Complexity, attack success rate, compromised servers	✓	✓

that the adversary is an active internal or external intruder, who can probe the hosts and compromise them in order to create an army that launches a DDoS attack against all the critical servers under his/her command.

As it is shown in Figure 5, an insider adversary is located on one of the hosts and connects to the other hosts through the internal connections in the network. This happens when a malicious user is illegally authorized as one of the network members and has complete access to one of the hosts. An external adversary connects to the hosts from outside of the network. This type of adversary can only communicate with the hosts that are permitted to connect to external nodes or the Internet. However, in our threat model, we have considered that all the hosts have Internet access.

The adversary first scans the network to probe the vulnerable hosts and then utilizes various intrusion tools to exploit their vulnerabilities and obtain special privileges to send traffic. When the adversary gains the related privilege in a host, that host becomes compromised and follows his/her commands. After the scanning/probing phase, the adversary's army is created, and he/she can send them an attack command as well as the address of the critical servers that must be targeted.

Before compromising the targets, the adversary scans the network to recognize network topology and to find vulnerable hosts. This is the reconnaissance phase and can be performed in two main methods [7]. *Sequential Scanning* and *Uniform Random Scanning* are different methods commonly used by the adversary to scan the network. In the sequential scanning method, the adversary probes the hosts sequentially in a linear way. All the addresses in the address space are probed one after the other. But in the random scanning method, the hosts are randomly probed. Random addresses within the address space are selected and probed. In the defined threat model, the adversary can perform both sequential and uniform random scanning techniques.

It is worth noting that all the hosts are not directly connected to all the servers in a network. Each server has a specific access list that controls who can communicate with them. On the other hand, the vulnerability levels of the hosts are not the same, and some of them are hard to be compromised. As a result, the adversary attempts to compromise an optimal set of hosts, compromising which is not resource consuming, and moreover, they are in the access list of the target servers. For an MTD approach deployed to prevent this threat, the question is "shuffling which hosts, based on their different features, causes the greatest impact on decreasing the number of critical servers that have become unavailable".

IV. NETWORK MODEL

We have modified TAG model to design a new network model which is more suitable for the networks under DDoS attacks. Our model consists of three layers. The first layer is an undirected graph that shows the connections between different nodes of the network. The second layer is a weighted Directed Acyclic Graph (DAG) which shows the vulnerabilities of the hosts and their exploiting cost. This layer is the combination of the second and third tiers of TAG. The key difference between our model and TAG is in the third layer. The third layer of our model consists of Petri nets for each critical server. Petri net is a principal modeling concept for studying distributed events. It is a directed bipartite graph with two types called places and transitions. All the edges in Petri nets are directed from places to transitions and vice versa. Each place contains zero or more tokens and different system states can be explained as different distributions of tokens among the places. A transition in a Petri net can be fired only if its input places contain a specific number of tokens. By firing a transition, the specified tokens are removed from input places and added to output places. Using Petri nets helps us modeling different states of the critical servers facing DDoS attacks, including safe and dangerous conditions.

According to our model, a network can be modeled as $\mathcal{N} = (\{\mathcal{C}\}, \{\mathcal{V}, \mathcal{H}\}, \{\mathcal{S}\})$, where $\mathcal{C}$ represents the first layer as an

adjacency matrix of the network, $\mathcal{V}$ and $\mathcal{H}$ construct the second layer and indicate the adjacency matrix of vulnerabilities and the set of typical hosts, respectively, and $\mathcal{S}$ is the set of critical servers as the third layer. We assume that we have H typical hosts and S critical servers. So, $\mathcal{H} = \{h_1, h_2, ..., h_H\}$ and $\mathcal{S} = \{s_1, s_2, ..., s_S\}$. h_i indicates the i^{th} host and s_i indicates the i^{th} critical server in the network.

A. First Layer

$\mathcal{C}$ is a symmetric square matrix of size $(H+S+1)$ and the element in its i^{th} row and j^{th} column is indicated by $c_{i,j}$. In general, $c_{i,j}$ is one when c_i and c_j are connected and otherwise it is zero. The explanation of c_i is shown in Equation 1

$$c_i = \begin{cases} \text{The adversary} & \text{if } i = 1 \\ h_{i-1} & \text{if } 1 < i \leq H+1 \\ s_{i-H-1} & \text{otherwise} \end{cases} \quad (1)$$

For example, if $c_{1,2}$ is 1, it means that the adversary is directly connected to the first host in the network. It is worth noting that based on our defined threat model (section III), the value of $c_{1,i}$ for $1 < i \leq H+1$ is always 1, meaning that the adversary can establish a connection with all of the hosts.

B. Second Layer

We assume that there are V vulnerabilities in the network. Some of them are called remote, and they can be exploited directly by the adversary. Some others are the local vulnerabilities, and they can perform DDoS attack to the critical servers when exploited. The other vulnerabilities are intermediary, and they are exploited by remote vulnerabilities to exploit the local ones. All types of vulnerabilities are modeled as a weighted DAG and the edge weights are the cost of successfully exploiting them. If the probability of exploiting a vulnerability is p, we assumed that its exploiting cost is $1-p$. The exploiting cost is fixed for each vulnerability, and can be calculated by Common Vulnerability Scoring System (CVSS) [27].

The logical structure of $\mathcal{V}$ is similar to $\mathcal{C}$. It is a square matrix of size $(V+2)$, and the element in its i^{th} row and j^{th} column is named $v_{i,j}$. The value of $v_{i,j}$ is defined in different cases as follows:

- In the case that $i = 1$ and $1 < j \leq V+1$, $v_{i,j}$ is the probability that the adversary can exploit the $(j-1)^{th}$ vulnerability (v_{j-1}) directly without exploiting other vulnerabilities. Hence, if it is zero, it means that the adversary has to first exploit other vulnerabilities. As an example, if $v_{1,2}$ is 0.5, the adversary can directly exploit v_1 without exploiting any other vulnerabilities with a probability of 0.5.
- In the case that $1 < i \leq V+1$ and $1 < j \leq V+1$, $v_{i,j}$ is the probability of successfully exploiting the $(j-1)^{th}$ vulnerability (v_{j-1}) provided that the $(i-1)^{th}$ vulnerability (v_{i-1}) is currently exploited.
- In the case that $1 < i \leq V+1$ and $j = V+2$, $v_{i,j}$ is the probability of successfully launching a DDoS attack against the critical servers, provided that the $(i-1)^{th}$ vulnerability (v_{i-1}) is currently exploited. For example, if $v_{2,V+2}$ is zero, the adversary cannot launch a DDoS attack only by exploiting v_1.
- In other cases, the value of $v_{i,j}$ is zero, because it indicates an impossible event. For example, the value of $v_{1,V+2}$ is 0, because the adversary is unable to directly launch a DDoS attack without exploiting any of the vulnerabilities.

$\mathcal{V}$ is not a symmetric matrix, and therefore, the values of $v_{i,j}$ and $v_{j,i}$ are not always equal. We have certainly assumed that if a vulnerability can perform a DDoS attack to a specific server, it can also attack the other servers.

Each host, h_i, can be shown as $h_i = \{v_1^i, v_2^i, ..., v_{V_i}^i\}$, where v_j^i is the index of the j^{th} vulnerability that exists in h_i, and $1 \leq v_j^i \leq V$. For example, if $h_1 = \{1, 3\}$, the first host in the network suffers from v_1 and v_3 as the security vulnerabilities.

C. Third Layer

The servers are modeled as Petri nets. Each server has three states. *Safe*, *Warning*, and *Dangerous*. When less than μ neighbor hosts of a server are compromised, the server is in a *Safe* state. If more than μ hosts and less than $\mu+\rho$ neighbor hosts are compromised, the server is in a *Warning* state. If more than $\mu+\rho$ neighbor hosts are compromised, the server is in a *Dangerous* state. According to these states, we have assumed that $\mu < \rho$. These states are equivalent as three places in a Petri net. So, each server, s_i, can be shown as $s_i = (\mathcal{P}, \mathcal{T}, \mathcal{M}_i)$ where $\mathcal{P} = \{P, P', P''\}$ is the set of three places of the Petri net, $\mathcal{T} = \{T, T', T''\}$ is the set of three transitions, and $\mathcal{M}_i$ is the initial marking of s_i. P, P', and P'' are the places in which, the server is in *Safe* state, *Warning* state, and *Dangerous* state, respectively. T is the transition from P to P' and fires when μ neighbor hosts are compromised. The transition from P' to P is T' and it fires when the compromised hosts are recovered by performing a shuffling procedure. T'' is the transition from P and P' to P'' and fires when $\mu+\rho$ neighbor hosts are compromised and can cause a DDoS attack.

For each server i, the initial marking $\mathcal{M}_i$ indicates the number of tokens in each place at the initial state. As all the hosts are initially uncompromised, all of them are in P. So, we have $\mathcal{M}_i = (n_i, 0, 0)$, where n_i is the number of hosts which are directly connected to s_i and are considered as its neighbors. n_i can be calculated as $n_i = \sum_{j=2}^{H+1} c_{i+H+1,j}$.

Each Petri net has an incidence matrix, D. Its rows are associated with the transitions and its columns are associated with the places of the Petri net. The element in the i^{th} row and the j^{th} column shows the number of tokens that are added to the j^{th} place after the i^{th} transition firing. As the value of μ and ρ are considered to be the same for all the servers, D is similar for the servers and is shown in Equation 2.

$$D = \begin{bmatrix} -\mu & +1 & 0 \\ +\mu & -1 & 0 \\ -\rho & -1 & 1 \end{bmatrix} \quad (2)$$

A sample server modeled with a Petri net is shown in Figure 1. Three different states for this server are illustrated, and the value of μ and ρ are 5 and 2, respectively.

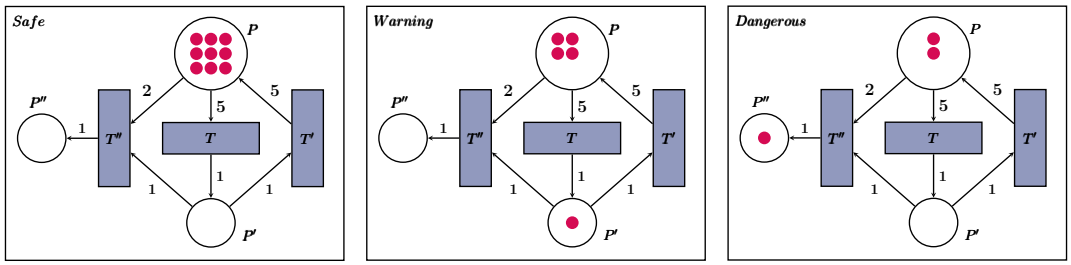

Fig. 1. A sample server shown as the proposed Petri net model. (a) In safe mode. (b) In warning mode. (c) In dangerous mode.

D. Cost Calculations

In a DDoS attack, the adversary selects one or more targets in the network and commands his army to perform the attack against them. The cost of performing a DDoS attack, which we call Attack Cost (AC), includes managing the attack and compromising the army. The cost of compromising the army is the cost of exploiting related vulnerabilities of the hosts. Compromising Cost (CC) is a metric that can guide the defensive method to find the most desirable targets for the adversary's army. We also call the cost of exploiting the vulnerabilities of a host an Exploiting Cost (EC).

Using an MTD strategy is costly, and the cost of shuffle-based approaches includes Implementation Cost (IC) and Shuffling Cost (SC). IC consists of the complexity of executing the defensive strategies. For example, implementing an MTD mechanism in SDN brings IC for the controller and the extra time consumption for the SDN controller is considered as IC. SC is the cost relating to the reconfiguration of the hosts, and it has a direct relation to the number of shuffled hosts. Shuffling the network leads to several configuration changes, and these changes are considered as a cost. While IC and EC are important for evaluating an MTD approach, there is not general rule to theoretically calculate IC and EC. They are commonly measured based on the simulation results or real testbed reports. We have also calculated the complexity order of the proposed method in subsection VI-C. In this section, we focus on theoretically calculating EC, CC, and AC.

We define the EC value of a vulnerability, as the minimum cost that the adversary must pay to exploit that. The costs which are presented in $\mathcal{V}$ are not always the exact EC of a vulnerability. Some vulnerabilities have prerequisites vulnerabilities. So, the adversary must first pay the cost for the prerequisites ones and then exploit that vulnerability. The set of prerequisites vulnerabilities indices for the j^{th} vulnerability in the i^{th} host ($v_{v_j^i}$) is shown as $pr(i,j)$. For example, if $pr(1,1) = \{v_2^1, v_3^1\}$, the adversary must exploit one of v_2 or v_3 to exploit v_1 in the first host. v_k^i is in $pr(i,j)$ if and only if $v_{v_k^i+1,v_j^i+1} \neq 0$. There may be several ways to exploit a vulnerability, but the adversary tries to use the way with the lowest EC. We define $co_{EC}(i,j)$ as the lowest EC of the j^{th} vulnerability of the i^{th} host (v_j^i) to be exploited, and it can be calculated by Equation 3, where $\alpha(i,j) = \min(\bigcup_{v_k^i \in pr(i,j)}\{co_{EC}(i,k)+(1-v_{v_k^i+1,v_j^i+1})\}$ and $\beta(i,j) = \min(\{1 - v_{1,v_j^i+1}, \alpha(i,j)\})$.

$$co_{EC}(i,j) = \begin{cases} \infty, & \text{if } v_{1,v_j^i+1} = 0 \text{ and } pr(i,j) = \phi \\ \alpha(i,j), & \text{if } v_{1,v_j^i+1} = 0 \text{ and } pr(i,j) \neq \phi \\ \beta(i,j) & \text{otherwise} \end{cases} \quad (3)$$

Now the CC of a host can easily be calculated as the minimum cost of the local vulnerabilities of that host. We name the set of local vulnerability indices of h_i as $lo(h_i)$. For example, if $lo(h_1) = \{2,3\}$, the local vulnerabilities of the first host are v_2 and v_3. v_j^i is in $lo(h_i)$ if and only if $v_{v_j^i+1,V+2}$ is not zero. The cost of compromising h_i, $co_{CC}(h_i)$, can be calculated as Equation 4, where $\gamma(i) = \min(\bigcup_{v_j^i \in lo(h_i)}\{co_{EC}(i,j) + (1 - v_{v_j^i+1,V+2})\}$.

$$co_{CC}(h_i) = \begin{cases} \infty & \text{if } lo(h_i) = \phi \\ \gamma(i) & \text{otherwise} \end{cases} \quad (4)$$

Now we can calculate the cost of firing T'' to find out the attack cost (AC). The AC value of a DDoS attack to s_i is equal to the cost of firing T'' in our model. A transition cost can be calculated as the sum of the total cost of its previous transitions and the total cost of its input tokens. T'' can be fired only if T is fired. The cost of T is equal to its input tokens which is compromising μ hosts connected to a single server. In addition to firing T, T'' needs another ρ host to be compromised. So, if the adversary selects a set of $\mu + \rho$ hosts, A_i, which are connected to s_i, the cost of his attack that is firing T'' is shown by $co_{AC}(A_i)$. The adversary's goal is attacking to all the servers, and the final attack cost is $co_{AC}(A)$. We have $co_{AC}(A_i) = \sum_{a \in A_i} co_{CC}(a)$, and $co_{AC}(A)$ can be calculated as Equation 5.

$$co_{AC}(A) = co_{AC}(\bigcup_{i=1}^{S} A_i) \quad (5)$$

V. Proposed Method (SCEMA)

The main problem which we aim to propose a solution for is how to optimally shuffle the hosts to reduce SC while keeping the security level high. In this section, we present the main idea of this research and a numerical example of that. We also explain the proof of its superiority over the previous works.

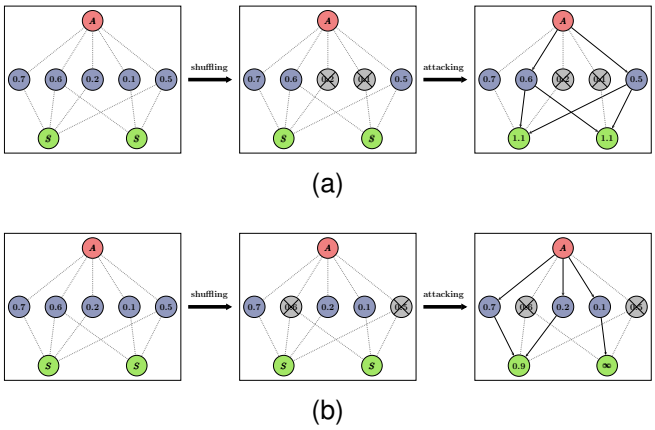

Fig. 2. Comparing the effectiveness of SCEMA and BAP in a sample network. (a) BAP solution: shuffling the hosts regarding their compromising cost (CC). (b) SCEMA solution: shuffling the hosts regarding the number of their connections to the servers.

A. SCEMA Approach

In distributed attacks, such as DDoS, the adversary creates an army of compromised hosts and then sends a command to that army to make all of them perform an attack on a specific target within a specific time interval. Since the adversary tries to perform the attack with the possible lowest AC, he/she searches for the minimal set of hosts which can join his army and which is enough to run the attack. We name the set of selected hosts by the adversary as A. We should find out a metric that can lead to the lowest cost A. Other MTD solutions, such as BAP [26], believe that this metric is the CC value of the host. It is assumed in BAP that the adversary wishes to fill A with the hosts that have the lowest cost of compromising. Hence, they shuffled the hosts with the lowest CC to bring security to the network. The process of finding these hosts is time-consuming and can be more complex in larger networks.

We introduce another metric that can be measured in lower complexity and get acceptable or even better results in many cases. The adversary's willingness to find the minimal army and the behavior of distributed attacks motivate us to design a low-complexity MTD method that shuffles only the hosts which have a higher number of connections to the critical servers. In other words, we believe that the metric which can attract the adversary's attention in many cases is the number of neighbor servers (edges) for each host. In distributed attacks, the group of hosts are more important than the individual ones. Therefore, we should concentrate on the connections between the hosts and the critical servers (i.e., the edges) instead of the CC value of each host. The hosts which are connected to more critical servers are the best targets for the adversary's army. Compromising a host which is connected to three critical servers is much easier than compromising three hosts which are connected to only one server.

Figure 2 shows an example that compares SCEMA and BAP. The cost of compromising each host and performing DDoS attack on each critical server is shown in the nodes. In BAP, the CC value of each host is important, but in SCEMA the number of connections is important. This example illustrates that performing a DDoS attack on all the servers using our defensive method is impossible. However, using BAP can cause an attack.

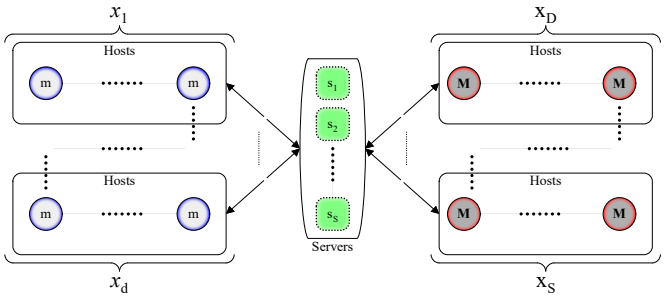

Fig. 3. The general schema of bipartite networks.

We define a shuffling degree for each host. This degree is related to the number of servers that are directly connected to that host. The number of neighbor servers for h_i (i.e., its edges) is shown as $ne(h_i)$, and we have $ne(h_i) = \sum_{j=2+H}^{H+S+1} c_{i+1,j}$. The shuffling degree of h_i (d_i) is calculated as Equation 6. We can say that d_i is the normalized value of $ne(h_i)$.

$$d_i = \frac{ne(h_i)}{\sum_{j=1}^{H} ne(h_j)} \tag{6}$$

B. Proof

In this section, we present a theoretical proof of a theorem that says SCEMA achieves a higher or equal security level compared with BAP. For simplicity in this section, we change the name of A_{BAP} and A_{SCEMA} into A_1 and A_2, respectively, and also use $co()$ instead of co_{AC}.

In homogeneous networks, the hosts are similar and their vulnerabilities are nearly identical. Most of the time, some hosts are more critical, and the network administrator performs security mechanisms to protect them and improve their safety. As a result, these hosts are more connected to the servers and can be used for serious tasks. On the other hand, the remained hosts are more vulnerable and treated as public hosts. We define these types of networks in Definition 1 and call them *bipartite networks*.

Definition 1. *A network, $\mathcal{N}$, is bipartite if all its hosts, $\mathcal{H}$, can be partitioned into two sets, x and X, where all the following conditions are satisfied*

1) $\mathcal{H} = x \cup X$
2) $x \cap X = \phi$
3) $\exists m > 0 : \forall i \in x : co_{CC}(h_i) = m$
4) $\exists M > 0 : \forall i \in X : co_{CC}(h_i) = M$
5) $0 < m < M$
6) $\exists d > 0 : \forall i \in x : 1 \leq ne(h_i) \leq d$
7) $\exists D > 0 : \forall i \in X : D \leq ne(h_i) \leq S$
8) $0 < d < D \leq S$

According to Definition 1, we can write x as $\bigcup_{i=1}^{d} x_i$, where x_i is the set of hosts, such as h, that $ne(h) = i$. We can also write X as $\bigcup_{i=D}^{S} X_S$, where X_i is the set of hosts, such as h, that $ne(h) = i$. The general schema of bipartite networks is shown in Figure 3.

The adversary tries to find the optimal set of host, $A_{adversary}$, that has the lowest attack cost and also has $\mu+\rho$ connections to each server. BAP and SCEMA algorithms try to find $A_{adversary}$ by their own mechanisms. The sets which are selected by BAP and SCEMA can be defined as Definition 2 and Definition 3, respectively.

Definition 2. *A set, A_1, is BAP selected, if all of the following conditions are satisfied.*

1) $A_1 \subseteq x \cup X$
2) $\nexists i < d : x_i \cap A_1 \neq \phi \wedge x_{i+1} \nsubseteq A_1$
3) $\nexists i < S : X_i \cap A_1 \neq \phi \wedge X_{i+1} \nsubseteq A_1$
4) $\nexists i \leq S : X_i \cap A_1 \neq \phi \wedge x \nsubseteq A_1$

Definition 3. *A set, A_2, has SCEMA selected if all of the following conditions are satisfied.*

1) $A_2 \subseteq x \cup X$
2) $\nexists i < S : X_i \cap A_2 \neq \phi \wedge X_{i+1} \nsubseteq A_2$
3) $\nexists i < d : x_i \cap A_2 \neq \phi \wedge x_{i+1} \nsubseteq A_2$
4) $\nexists i \leq d : x_i \cap A_2 \neq \phi \wedge X \nsubseteq A_2$

We have considered two assumptions mentioned in Assumption 1 and Assumption 2.

Assumption 1. *We assume that $\frac{m}{M} \geq \frac{d}{D}$, or in other words, $mD \geq Md$.*

Assumption 2. *We assume that the sum of $ne()$ for all the hosts in A_1 and A_2 are exactly $S(\mu+\rho)$. In other words, we assume that $ne(A_1) = ne(A_2)$.*

The number of connections to each server from hosts in $A_{adversary}$ is greater than or equal to $\mu+\rho$. By Assumption 2, we have assumed that all the servers are connected to exactly $\mu+\rho$ hosts in both A_1 and A_2. There are S servers in the network. So, the value of $ne(A_1)$ and $ne(A_2)$ is $S(\mu+\rho)$.

Both BAP and SCEMA claim that their selected sets are the optimal set which is selected by the adversary ($A_{adversary}$). The number of connections to the servers for A_1 and A_2 is satisfied as mentioned in Assumption 2. But the attack cost is not checked yet. In Theorem 1 we define a theorem that says in bipartite networks, SCEMA is more precise in finding $A_{adversary}$ than BAP.

Theorem 1. *In all bipartite networks under Assumption 1 and Assumption 2, the attack cost of each possible BAP selected set is greater than or equal to each possible SCEMA selected set. In other words, $co(A_1) \geq co(A_2)$.*

To prove Theorem 1, first we define a lemma (Lemma 1) and prove it.

Lemma 1. *If $p \leq q \leq r$ and $p' \leq q' \leq r'$ then $\frac{m}{r}\sum_{i=p}^{q} i|x_i| - \frac{M}{p'}\sum_{i=q'}^{r'} i|X_i| \leq m\sum_{i=p}^{q}|x_i| - M\sum_{i=q'}^{r'}|X_i|$.*

Proof. As r is greater than or equal to all the numbers from p to q, we can say that for all i between p and q, $i \leq r$. By multiplying a positive number, such as $|x_i|$, the inequality remains valid. So, we have $i|x_i| \leq r|x_i|$ for all i between p and q, and we can say that $\sum_{i=p}^{q} i|x_i| \leq \sum_{i=p}^{q} r|x_i|$. Since r is fixed and independent from the values of i, we can say that $\sum_{i=p}^{q} i|x_i| \leq r\sum_{i=p}^{q}|x_i|$. Now multiply both sides of this inequality by a positive number, $\frac{m}{r}$, leads to Equation 7.

$$\frac{m}{r}\sum_{i=p}^{q} i|x_i| \leq m\sum_{i=p}^{q}|x_i| \tag{7}$$

On the other hand, p' is smaller than or equal to all the numbers from q' to r'. So, we can say that for all i between q' and r', $p' \leq i$. By multiplying a positive number, such as $|X_i|$, the inequality remains valid. So, we have $p'|X_i| \leq i|X_i|$ for all i between q' and r', and we can say that $\sum_{i=q'}^{r'} p'|X_i| \leq \sum_{i=q'}^{r'} i|X_i|$. As p' is fixed and independent from the values of i, we obtain $p'\sum_{i=q'}^{r'}|X_i| \leq \sum_{i=q'}^{r'} i|X_i|$. Now we multiply both sides of this inequality by a negative number, $\frac{-M}{p'}$, and change the sign to get Equation 8.

$$\frac{-M}{p'}\sum_{i=q'}^{r'} i|X_i| \leq -M\sum_{i=q'}^{r'}|X_i| \tag{8}$$

Using Equation 7 together with Equation 8, we can easily reach $\frac{m}{r}\sum_{i=p}^{q} i|x_i| - \frac{M}{p'}\sum_{i=q'}^{r'} i|X_i| \leq m\sum_{i=p}^{q}|x_i| - M\sum_{i=q'}^{r'}|X_i|$. □

We also suggest Remark 1 and Remark 2 to better show the steps of the proof. To find $co(A_{adversary})$ we need to find the number of hosts from x that are in $A_{adversary}$ and multiply it by m. Then we have to find the number of hosts from X that are in $A_{adversary}$ and multiply it by M. Finally, by adding up the obtained values, we can reach $co(A_{adversary})$. If y_i and Y_i are $x_i \cap A_{adversary}$ and $X_i \cap A_{adversary}$, respectively, the total number of hosts from x and X are $\sum_{i=1}^{d}|y_i|$ and $\sum_{i=D}^{S}|Y_i|$, respectively. So, the attack cost of $A_{adversary}$ can be calculated as Remark 1.

Remark 1. *If $y_i = x_i \cap A_{adversary}$ and $Y_i = X_i \cap A_{adversary}$, then we have $co(A_{adversary}) = m\sum_{i=1}^{d}|y_i| + M\sum_{i=D}^{S}|Y_i|$.*

The value of $ne(A_{adversary})$ is the sum of $ne(h)$ for all the hosts in $A_{adversary}$. So, if z_i is the set of all the hosts in $A_{adversary}$ that have i connections to the servers, we can say that $ne(A_{adversary}) = \sum_{i=1}^{D} i|z_i|$. Now, we can calculate $ne(A_{adversary})$ as Remark 2.

Remark 2. *If $y_i = x_i \cap A_{adversary}$ and $Y_i = X_i \cap A_{adversary}$, then we have $ne(A_{adversary}) = \sum_{i=1}^{d} i|y_i| + \sum_{i=D}^{S} i|Y_i|$.*

Now we start proving Theorem 1. We consider all possible cases for BAP and SCEMA selected sets, A_1 and A_2, and prove Theorem 1 for each case. If in all possible cases the theorem is proved, we can say that it is completely proved. According to Definition 1, we have only four possible cases as follows.

Case 1. *We have $A_1 = x'_a \cup \bigcup_{i=a+1}^{d} x_i$ and $A_2 = X''_b \cup \bigcup_{i=b+1}^{S} X_i$ where $1 \leq a \leq d$, $x'_a \neq \phi$, $x'_a \subseteq x_a$, $D \leq b \leq S$, $X''_b \neq \phi$, and $X''_b \subseteq X_b$.*

Case 2. *We have $A_1 = x'_a \cup \bigcup_{i=a+1}^{d} x_i$ and $A_2 = X \cup x''_b \cup \bigcup_{i=b+1}^{d} x_i$ where $1 \leq a \leq d$, $x'_a \neq \phi$, $x'_a \subseteq x_a$, $1 \leq b \leq d$, $x''_b \neq \phi$, and $x''_b \subseteq x_b$.*

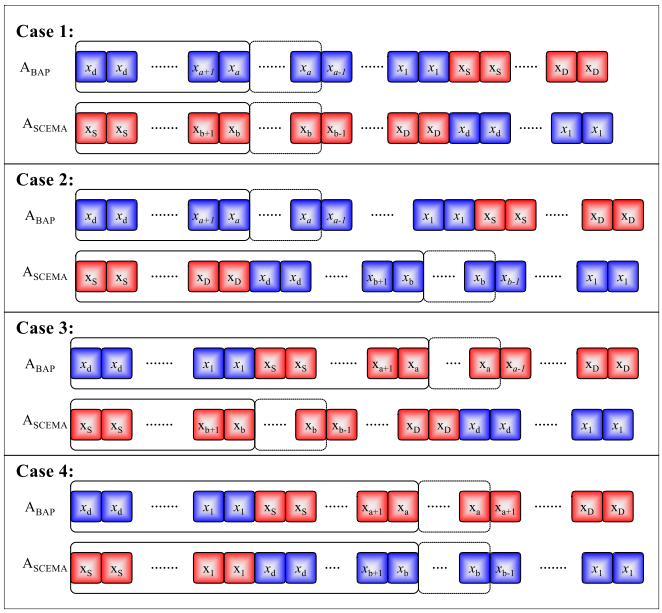

Fig. 4. Possible cases of BAP and SCEMA selected sets in a bipartite network.

Case 3. *We have* $A_1 = x \cup X'_a \cup \bigcup_{i=a+1}^{S} X_i$ *and* $A_2 = X''_b \cup \bigcup_{i=b+1}^{S} X_i$ *where* $D \leq a \leq S$, $X'_a \neq \phi$, $X'_a \subseteq X_a$, $D \leq b \leq S$, $X''_b \neq \phi$, *and* $X''_b \subseteq X_b$.

Case 4. *We have* $A_1 = x \cup X'_a \cup \bigcup_{i=a+1}^{S} X_i$ *and* $A_2 = X \cup x''_b \cup \bigcup_{i=b+1}^{d} x_i$ *where* $D \leq a \leq S$, $X'_a \neq \phi$, $X'_a \subseteq X_a$, $1 \leq b \leq d$, $x''_b \neq \phi$, *and* $x''_b \subseteq x_b$.

The four possible cases are shown Figure 4. We have proved Theorem 1 for all these cases, but only the proof for Case 1 is presented in this section. The other cases are proved in a similar way, and a sketch of their proof is presented in section IX. These proofs demonstrate that SCEMA has a higher or equal security level compared with BAP in all bipartite networks.

Now let us start the proof of Case 1. We have Equation 9 in consequence of Remark 2 and Assumption 2.

$$ne(A_1) = a|x'_a| + \sum_{i=a+1}^{d} i|x_i|, ne(A_2) = b|X''_b| + \sum_{i=b+1}^{S} i|X_i|$$
$$\Rightarrow a|x'_a| + \sum_{i=a+1}^{d} i|x_i| = b|X''_b| + \sum_{i=b+1}^{S} i|X_i| \Rightarrow \quad (9)$$
$$\frac{m}{d}\sum_{i=a+1}^{d} i|x_i| = \frac{mb}{d}|X''_b| - \frac{ma}{d}|x'_a| + \frac{m}{d}\sum_{i=b+1}^{S} i|X_i|$$

Recalling Remark 1, the attack cost of A_1 and A_2 can be calculated as Equation 10.

$$co(A_1) = m(|x'_a| + \sum_{i=a+1}^{d} |x_i|), co(A_2) = M(|X''_b| + \sum_{i=b+1}^{S} |X_i|) \quad (10)$$

Now let $\alpha = co(A_1) - co(A_2)$. If $\alpha \geq 0$, we can say that $co(A_1) \geq co(A_2)$. So, we compare the cost of A_1 and A_2 by subtracting $co(A_2)$ from $co(A_1)$. This subtraction uses Equation 10 and results in Equation 11.

$$\alpha = m(|x'_a| + \sum_{i=a+1}^{d} |x_i|) - M(|X''_b| + \sum_{i=b+1}^{S} |X_i|) \Rightarrow$$
$$\alpha = m|x'_a| + m\sum_{i=a+1}^{d} |x_i| - M|X''_b| - M\sum_{i=b+1}^{S} |X_i| \quad (11)$$

From Lemma 1, we obtain $\frac{m}{d}\sum_{i=a+1}^{d} i|x_i| - \frac{M}{D}\sum_{i=b+1}^{S} i|X_i| \leq m\sum_{i=a+1}^{d}|x_i| - M\sum_{i=b+1}^{S}|X_i|$. Now together with Equation 11 we have Equation 12.

$$\alpha \geq m|x'_a| - M|X''_b| + \frac{m}{d}\sum_{i=a+1}^{d} i|x_i| - \frac{M}{D}\sum_{i=b+1}^{S} i|X_i| \quad (12)$$

Now we can replace the value of $\frac{m}{d}\sum_{i=a+1}^{d} i|x_i|$ in Equation 12 with its value in Equation 9 to obtain Equation 13.

$$\alpha \geq m|x'_a| + \frac{mb}{d}|X''_b| - \frac{ma}{d}|x'_a| + \frac{m}{d}\sum_{i=b+1}^{S} i|X_i| - M|X''_b| +$$
$$\frac{-M}{D}\sum_{i=b+1}^{S} i|X_i| \Rightarrow$$
$$\alpha \geq \frac{md-ma}{d}|x'_a| + \frac{mb-Md}{d}|X''_b| + \frac{mD-Md}{dD}\sum_{i=b+1}^{S} i|X_i| \quad (13)$$

We know that $a \leq d$. So, $ma \leq md$ and $ma - md \geq 0$. So, we obtain Equation 14.

$$\left.\begin{array}{l} md - ma \geq 0 \\ d > 0 \\ |x'_a| \geq 0 \end{array}\right\} \Rightarrow \frac{md-ma}{d}|x'_a| \geq 0 \quad (14)$$

We also know that $b \geq D$. So, $mb \geq mD$. From Assumption 1 we have $mD \geq Md$. Hence, $mb \geq Md$ and $mb - Md \geq 0$. Now we obtain Equation 15.

$$\left.\begin{array}{l} mb - Md \geq 0 \\ d > 0 \\ |X''_b| \geq 0 \end{array}\right\} \Rightarrow \frac{mb-Md}{d}|X''_b| \geq 0 \quad (15)$$

According to Assumption 1, $mD \geq Md$ and $mD - Md \geq 0$. We also know that both d and D are positive and all the values of $|X_i|$ are non-negative. So, we reach Equation 16.

$$\left.\begin{array}{l} mD - Md \geq 0 \\ dD > 0 \\ \sum_{i=b+1}^{S} i|X_i| \geq 0 \end{array}\right\} \Rightarrow \frac{mD-Md}{dD}\sum_{i=b+1}^{S} i|X_i| \geq 0 \quad (16)$$

At last, according to Equation 14, Equation 15, and Equation 16, we find out that the right-hand side of Equation 13 is non-negative. So, we obtain Equation 17.

$$\frac{md-ma}{d}|x'_a| + \frac{mb-Md}{d}|X''_b| + \frac{mD-Md}{dD}\sum_{i=b+1}^{S} i|X_i| \geq 0$$
$$\Rightarrow \alpha \geq 0 \Rightarrow co(A_1) - co(A_2) \geq 0 \Rightarrow co(A_1) \geq co(A_2) \quad \square \quad (17)$$

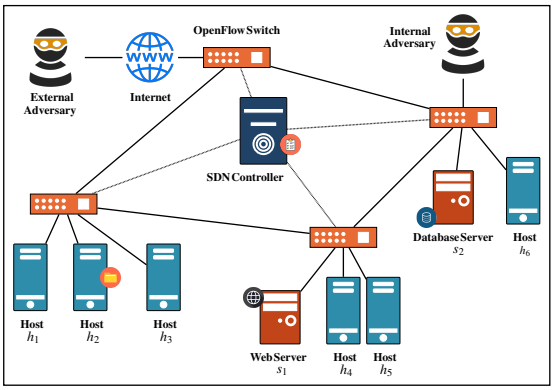

Fig. 5. $\mathcal{N}_\mathcal{E}$ topology with two critical servers and six typical hosts.

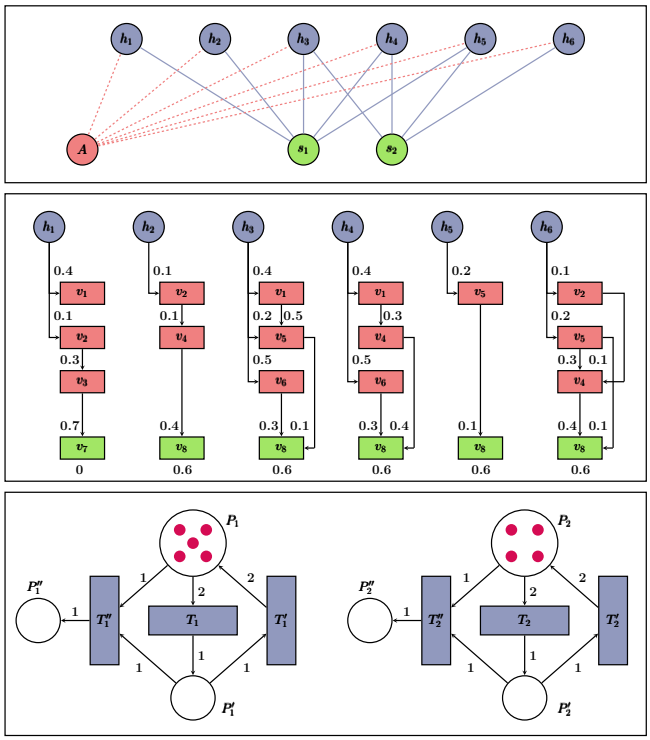

Fig. 6. Proposed model for the numerical example ($\mathcal{N}_\mathcal{E}$).

C. Numerical Example

In this section, we consider a sample software-defined network, $\mathcal{N}_\mathcal{E}$, and present the numerical model for it. The network topology of $\mathcal{N}_\mathcal{E}$ is shown in Figure 5 and its schematic diagram regarding our model is shown in Figure 6. Note that the connection between the hosts is not shown in Figure 6 for simplicity. But the details are in model numeric representation.

Network $\mathcal{N}_\mathcal{E}$ has two servers as $\mathcal{S} = \{s_1, s_2\}$. The first server is $s_1 = (\mathcal{P}, \mathcal{T}, \mathcal{M}_1)$, where $\mathcal{M}_1 = (5, 0, 0)$ and the second server is $s_2 = (\mathcal{P}, \mathcal{T}, \mathcal{M}_2)$, where $\mathcal{M}_2 = (4, 0, 0)$. $\mathcal{C}$ and the relation between the vulnerabilities and their EC is specified in Equation 18 and Equation 19.

$$\mathcal{C} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \tag{18}$$

$$\mathcal{V} = \begin{bmatrix} 0 & .4 & .1 & 0 & 0 & .2 & .5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & .3 & .5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & .3 & .1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & .7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .4 & 0 \\ 0 & 0 & 0 & 0 & .3 & 0 & 0 & 0 & .1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{19}$$

There are six hosts in $\mathcal{N}_\mathcal{E}$ and we have $\mathcal{H} = \{h_1, h_2, h_3, h_4, h_5, h_6\}$. The numerical representation of each host is shown in Equation 20.

$$\begin{aligned} &h_1 = \{1, 2, 3, 7\}, h_2 = \{2, 4, 8\}, h_3 = \{1, 5, 6, 8\}, \\ &h_4 = \{1, 4, 6, 8\}, h_5 = \{5, 8\}, h_6 = \{2, 4, 5, 8\} \end{aligned} \tag{20}$$

We consider that $\mu = 2$ and $\rho = 1$. So, according to Equation 2, D is specified as Equation 21.

$$D = \begin{bmatrix} -2 & +1 & 0 \\ +2 & -1 & 0 \\ -1 & -1 & 1 \end{bmatrix} \tag{21}$$

Using Equation 4 we can calculate the CC value of each host. These costs are shown in Equation 22.

$$\begin{aligned} &co_{CC}(h_1) = \infty, co_{CC}(h_2) = 1.2, co_{CC}(h_3) = 0.9, \\ &co_{CC}(h_4) = 1.4, co_{CC}(h_5) = 0.9, co_{CC}(h_6) = 0.9 \end{aligned} \tag{22}$$

The shuffling degrees of the hosts are calculated according to Equation 6 and are shown in Equation 23.

$$d_1 = \frac{1}{9}, d_2 = \frac{1}{9}, d_3 = \frac{2}{9}, d_4 = \frac{2}{9}, d_5 = \frac{2}{9}, d_6 = \frac{1}{9} \tag{23}$$

BAP suggests selecting the hosts for shuffling among the most vulnerable ones to prevent the attack. So, h_3, h_5, and h_6 are selected. But s_1 has still three unblocked connections. So, another host which is connected to s_1 must be shuffled. As h_2 has the lowest cost, it will be selected. Now the set of hosts for shuffling is $A_{BAP} = \{h_2, h_3, h_5, h_6\}$. But SCEMA selects the hosts with the highest shuffling degree. So, we have $A_{SCEMA} = \{h_3, h_4, h_5\}$ and the hosts in this set can prevent the attack (s_1 and s_2 have less than three unblocked connections). The cost of these two sets, regarding to Equation 5 are shown in Equation 24.

$$\begin{aligned} &co(A_{BAP}) = 1.2 + 0.9 + 0.9 + 0.9 = 3.9, \\ &co(A_{SCEMA}) = 0.9 + 1.4 + 0.9 = 3.2 \end{aligned} \tag{24}$$

We can see that SCEMA finds a lower-cost set of hosts.

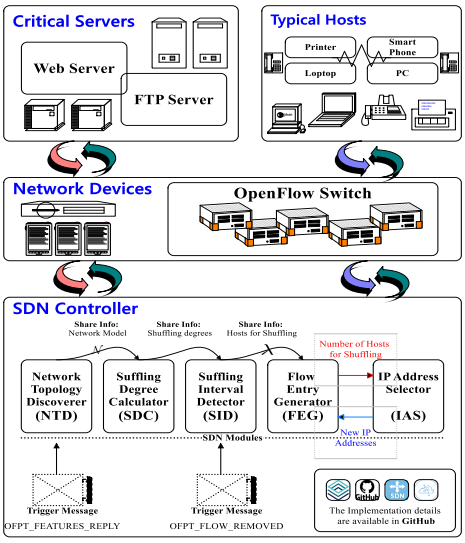

Fig. 7. Proposed system architecture.

VI. System Architecture

We have designed a system in SDN that implements SCEMA. This system, which is shown in Figure 7, contains four main components. Critical servers, typical hosts, network devices, and an SDN controller. Critical servers are the valuable assets in the network and the network admin tries to prevent DDoS attacks against them. The typical hosts are the vulnerable nodes in the network that the adversary attempts to compromise to create his army for performing a DDoS attack. The hosts and the servers are connected through network devices, which are OpenFlow switches in our case. The forwarding rules and management messages are sent to the network devices by an SDN controller. The controller uses five modules to implement SCEMA and manage the network. NTD, SDC, IAS, SID, and FEG. The modules are described as follows.

A. Network Topology Discoverer (NTD)

NTD module uses OpenFlow Discovery Protocol (OFDP) to figure out the current state of the network and its topology. The different network nodes and their connections are found and $\mathcal{C}$ can be generated. The network admin also provides the vulnerabilities and their relations and also the list of critical servers. Finally, the NTD module generates the network model, $\mathcal{N}$, and passes this model to the SDC module. This module is triggered by network startup. Then the network topology is discovered and passed to the SDC module.

B. Shuffling Degree Calculator (SDC)

The SDC module is responsible for finding the shuffling degree of each host in the network. This module gets the network model from the NTD module and generates the shuffling degrees of each host. d_i for every i is calculated in this module using the information about the connection provided in $\mathcal{C}$. The algorithm performed by *shuffling degree calculator* module is shown in Algorithm 1. The list of shuffling degrees is then passed to the SID module.

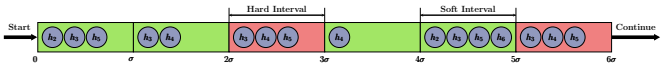

Fig. 8. Shuffled hosts in continuous intervals of SCEMA for $\mathcal{N}_\mathcal{E}$ with $\delta = 3$.

Algorithm 1 SDC module procedure

```
ne ← a list of H zeros                ▷ A list storing ne(h_i) for each host
sum ← 0 ▷ A variable storing the sum of all the members in ne
for i ← 1 to H do ▷ A loop on all the hosts to calculate ne(h_i)
    for j ← 1 to S do
        if c_{i+1,j+H+1} = 1 then          ▷ If there is a connection
            ne[i] ← ne[i] + 1
            sum ← sum + 1
d ← a list of H zeros                 ▷ A list storing d_i for each host
for i ← 1 to H do          ▷ A loop on all the hosts to calculate d_i
    d[i] ← ne[i]/sum
```

C. Shuffling Interval Detector (SID)

SID finds the hosts that must be shuffled, according to SCEMA. The required information is received from the SDC module. All the reconfigurations and shuffling processes are performed at the beginning of a shuffling interval. Each shuffling interval in our system is a fixed period of time and lasts σ seconds. We have proposed two types of shuffling intervals. *Soft* intervals and *Hard* intervals. In *Soft* intervals, each host has a probability of being shuffled which is its shuffling degree. So, h_i is shuffled with a probability of d_i. In *Hard* intervals, all the first $\mu + \rho$ hosts that have the highest value of d_i are shuffled. So, we make sure that all the important hosts are shuffled. Each *Hard* interval comes after $\delta - 1$ *Soft* intervals. By changing the value of δ, we can change the level of security. Figure 8 shows the first six intervals of the sample network mentioned in subsection V-C and the shuffled hosts in each interval are illustrated. The value of δ is three in this example. In *Hard* intervals, three hosts with the highest degree are always shuffled. But in *Soft* intervals, the hosts are with the probability of their shuffling degree. For example, h_2 is shuffled in the first interval but not in the second interval.

A flow entry timeout notifies the SID module about shuffling interval shifting. Hence, IDS checks the type of current interval and generates the set of hosts that have to be shuffled in that interval. We name this set as λ. λ is then passed to the FEG module for setting the related flow entries. The OpenFlow message that indicates flow entry timeout is called *OFPT_FLOW_REMOVED*. The algorithm of the SID module is shown in Algorithm 2.

Algorithm 2 SID module procedure

```
top ← an empty list  ▷ A list storing μ + ρ highest degree hosts
while top has less member than μ + ρ do ▷ A loop to create top
    max ← −1
    for i ← 1 to H do          ▷ Finding host with highest degree
        if i is not in top then
            if max = −1 or d[i] > d[max] then
                max ← i
    add max to top
ints ← 0                ▷ A variable storing the number of intervals
for each shuffling interval do
    ints ← ints + 1
    λ ← an empty list          ▷ A list storing hosts for shuffling
    if ints mod δ = 0 then                          ▷ Hard interval
        for h ∈ top do         ▷ Adding all the hosts in top to λ
            add h to λ
    else                                            ▷ Soft interval
        for i ← 1 to H do
            r ← a random number between 0 and 1
            if r < d[i] then   ▷ The hosts with d_i probability
                add i to λ
```

As long as the network configuration has not changed, the shuffling degrees are fixed, and hence, there is no need for repeating Algorithm 1. This is the same for the first part of Algorithm 2, where the hosts are sorted based on their shuffling degree. The second part of Algorithm 2, where the hosts to be shuffled are selected, is repeated during time intervals. So, there are two parts to the whole procedure of the proposed method. The first fixed part is of $O(S \times H)$, and the second repeated part is of $O(H)$. The procedure of most of the MTD methods can be also divided into the same parts, where the second part is of $O(H)$. In the fixed part, the degrees/scores of the hosts are calculated, and then in the second part, which is repeated in each interval, the hosts to be shuffled are selected. By this division, we can compare the computational complexity of different MTD methods by focusing on the first part. We can say that the IC of SCEMA is $O(S \times H)$. The fixed part of BAP (i.e., its IC) is of $O(S \times k \times o)$, where o is the complexity of finding the vulnerable attack path from the critical server to one of the hosts in the network. The value of o is completely dependent on the network topology and the attack path length. The worst case for BAP is when all the hosts are connected to all the other hosts (i.e., a mesh topology). Since, in this case, for each hop in the attack path, all the hosts are considered, o is H^2, and the total complexity of BAP is $O(S \times k \times H^2)$. In the best case for BAP, the length of the attack path is one, and we have $o = H$. As a result, the best complexity of BAP is $O(S \times k \times H)$, and it is higher than the complexity of SCEMA in any case. Moreover, the complexity of SCEMA is independent of the attack path, k.

D. IP Address Selector (IAS)

IAS module keeps a pool of IP addresses in the network address range. Each address in the pool has a flag that avoids conflicts between the used addresses. When a shuffling process is performed and the hosts need another IP address, the IAS module selects a random address among the addresses in its pool and its flag is not set. The random addresses are passed to the FEG module, and their flag is set.

E. Flow Entry Generator (FEG)

When a shuffling interval is detected by the SID module, the FEG module gets the host information from the SID module and then requests new IP addresses equal to the number of hosts in λ, from the IAS module. Finally, the FEG module generates appropriate flow rules according to the information received from SID and IAS and sets them on network switches.

VII. Evaluation Results

We have compared SCEMA with BAP [26] and TGCESA [15] as they are comparable with SCEMA. But our main focus is on comparing SCEMA with BAP.

A. Evaluation Metrics

Our design goals are reducing the defense cost and retaining network security. So, we need to measure appropriate metrics to clarify high-goal achievement. The selected metrics are described in the following.

1) Algorithm complexity: To measure our algorithm complexity, we have calculated the time required for finding the important hosts. Time complexity and space complexity can be used to measure this metric, such as the IC.

2) End-to-end delay: An efficient security mechanism is one which does not significantly increase the end-to-end delay between the hosts. We have considered end-to-end delay as a metric that can show the SC. Since shuffling a host changes the forwarding paths of the network packets, we expect an extra end-to-end delay when an MTD approach is deployed.

3) Adversary's success rate: The adversary's success rate is the ratio of the number of experiments in which the adversary reaches his goal to the total number of experiments. A lower rate for the adversary's success shows a better security performance in SCEMA.

4) Compromised servers rate: Even though the adversary's success is reached only when all the servers in the network are compromised, the number of compromised servers is also important in measuring the security level of the network. The compromised servers rate can be calculated as the ratio of the number of compromised servers to the total number of servers.

B. Simulation Environment

We have simulated our system, implementing SCEMA, with different network scenarios in *Mininet*. The hosts are connected through *OpenVSwitches* and the switches are controlled by a single *POX* controller. We have used Ubuntu 18.04 operation system, and the simulation machine has 16 G RAM, and an Intel i7 processor running at 3.2 GHz.

We have defined multiple different network topologies, in all of which, the adversary's node is directly connected to all the host nodes. Three of these networks are shown in Figure 9.

The vulnerabilities of the hosts in the first network are shown in Equation 25.

$$h_1 = \{1,2,3\},\ h_2 = \{2,3\},\ h_3 = \{2,4,5\},\ h_4 = \{1,2,4\},\\ h_5 = \{2,3,4\},\ h_6 = \{4,6\}, h_7 = \{4,5,6\},\ h_8 = \{1,2\} \tag{25}$$

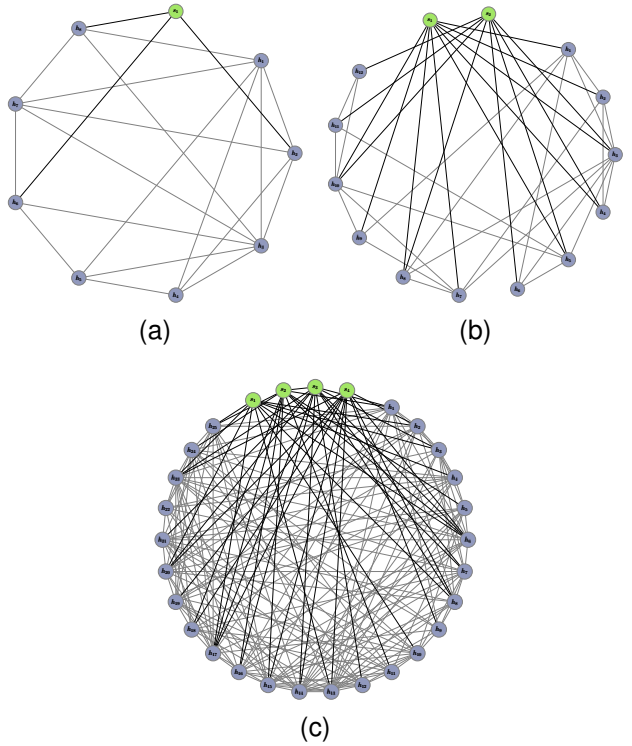

Fig. 9. Three of the simulated networks topologies. (a) A network with a single server. (b) A network with two servers. (c) A network with four servers.

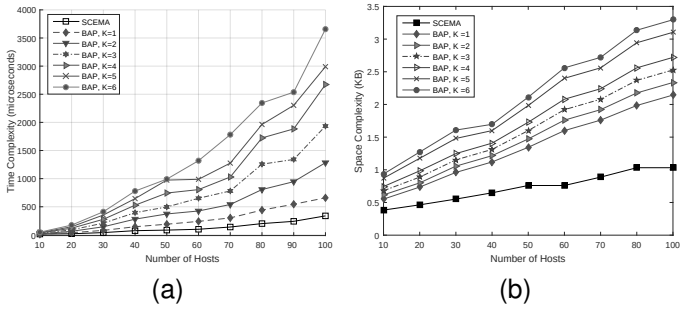

Fig. 10. Comparing SCEMA with BAP regarding their complexity. (a) Time complexity. (b) Space complexity.

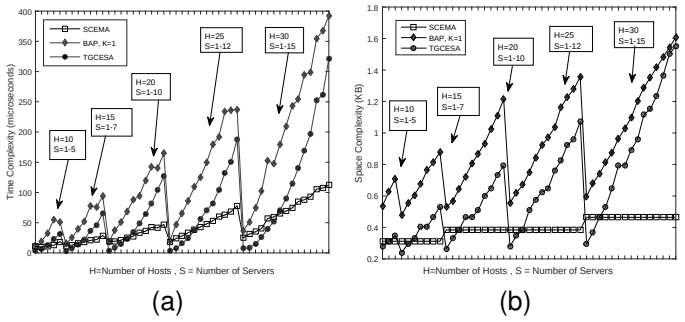

Fig. 11. Comparing the complexity of SCEMA, BAP, and TGCESA. (a) Time complexity. (b) Space complexity.

The first eight hosts in the second simulated network are the same as what is mentioned in Equation 25, and its other hosts are represented in Equation 26.

$$h_9 = \{3,5,6\},\ h_{10} = \{2,6\},\ h_{11} = \{3,4,6\},\ h_{12} = \{1,4,6\} \tag{26}$$

The first 12 hosts in the third network are similar to the second network. The other hosts are mentioned in Equation 27.

$$\begin{aligned} &h_{13} = \{1,2,3\},\ h_{14} = \{2,5\},\ h_{15} = \{1,4,5\},\ h_{16} = \{1,2,4\},\\ &h_{17} = \{2,3,4\},\ h_{18} = \{3,6\},\ h_{19} = \{4,5,6\},\ h_{20} = \{2,4\},\\ &h_{21} = \{1,5,6\},\ h_{22} = \{2,6\},\ h_{23} = \{3,4,6\},\ h_{24} = \{2,4,6\},\\ &h_{25} = \{1\} \end{aligned} \tag{27}$$

The value of $\mathcal{V}$ for all the simulated networks is the same, and it is shown in Equation 28.

$$\mathcal{V} = \begin{bmatrix} 0 & 0.8 & 0.6 & 0 & 0.7 & 0.6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.8 \\ 0 & 0 & 0 & 0.7 & 0.4 & 0 & 0.8 & 0.6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.3 & 0.7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{28}$$

In the simulation scenarios, if one-third of the hosts connected to a critical server is compromised, the adversary can perform a successful DDoS attack against that server. It means that the values of μ and ρ are different in each scenario. The adversary probes five hosts in each scan, and the scanning interval is 15 seconds on average. To prepare a fair condition for comparing different methods with SCEMA, we have considered a fixed number of shuffles in each interval of all the simulation scenarios. We have considered both sequential and uniform random scanning methods in our simulations, based on the defined threat model in section III, to find out how our solution can protect the network against different types of scanning strategies.

C. Simulation Results

The obtained results of each metric mentioned in subsection VII-A are presented in this section.

1) Algorithm complexity: The time and space complexity of executing BAP and SCEMA are shown in Figure 10. In all the cases, the complexity of our proposed algorithm is less than BAP. k is the number of hosts that are shuffled in an interval. The diagram indicates that the time complexity of BAP is markedly increased as both k and network size are increased. But our proposed algorithm is almost independent of the network size. For comparing the complexity of SCEMA, BAP, and TGCESA, all together, we have executed them on different networks. Since the complexity of BAP grows as k gets higher, we have only presented the results for BAP with $k = 1$. TGCESA focuses on shuffling the servers instead of the hosts. So, its complexity gets higher as the number of servers grows. The time and space complexity are shown in Figure 11. We can see that the complexity of BAP and TGCESA grows as the number of servers increases. BAP and TGCESA also become more complex when the number of hosts is increased. The hosts which are connected to the shuffled server must be migrated to another server in TGCESA. So, the growth in TGCESA complexity is reasonable in the case the hosts are growing. The space complexity of SCEMA is not growing

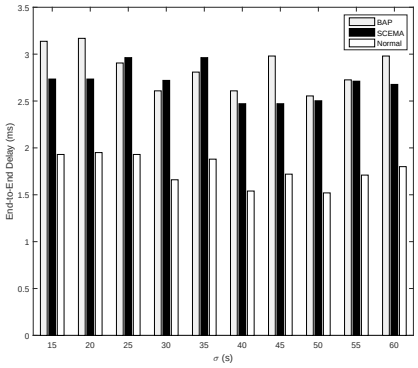

Fig. 12. Comparing the average end-to-end delay in the simulated network deploying no MTD approaches (Normal), SCEMA and BAP.

heavily. Because only a simple array of size $H + S$ can handle its implementation. The time complexity of SCEMA has almost a linear growth.

2) End-to-end delay: The average values of end-to-end delay in all scenarios are shown in Figure 12. As the graph illustrates, when an MTD approach is not deployed, the end-to-end delay is lower than in the cases with shuffling scenarios. However, the point is to consider the trade-off between the end-to-end delay and the security level. BAP and SCEMA cause extra delay, however, the security they bring is acceptable. Moreover, we can see that there is only a small difference between the average delay in BAP and SCEMA scenarios, which indicates that SCEMA does not produce extra delay compared with BAP, and the SC of the proposed method is acceptable.

3) Adversary's success rate: Figure 13 illustrates the adversary's success rate in different scenarios. In the simulated scenarios, the results of which are presented in Figure 13a, the number of shuffled hosts is not the same, and it grows as the number of hosts increases. We have shuffled one-third of the hosts in these scenarios to make some changes in the scale of MTD and the network. So, in the networks with 9 and 48 hosts, the number of shuffled ones is 3 and 16, respectively. We have considered this situation to make the results independent from the shuffled set size. However, the adversary's resources are fixed in the simulation scenarios. Hence, its impact on large-scale networks is low. In other words, in both networks with h and h' hosts, where $h < h'$, the adversary can only probe H hosts. As a result, the army sizes in different networks are almost the same, and in large-scale networks, the army size is too small compared with the network size and has not had enough power to reach the goal. This is why a descending graph in Figure 13a. About the general results, we can say that it is obvious that in a defenseless network, which we call Normal, the adversary's success rate is higher than the cases utilize a defensive method. Moreover, in all the scenarios, the adversary is more successful when he/she probes a network that deploys BAP compared with SCEMA. This demonstrates that SCEMA is effective in reducing the adversary's success rate.

Another point to mention is that the adversary who uses a sequential scanning method may experience higher success in the presence of an MTD mechanism. When a random scanning method is adopted, the adversary is scanning both valid and invalid hosts, and the valid ones are shuffled before the adversary can create a collaborated army.

4) Compromised servers rate: The compromised servers rate is shown in Figure 14. Again, we see that a Normal network (i.e., without any defensive methods) has a higher number of compromised servers compared with the other cases. In addition, even though our goal is not to reduce the number of compromised servers, we can see that this metric also has a lower amount in SCEMA against BAP.

VIII. Conclusion

This paper proposed an SDN-oriented Cost-effective Edge-based MTD Approach, SCEMA, to efficiently mitigate DDoS attacks. SCEMA finds an optimal set of hosts for shuffling to reduce the cost of implementing MTD with acceptable performance. The main idea of SCEMA is to shuffle the hosts with more connections to the critical servers. We propose a three-layer network model to present different security states of the network using Petri nets. We also provide a system architecture that implements SCEMA and simulates this system in Mininet. We observe that SCEMA has lower complexity than previous related MTD methods, and its complexity is independent of the attack path. Thus, it is a cost-effective solution and can easily develop large-scale networks. The results also show that with our approach, the security level is kept high with a low shuffling cost. We plan to extend SCEMA in virtual networks [28]. Virtualization can split the network into small parts and reduce the cost of implementing the MTD approach. Furthermore, virtualization has the potential to confuse the attacker. Hence, implementing SCEMA in a virtual network may lead to gaining a higher security level. Moreover, we have planned to improve SCEMA's performance by focusing on the shuffling intervals in our future research. We can utilize machine learning models in order to find the optimal shuffling intervals. In other words, we planned to answer these two questions in future works on SCEMA using learning approaches: (1) *how frequent the hosts must be shuffled*, and (2) *when the shuffling process must be started.*

IX. Appendix

The steps of proving Theorem 1 for Case 1 are explained in subsection V-B. Proving this theorem for the other cases follows similar steps, and we present a sketch of these proofs in this section. We partition these cases into multiple covering subcases and then prove the theorem for all of these possible cases. Hereafter in this section, α is $co(A_1) - co(A_2)$.

A. The proof for Case 2 when $a = b$

In this case, we have $\alpha = \frac{m}{a}\sum_{i=D}^{S} i|X_i| - M\sum_{i=D}^{S}|X_i|$, and based on Assumption 2, we conclude $|x'_a| - |x''_a| = \frac{1}{a}\sum_{i=D}^{S} i|X_i|$. Hence, we have Equation 29.

$$\alpha \geq \frac{mD - Ma}{aD}\sum_{i=D}^{S} i|X_i| \tag{29}$$

Since α is greater than a non-negative number, it is not negative, and we conclude that $co(A_1) \geq co(A_2)$. □

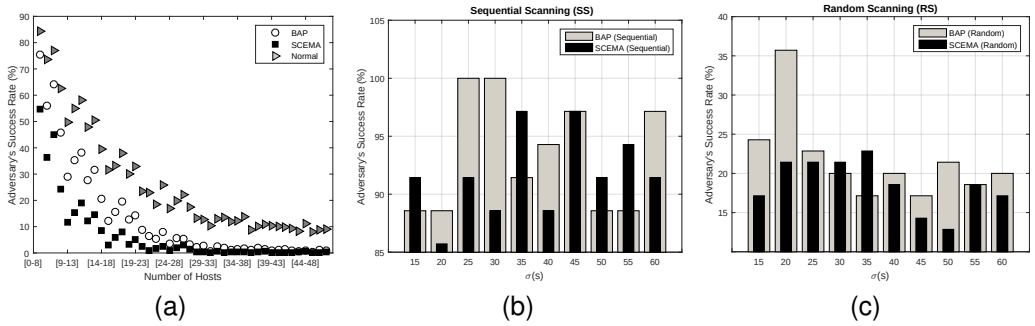

Fig. 13. The evaluation results regarding the adversary's success rate. (a) All scenarios. (b) Sequential scanning. (c) Random scanning.

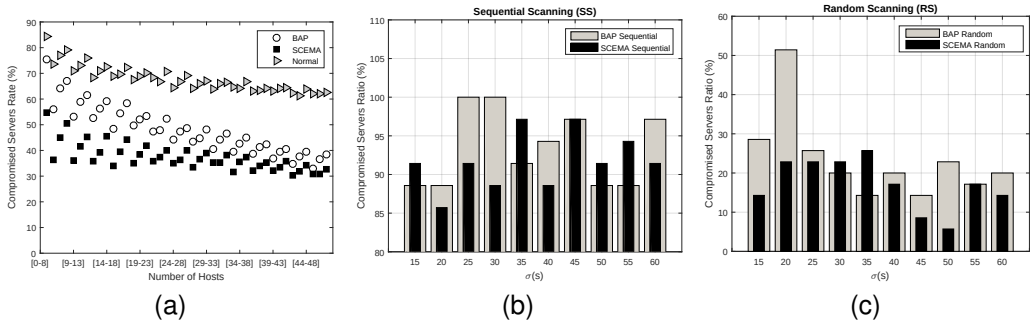

Fig. 14. The evaluation results regarding the rate of the compromised server. (a) All scenarios. (b) Sequential scanning. (c) Random scanning.

B. The proof for Case 2 when $a < b$

There exists a positive number where $b = a + k$. Now, we have two conditions: $k = 1$ and $k > 1$. Now we prove the theorem for both conditions.

When $k = 1$, we have $\alpha = m(|x'_a| - |x''_{a+1}| + |x_{a+1}|) - M\sum_{i=D}^{S}|X_i|$, and based on Assumption 2, we conclude $\sum_{i=D}^{S} i|X_i| = a|x'_a| + (a+1)|x_{a+1}| - (a+1)|x''_{a+1}|$. Hence, we reach Equation 30.

$$\alpha \geq m(|x'_a| - |x''_{a+1}| + |x_{a+1}|) - \frac{M}{D}\sum_{i=D}^{S} i|X_i| \tag{30}$$

Moreover, we know that $x''_{a+1} \subseteq x_{a+1}$. So, $|x_{a+1}| - |x''_{a+1}| \geq 0$, and we obtain Equation 31.

$$\frac{mD - M(a+1)}{D}(|x_{a+1}| - |x''_{a+1}|) \geq 0 \tag{31}$$

The summation of two non-negative numbers is not negative. Hence, we have $\alpha > 0$, and we conclude that $co(A_1) \geq co(A_2)$. □

When $k > 1$, there exists a positive number, q, where $k = q + 1$, and so, $b = a + q + 1$. We have $\alpha = m|x'_a| - m|x''_{a+q+1}| + m\sum_{i=a+1}^{a+q+1}|x_i| - M\sum_{i=D}^{S}|X_i|$, and based on Assumption 2, we have $\sum_{i=D}^{S} i|X_i| = a|x'_a| + \sum_{i=a+1}^{a+q+1} i|x_i| - (a+q+1)|x''_{a+q+1}|$. Hence, we obtain Equation 32.

$$\begin{aligned}\alpha \geq \frac{mD - Ma}{D}|x'_a| + \frac{mD - Md}{dD}\sum_{i=a+1}^{a+q} i|x_i| + \\ \frac{mD - M(a+q+1)}{D}(|x_{a+q+1}| - |x''_{a+q+1}|)\end{aligned} \tag{32}$$

We know that $x''_{a+q+1} \subseteq x_{a+q+1}$ and then, $|x''_{a+q+1}| \leq |x_{a+q+1}|$. So, we obtain Equation 33.

$$\frac{mD - M(a+q+1)}{D}(|x_{a+q+1}| - |x''_{a+q+1}|) \geq 0 \tag{33}$$

Finally, according to the summation of non-negative numbers, we have $\alpha \geq 0$, and hence, $co(A_1) \geq co(A_2)$. □

C. The proof for Case 2 when $a > b$

We prove that $a > b$ is impossible by contradiction. Assume for contradiction that $a > b$ is a possible subcase. If $a > b$, there exists a positive number, k, that $a = b + k$. Based on Assumption 2, we get $(b+k)|x'_{b+k}| = \sum_{i=D}^{S} i|X_i| + \sum_{i=b+1}^{b+k} i|x_i| + b|X''_b|$. As $x'_{b+k} \subseteq x_{b+k}$, we conclude $|x'_{b+k}| \leq |x_{b+k}|$ and so, $(b+k)|x'_{b+k}| \leq (b+k)|x_{b+k}|$. As a result $\sum_{i=b+1}^{b+k} i|x_i|$, that contains $(b+k)|x_{b+k}|$, is greater than or equal to $(b+k)|x'_{b+k}|$. That means $(b+k)|x'_{b+k}| \leq \sum_{i=b+1}^{b+k} i|x_i|$. Since $\sum_{i=D}^{S} i|X_i|$ is non-negative, we conclude Equation 34.

$$\sum_{i=D}^{S} i|X_i| \geq 0 \Rightarrow b|X''_b| \leq 0 \tag{34}$$

But we know that $b > 0$ and $|X''_b| > 0$, because $|X''_b| = 0$ means that $X''_b = \phi$ which is a contradiction to the condition of Case 2). So, $b|X''_b| > 0$, which gives a contradiction to Equation 34. □

D. The proof for Case 3 when $a = b$

According to Assumption 2, we have $\sum_{i=1}^{d} i|x_i| = a(|X''_a| - |X'_a|)$, and moreover, α is $m\sum_{i=1}^{d}|x_i| + M(|X'_a| - M|X''_a|)$. As a result, we conclude Equation 35.

$$\alpha \geq \frac{ma - Md}{d}(|X''_a| - |X'_a|) \tag{35}$$

Again, based on Assumption 2, $a(|X''_a| - |X'_a|)$ is not negative. Consequently, we have $\alpha \geq 0$, and hence, $co(A_1) \geq co(A_2)$. □

E. The proof for Case 3 when $a > b$

There exists a positive number, k, that $a = b + k$. So, we have two conditions: $k = 1$ and $k > 1$. Their proof is as follows.

When $k = 1$, Assumption 2 leads to $\sum_{i=1}^{d} i|x_i| = b|X''_b| + (b+1)|X_{b+1}| - (b+1)|X'_{b+1}|$. On the other hand, the value of

α is $m\sum_{i=1}^{d}|x_i| + M|X'_{b+1}| - M|X''_b| - M|X_{b+1}|$. So, we obtain Equation 36.

$$\alpha \geq \frac{mb - Md}{d}|X''_b| + \frac{m(b+1) - Md}{d}(|X_{b+1}| - |X'_{b+1}|) \quad (36)$$

Since $X_{b+1} \subseteq X'_{b+1}$, we have $|X_{b+1}| \geq |X'_{b+1}|$. So, we conclude that α is non-negative, and $co(A_1) \geq co(A_2)$. □

When $k > 1$, a positive number, say q, exists that $k = q + 1$. We have $\alpha = m\sum_{i=1}^{d}|x_i| - M\sum_{i=b+1}^{b+q}|X_i| - M|X_{b+q+1}| + M|X'_{b+q+1}| - M|X''_b|$, and based on Assumption 2, we have $\sum_{i=1}^{d} i|x_i| = b|X''_b| - (b+q+1)|X'_{b+q+1}| + \sum_{i=b+1}^{b+q+1} i|X_i|$. Combining these two statements, we obtain Equation 37.

$$\alpha \geq \frac{mb - Md}{d}|X''_b| + \frac{mD - Md}{dD}\sum_{i=b+1}^{b+q} i|X_i| + \frac{m(b+q+1) - Md}{d}(|X_{b+q+1}| - |X'_{b+q+1}|) \quad (37)$$

We know that $X'_{b+q+1} \subseteq X_{b+q+1}$, and so, $|X_{b+q+1}| - |X'_{b+q+1}| \geq 0$. Consequently, α is the summation of three non-negative numbers, and hence $co(A_1) \geq co(A_2)$. □

F. The proof for Case 3 when $a < b$

We prove that $a < b$ is impossible. Assume for contradiction that $a < b$ is a possible subcase. So, there exists a positive number, k, where $b = a + k$. Based on Assumption 2, we have $(a+k)|X''_{a+k}| = \sum_{i=1}^{d} i|x_i| + a|X'_a| + \sum_{i=a+1}^{a+k} i|X_i|$. As $X''_{a+k} \subseteq X_{a+k}$, we obtain $(a+k)|X''_{a+k}| \leq (a+k)|X_{a+k}|$. As a result, $(a+k)|X''_{a+k}|$ is smaller than or equal to $\sum_{i=a+1}^{a+k} i|X_i|$ which contains $(a+k)|X_{a+k}|$. Hence, we have $(a+k)|X''_{a+k}| \leq \sum_{i=a+1}^{a+k} i|X_i|$. Again from Assumption 2, we obtain $\sum_{i=1}^{d} i|x_i| + a|X'_a| \leq 0$. Since $\sum_{i=1}^{d} i|x_i|$ is non-negative, we reach Equation 38.

$$\sum_{i=1}^{d} i|x_i| \geq 0 \Rightarrow a|X'_a| \leq 0 \quad (38)$$

But since $X'_a \neq \phi$ according to Case 3, $a|X'_a| > 0$, and it gives a contradiction to Equation 38. □

G. The proof for Case 4 when $a = D$ and $b = 1$

According to Assumption 2, we have $|x_1| - |x''_1| = D(|X_D| - |X'_D|)$. Moreover, α is $(mD - M)(|X_D| - |X'_D|)$. Now, since $X'_D \subseteq X_D$, α is the multiplication of two non-negative numbers, and hence, $co(A_1) \geq co(A_2)$. □

H. The proof for Case 4 when $a = D$ and $b > 1$

There exists a positive number, k, where $b = k + 1$. We have also $\alpha = m\sum_{i=1}^{k}|x_i| - \frac{M}{D}\sum_{i=1}^{k} i|x_i| + \frac{mD - M(k+1)}{D}(|x_{k+1}| - |x''_{k+1}|)$. Considering Assumption 2, we have $|X_D| - |X'_D| = \frac{1}{D}\sum_{i=1}^{k+1} i|x_i| - \frac{k+1}{D}|x''_{k+1}|$. Consequently, we obtain Equation 39.

$$\alpha \geq \frac{mD - Md}{dD}\sum_{i=1}^{k} i|x_i| + \frac{mD - M(k+1)}{D}(|x_{k+1}| - |x''_{k+1}|) \quad (39)$$

Since, $x''_{k+1} \subseteq x_{k+1}$, α is the summation of non-negative numbers, and so, $co(A_1) \geq co(A_2)$. □

I. The proof for Case 4 when $a > D$ and $b = 1$

We can find a positive number, say k, where $a = D + k$. The value of α is $m\sum_{i=D}^{D+k-1} i|X_i| - M\sum_{i=D}^{D+k-1}|X_i| + (m(D+k) - M)|X_{D+k}| + (M - m(D+k))|X'_{D+k}|$. On the other hand, based on Assumption 2, we have $|x_1| - |x''_1| = \sum_{i=D}^{D+k} i|X_i| - (D+k)|X'_{D+k}|$. As a result, we can conclude Equation 40.

$$\alpha \geq \frac{mD - M}{D}\sum_{i=D}^{D+k-1} i|X_i| + \frac{m(D+k) - M}{D}(|X_{D+k} - X''_{D+k}|) \quad (40)$$

Since $X''_{D+k} \subseteq X_{D+k}$, the value of α is not negative, and consequently $co(A_1) \geq co(A_2)$. □

J. The proof for Case 4 when $a > D$ and $b > 1$

We can find a positive number, such as k, where $a = D + k$. On the other hand, since $b > 1$, there exists a positive number, say q, where $b = q + 1$. Due to Assumption 2, we have $\sum_{i=D}^{D+k-1} i|X_i| = \sum_{i=1}^{q+1} i|x_i| - (q+1)|x''_{q+1}| + (D+k)(|X'_{D+k}| - |X_{D+k}|)$. On the other hand, the value of α in this case is $\alpha = M|X'_{D+k}| - m|x''_{q+1}| + m|x_{q+1}| - M|X_{D+k}| + m\sum_{i=1}^{q}|x_i| - M\sum_{i=D}^{D+k-1}|X_i|$. Therefore, we can obtain Equation 41.

$$\alpha \geq \frac{mD - M(q+1)}{D}(|x_{q+1}| - |x''_{q+1}|) + \frac{Mk}{D}(|X_{D+k}| - |X'_{D+k}|) + \frac{mD - Md}{dD}\sum_{i=1}^{q} i|x_i| \quad (41)$$

We know that $x''_{q+1} \subseteq x_{q+1}$ and $X'_{D+k} \subseteq X_{D+k}$. Hence, α is the summation of positive values, and consequently, $co(A_1) \geq co(A_2)$. □

REFERENCES

[1] A. Javadpour, "Providing a way to create balance between reliability and delays in sdn networks by using the appropriate placement of controllers," *Wireless Personal Communications*, vol. 110, no. 2, pp. 1057–1071, 2020.

[2] A. Javadpour and G. Wang, "ctmvsdn: improving resource management using combination of markov-process and tdma in software-defined networking," *The Journal of Supercomputing*, vol. 78, no. 3, pp. 3477–3499, 2022.

[3] M. Cirillo, M. Di Mauro, V. Matta, and M. Tambasco, "Botnet identification in ddos attacks with multiple emulation dictionaries," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 3554–3569, 2021.

[4] AWS, "Aws shield threat landscape report," https://aws-shield-tlr.s3.amazonaws.com/2020-Q1_AWS_Shield_TLR.pdf, 2020, [Accessed: March 2021].

[5] Akamai, "State of the internet," https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/soti-security-gaming-you-cant-solo-security-report-2020.pdf, 2020, [Accessed: March 2021].

[6] 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 Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 709–745, 2020.

[7] F. Ja'fari, S. Mostafavi, K. Mizanian, and E. Jafari, "An intelligent botnet blocking approach in software defined networks using honeypots," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 2, pp. 2993–3016, 2021.

[8] J. Zheng and A. S. Namin, "A survey on the moving target defense strategies: An architectural perspective," *Journal of Computer Science and Technology*, vol. 34, no. 1, pp. 207–233, 2019.

[9] M. Rawski, S. Kukliński, P. Sapiecha, M. Pelka, G. Przytuła, P. Wojs, K. Szczypiorski *et al.*, "Mmtd: Mano-based moving target defense for corporate networks," in *2020 World Conference on Computing and Communication Technologies (WCCCT)*. IEEE, 2020, pp. 79–87.

[10] J. Steinberger, B. Kuhnert, C. Dietz, L. Ball, A. Sperotto, H. Baier, A. Pras, and G. Dreo, "Ddos defense using mtd and sdn," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2018, pp. 1–9.

[11] X. Luo, Q. Yan, M. Wang, and W. Huang, "Using mtd and sdn-based honeypots to defend ddos attacks in iot," in *2019 Computing, Communications and IoT Applications (ComComAp)*. IEEE, 2019, pp. 392–395.

[12] S. Macwan and C.-H. Lung, "Investigation of moving target defense technique to prevent poisoning attacks in sdn," in *2019 IEEE World Congress on Services (SERVICES)*, vol. 2642. IEEE, 2019, pp. 178–183.

[13] A. Aydeger, M. H. Manshaei, M. A. Rahman, and K. Akkaya, "Strategic defense against stealthy link flooding attacks: A signaling game approach," *IEEE Transactions on Network Science and Engineering*, 2021.

[14] Y. Zhou, G. Cheng, and S. Yu, "An sdn-enabled proactive defense framework for ddos mitigation in iot networks," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 5366–5380, 2021.

[15] Y. Zhou, G. Cheng, S. Jiang, Y. Zhao, and Z. Chen, "Cost-effective moving target defense against ddos attacks using trilateral game and multi-objective markov decision processes," *Computers & Security*, vol. 97, p. 101976, 2020.

[16] 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 *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks–Supplemental Volume (DSN-S)*. IEEE, 2019, pp. 15–16.

[17] Z. Karim, A. Sebbar, Y. Baddi, and M. Boulmalf, "Secure multipath mutation smpm in moving target defense based on sdn," *Procedia Computer Science*, vol. 151, pp. 977–984, 2019.

[18] Z. Liu, Y. He, W. Wang, S. Wang, X. Li, and B. Zhang, "Aeh-mtd: Adaptive moving target defense scheme for sdn," in *2019 IEEE International Conference on Smart Internet of Things (SmartIoT)*. IEEE, 2019, pp. 142–147.

[19] A. Chowdhary, A. Alshamrani, D. Huang, and H. Liang, "Mtd analysis and evaluation framework in software defined network (mason)," in *Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, 2018, pp. 43–48.

[20] Y. Shi, H. Zhang, J. Wang, F. Xiao, J. Huang, D. Zha, H. Hu, F. Yan, and B. Zhao, "Chaos: An sdn-based moving target defense system," *Security and Communication Networks*, vol. 2017, 2017.

[21] S. Debroy, P. Calyam, M. Nguyen, R. L. Neupane, B. Mukherjee, A. K. Eeralla, and K. Salah, "Frequency-minimal utility-maximal moving target defense against ddos in sdn-based systems," *IEEE Transactions on Network and Service Management*, 2020.

[22] M. F. Hyder and M. A. Ismail, "Securing control and data planes from reconnaissance attacks using distributed shadow controllers, reactive and proactive approaches," *IEEE Access*, vol. 9, pp. 21 881–21 894, 2021.

[23] C. Medina-López, L. Casado, V. González-Ruiz, and Y. Qiao, "An sdn approach to detect targeted attacks in p2p fully connected overlays," *International Journal of Information Security*, pp. 1–11, 2020.

[24] S.-Y. Chang, Y. Park, and B. B. A. Babu, "Fast ip hopping randomization to secure hop-by-hop access in sdn," *IEEE Transactions on Network and Service Management*, vol. 16, no. 1, pp. 308–320, 2018.

[25] A. Chowdhary, S. Pisharody, and D. Huang, "Sdn based scalable mtd solution in cloud network," in *Proceedings of the 2016 ACM Workshop on Moving Target Defense*, 2016, pp. 27–36.

[26] 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 Transactions on Network and Service Management*, vol. 17, no. 3, pp. 1653–1668, 2020.

[27] P. Johnson, R. Lagerström, M. Ekstedt, and U. Franke, "Can the common vulnerability scoring system be trusted? a bayesian analysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 6, pp. 1002–1015, 2016.

[28] A. Javadpour, "Improving resources management in network virtualization by utilizing a software-based network," *Wireless Personal Communications*, vol. 106, no. 2, pp. 505–519, 2019.

Amir Javadpour obtained his MSc degree in Medical Information Technology Engineering from University of Tehran, Iran, in 2014. From Guangzhou University, China, he received a PhD in Computer Science/Mathematics/Cybersecurity. In addition, he has published papers with his colleagues in highly ranked journals and several ranked conferences on several topics, including Cloud Computing, Software-Defined Networking (SDN), Big Data, Intrusion Detection Systems (IDS), and the Internet of Things (IoT), Moving Target Defence (MTD), Machine Learning (ML) and optimisation algorithms. Additionally, he reviewed papers for several reputable venues such as IEEE Transactions on Cloud Computing, IEEE Transactions on Network Science and Engineering, ACM Transactions on Internet Technology, the Journal of Supercomputing, several journals of Springer and Elsevier, etc. He is a Technical Program Committee (TCP) Member of various conferences.

Forough Ja'fari is a Senior Researcher in cybersecurity and computer science. She received her Bachelor's degree from Sharif University of Technology and her Master's degree in Computer Network Engineering from Yazd University, Iran. She is a visiting scholar researcher at Guangzhou University, China. Cloud computing, software-defined Networking (SDN), cyber deception, Intrusion Detection Systems (IDS), Internet of Things (IoT), Moving Target Defence (MTD), and Machine Learning are some of her research interests. She is currently a Guest Editor (GE) of Cluster Computing (CLUS) Journal and a reviewer for several journals and conferences.

Tarik Taleb (Senior Member, IEEE) is currently a Professor at the Center of Wireless Communications, The University of Oulu, Finland. He is the founder and director of the MOSA!C Lab (www.mosaic-lab.org). He has been a Professor at the School of Electrical Engineering, Aalto University, Finland. Before that, he worked as a Senior Researcher and 3GPP Standards Expert at NEC Europe Ltd, Heidelberg, Germany. He then led the NEC Europe Labs Team working on R&D projects on carrier cloud platforms. Before joining NEC until Mar. 2009, he worked as an assistant professor at the Graduate School of Information Sciences, Tohoku University, Japan, in a lab fully funded by KDDI, the second-largest mobile operator in Japan. From Oct. 2005 till Mar. 2006, he worked as a research fellow at the Intelligent Cosmos Research Institute, Sendai, Japan. He received his B. E degree in Information Engineering with distinction, M.Sc. and PhD degrees in Information Sciences from Tohoku University in 2001, 2003, and 2005, respectively. His research interests lie in the telco cloud, network softwarization & network slicing, AI-based software-defined security, immersive communications, mobile multimedia streaming, & next-generation mobile networking. He has also been directly engaged in the development and standardization of the Evolved Packet System as a member of 3GPP's System Architecture working group 2. Prof. Taleb served on the IEEE Communications Society Standardization Program Development Board. As an attempt to bridge the gap between academia and industry, Prof. Taleb founded the "IEEE Workshop on Telecommunications Standards: from Research to Standards", a successful event that got awarded the "best workshop award" by the IEEE Communication Society (ComSoC). Based on the success of this workshop, Prof. Taleb has also founded and has been the steering committee chair of the IEEE Conf. on Standards for Communications and Networking.

Mohammad Shojafar (Senior Member, IEEE) is a Senior Lecturer (Associate Professor) in network security and an Intel Innovator, a Professional ACM member and ACM Distinguished Speaker, a Fellow of the Higher Education Academy, and a Marie Curie Alumni, working in the 5G & 6G Innovation Centre (5GIC & 6GIC), Institute for Communication Systems (ICS), at the University of Surrey, UK. Before, he was a Senior Researcher and a Marie Curie Fellow in the SPRITZ Security and Privacy Research group at the University of Padua, Italy. Dr Mohammad secured £310k for the ESKMARALD project funded by GCHQ, UK, in 2022. Also, he is a PI of AUTOTRUST, a secure autonomous 5G-based traffic management platform the European Space Agency supported for around €750k in 2021. Also, Mohammad was a PI of the PRISENODE project, a €275k Horizon 2020 Marie Curie project in network security and Fog task/resource scheduling collaborating at the University of Padua. He also was a PI on an Italian SDN security and privacy (€60k) supported by the University of Padua in 2018 and a Co-PI on an Ecuadorian-British project on IoT and Industry 4.0 resource allocation ($20k) in 2020. He contributed to some Italian projects in telecommunications, like GAUChO, SAMMClouds, and SC2. He received his PhD in ICT from Sapienza University of Rome, Rome, Italy, in 2016 with an "Excellent" degree. He is an Associate Editor in *IEEE Transactions on Network and Service Management*, *IEEE Transactions on Intelligent Transportation Systems*, *IEEE Consumer Electronics Magazine*, *IEEE Systems Journal* and Computer Networks. For additional information: http://mshojafar.com

Bin Yang Bin Yang received his PhD in systems information science from Future University Hakodate, Japan, in 2015. He is a Professor at the School of Computer and Information Engineering, Chuzhou University, China, and a Senior Researcher with MOSA!C Lab, Finland. His research interests include unmanned aerial vehicle networks, cyber security and the Internet of Things.