A Cost-Effective MTD Approach for DDoS Attacks in Software-Defined Networks

Amir Javadpour
Faculty of Information Technology and Electrical Engineering
University of Oulu
Oulu, 90570 Finland
(e-mail: a.javadpour87@gmail.com)

Forough Ja'fari
Department of Computer Engineering
Sharif University of Technology
Tehran, Iran
(e-mail: azadeh.mth@gmail.com)

Tarik Taleb
Faculty of Information Technology and Electrical Engineering
University of Oulu
Oulu, 90570 Finland
(e-mail: talebtarik@gmail.com)

Mohammad Shojafar
5GIC & 6GIC, Institute for Communication Systems (ICS)
University of Surrey
Guildford, GU27XH, United Kingdom
(e-mail: m.shojafar@surrey.ac.uk)

***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 with a lower cost by shuffling an optimized set of hosts have the highest number of connections to the critical servers. These connections are named *edges* from a graph-theoretical point of view. We have designed a system based on SCEMA and simulated it in Mininet. The results show that SCEMA has lower (52.58%) complexity than the previous related MTD methods with improving the security level by 14.32%.**

***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. Controlling the network traffic is assigned to a logically centralized component called controller. The controller can create appropriate policies and set related rules on the switches to forward network traffic [1]. However, SDNs are facing different security challenges among which are Distributed Denial of Service (DDoS) attacks. DDoS attacks are sophisticated and deleterious cyber threats that are categorized as powerful large-scale distributed attacks. They are becoming bigger and more common for extortion and malicious activities [2]. Hence, there is 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 [3]. The advantages of MTD compared to other security mechanisms are (1) their scalability, (2) almost removing the need 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 mechanism [4] such as MTD approaches.

Developing an MTD method must be cost-effective. There is a trade-off between implementing a defensive approach and its cost. The major part of MTD cost is related to the number of reconfigurations and the MTD algorithm complexity. However, the previous works in this field focused on complicated network features to reduce the number of reconfigurations. To the best of our knowledge, all of them failed to reduce the method complexity, and their execution time grows as the network gets larger.

We proposed an MTD shuffling method that finds the lowest-cost hosts to compromise and then shuffle 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 attempt to find the best hosts for shuffling that can bring more security against DDoS attacks. The main contributions of this paper are as follows.

1) Proposing a low-complexity shuffling method, SCEMA, that considers the number of connections between the

TABLE I
THE SUMMARY OF RELATED WORK

Reference	Evaluation Metrics	Cost	DDoS
[5]	Attack success rate	✗	✓
[6]	Delay	✗	✓
[7]	Cost, packet loss	✓	✓
TGCESA [8]	Delay, packet loss, CPU load	✓	✓
[9]	Delay, attack probability	✗	✗
[10]	Response time, service rate	✗	✓
[11]	Threat score, service risk value	✓	✗
[12]	Delay, information disclosure	✗	✗
[13]	Packet loss, attack success rate	✓	✓
[14]	Defender's success rate	✓	✗
[15]	Detection probability	✓	✗
[16]	Latency, reconnaissance cost	✓	✓
[17]	Attack graph generation time	✓	✗
BAP [18]	Delay, complexity, success rate	✓	✗
SCEMA	Complexity, adversary's success rate, compromised servers rate	✓	✓

hosts and the servers (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.

2) Proposing a system that implements SCEMA and simulating it using Mininet. Multiple different network topologies with several scenarios are considered in the simulations. We also present the experimental results that shows the effectiveness of SCEMA.
3) Presenting related metrics for measuring design goal achievement and comparing SCEMA with two other related MTD approaches. The shuffling algorithm complexity is the metric for measuring MTD cost and the adversary's success rate and the compromised servers rate are for measuring security level.

The remainder of this paper is as follows. section II consists of the previous researches about the MTD methods. section III explains the details of the proposed method, SCEMA, and section IV proposes a system architecture that indicates how to implement SCEMA in an SDN environment. section V represents the evaluation results of simulating the proposed system. And, finally, section VI 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.

Steinberger et al. [5] implemented an MTD method with the goal of showing that the success rate of DDoS attacks can be decreased by using MTD in SDN. Luo et al. [6] proposed a combined method of MTD and honeypot to improve network security over DDoS attack in SDN. Aydeger et al. [7] introduced an optimal MTD strategy to mitigate DDoS attacks. The MTD strategy is modeled as a signaling game. A similar gaming concept is used by Zhou et al. [8] 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. [9] used multiple SDN controllers in large-scale networks to improve both the security and the performance of an MTD approach.

Liu et al. [10] proposed a hopping strategy in which the switches change source and destination ports of the packets to confuse the adversary and prevent him from launching a DDoS attack. Chowdhary et al. [11] also employed a port hopping MTD strategy to mitigate multi-stage attacks. Shi et al. [12] proposed a flexible MTD method in which the obfuscation level is variable. Debroy et al. [13] proposed a frequency minimization MTD approach to secure SDN applications against DDoS attacks.

Hyder and Ismail [14] used MTD to improve both control and data plane security in SDN. Port shuffling and IP shuffling are employed to prevent reconnaissance attacks in data plane. Medina-López et al. [15] used MTD to find malicious nodes in the peer to peer overlay SDN. When messages are exchanged between peers, their destination IP address is changed. Chang et al. [16] proposed a cost-effective MTD method in SDN which randomizes the IP addresses and generates hash-based signatures that can synchronize different MTD phases in the network. Chowdhary et al. [17] used an SDN controller to mitigate cloud network attacks through network reconfiguration.

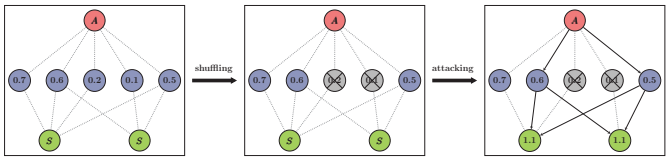
A three-tier model is proposed by Yoon et al. [18] which is used to reduce MTD cost in SDN by finding an optimal set of hosts for shuffling. A greedy backward attack path (BAP) prediction algorithm is proposed in this work to find optimal hosts to shuffle. In BAP, k most vulnerable attack paths from 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 have decided to improve the method proposed by Yoon et al. [18] (i.e. BAP) by considering the connections between the hosts and critical servers, in order to effectively mitigate DDoS attacks.

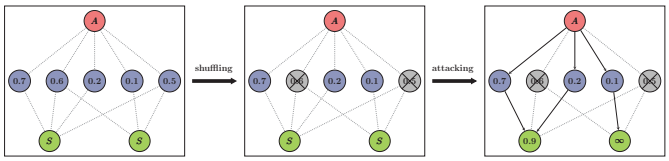
III. Proposed Method (SCEMA)

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 cost, he searches for the minimal set of hosts which can join his army and they are enough to run the attack.

The proposed method in this paper, SCEMA, mainly aims to reduce implementation cost while keeping the security level unchanged or even higher. The more complex algorithms we have, the more SCEMA costs. SCEMA attempts to find the shuffled host with low complexity, but with an efficient result.



(a) BAP solution: shuffling the hosts regarding their compromising cost



(b) SCEMA solution: shuffling the hosts regarding the number of their connections to the servers

Fig. 1. Comparing the effectiveness of SCEMA and BAP in a sample network

To reach these goals, we try to improve the BAP method [18]. In BAP, the critical paths which are more vulnerable than the others are selected and then, all the hosts in these paths are shuffled. The distributed nature of some attacks is not taken into account in this work. In some cases, the vulnerable hosts are less important than the hosts which can participate in the adversary's army to perform a distributed attack. Therefore, we can find other hosts for shuffling that are more important in DDoS attacks.

We introduce another parameter that can be measured in lower complexity and get acceptable or even better results in many cases. The adversary's willing 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 parameter which can attract the adversary's attention in many cases is the number of neighbor servers, which we call *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 compromising cost of each host. The hosts which are connected to more critical servers are the best targets for the adversary's army.

Figure 1 shows an example that compares SCEMA and BAP. The cost of compromising each host and performing DDoS attack to each critical server is shown in the nodes. In BAP, the compromising cost 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.

We have considered that the adversary's target is launching a DDoS attack on all the critical servers, which are more than one. A network under such attack can be modeled as $\mathcal{N} = (S, \mathcal{C})$, where S is the total number of critical servers and $\mathcal{C}$ is an ordered set of hosts compromising costs. We have $\mathcal{C} = \{c_1, c_2, \ldots, c_H\}$, where c_i indicates the minimum cost that the adversary must pay to compromise the i^{th} host (i.e. h_i) and H is the total number of hosts.

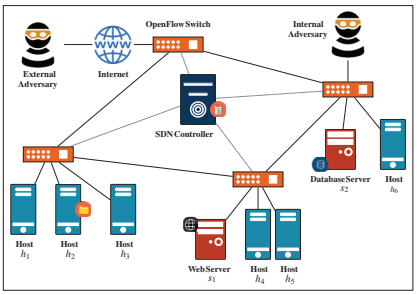


Fig. 2. The topology of $\mathcal{N}_\mathcal{E}$ with two critical servers and six typical hosts

We define a shuffling degree for each host. The shuffling degree of the i^{th} host, d_i, is the ratio of the number of servers that are directly connected to that host to the total number of servers. We believe that shuffling the hosts based on this degree has a better performance than shuffling based on the compromising cost.

A sample network, $\mathcal{N}_\mathcal{E}$, is shown in Figure 2. We have $\mathcal{C}_\mathcal{E} = \{\infty, 1.2, 0.9, 1.4, 0.9, 0.9\}$. We can figure out that the first host, h_1, does not have the vulnerabilities that lead to a successful DDoS attack against the critical servers. Or, for example, the compromising cost of h_3, h_5, and h_6 hosts are equal. The shuffling degrees of the hosts are $\{1/9, 1/9, 2/9, 2/9, 2/9, 1/9\}$. BAP suggests selecting the hosts for shuffling among the most vulnerable ones. 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 $\{h_2, h_3, h_5, h_6\}$ with the cost of 3.9. But SCEMA selects the hosts with the highest shuffling degree. So we have to shuffle $\{h_3, h_4, h_5\}$ and the hosts in this set can prevent the attack, because s_1 and s_2 have less than three unblocked connections. The costs of this set is 3.2. Hence, SCEMA finds a lower-cost set of hosts compared with BAP.

IV. System Architecture

We have designed a system in SDN that implements SCEMA. This system 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: NTD, SDC, IAS, SID, and FEG.

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 ← H + 1 to H + S + 1 do
        if c_{i,j} = 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
```

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
for each shuffling interval do
    λ ← an empty list                           ▷ A list storing hosts for shuffling
    for i ← 1 to H do
        r ← a random number between 0 and 1
        if r < d[i] then                        ▷ Adding the hosts to λ with d_i probability
            add i to λ
```

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, NTD module generates the network model, $\mathcal{N}$, and passes this model to SDC module. This module is triggered by network startup. Then the network topology is discovered and passed to SDC module.

B. Shuffling Degree Calculator (SDC)

SDC module is responsible for finding the shuffling degree of each host in the network. This module gets the network model from 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 which is 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 SID module.

C. Shuffling Interval Detector (SID)

SID finds the hosts that must be shuffled according to SCEMA. The required information is received from 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. In each shuffling interval, each host has a probability of being shuffled, which is its shuffling degree. So h_i is shuffled with a probability of d_i.

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

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 conflicting between the used addresses. When a shuffling process is performed and the hosts need another IP address, IAS module selects a random address among the addresses in its pool which its flag is not set. The random addresses are passed to FEG module, and their flag is set.

E. Flow Entry Generator (FEG)

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

V. Evaluation

We have compared SCEMA with BAP [18] and TGCESA [8] as they are comparable with SCEMA. But our main focus is on comparing SCEMA with BAP. We have presented appropriate metrics in this section to evaluate SCEMA and compare it with BAP and TGCESA. The features of the simulated networks are described, and the simulation results are also illustrated in this section.

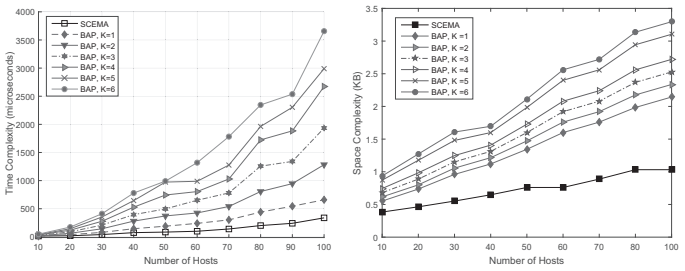
A. Evaluation Metrics

Our design goals are reducing the defense cost and retaining the network security. So we need to measure appropriate metrics to clarify high goal achievement as follows.

1) Algorithm complexity: Scalability is an essential feature that a defensive mechanism should have. Hence, the algorithms to find the important hosts and shuffling them in the network must have a low complexity and a low implementation cost to be scalable. Time complexity and space complexity can be used to measure the algorithm complexity.

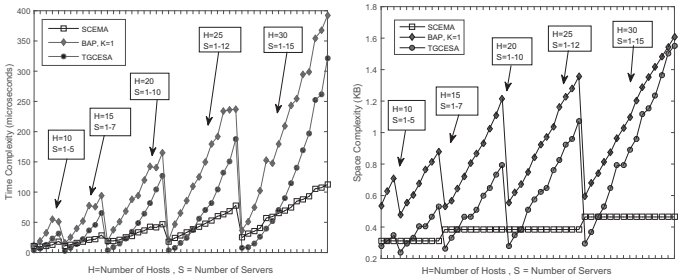
2) Adversary's success rate: The adversary's success rate is the ratio of the number of experiments in which the adversary reaches his goal to the total number of experiments. A lower rate for the adversary's success shows a better security performance in SCEMA.

3) 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. Compromised servers rate can be calculated as the ratio of the number of compromised servers to the total number of servers.



(a) Time complexity (b) Space complexity

Fig. 3. Comparing SCEMA with BAP regarding their complexity



(a) Time complexity (b) Space complexity

Fig. 4. Comparing the complexity of SCEMA, BAP, and TGCESA

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.

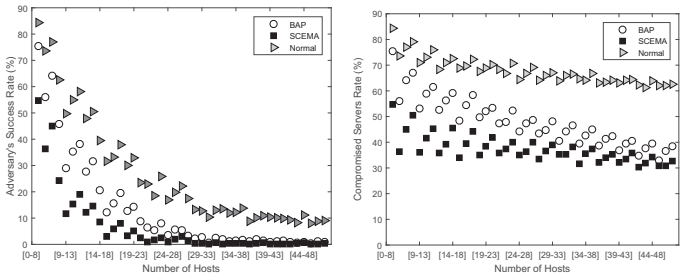
In the simulation scenarios, the simulation time is 1000 seconds, and the each shuffling interval lasts for five seconds, which means $\sigma = 5$. If third of the hosts which are connected to a critical server are compromised, the adversary can perform a successful DDoS attack against that server. The adversary randomly scans the network address space, and his/her target host is compromised in each probe with the probability of the reverse of that host's cost. When a host is compromised, it can follow the adversary's command to send a flooded traffic toward the critical servers.

We have defined multiple different network topologies, in all of which, the adversary's node is directly connected to all the host nodes. 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.

C. Simulation Results

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

1) Algorithm complexity: The time and space complexity of executing BAP and SCEMA are shown in Figure 3. 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 in different networks. Since the complexity of BAP grows as k gets higher, we have only presented the results with $k = 1$. TGCESA focuses on shuffling the servers instead of hosts. So its complexity gets higher as the number of servers grows. The time and space complexity are shown in Figure 4. 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 are 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 heavily; Because only a simple array of size $H + S$ can handle its implementation. The time complexity of SCEMA has almost a linear growth. The average result shows that the complexity of SCEMA is 52.58% lower than BAP and TGCESA.



(a) Adversary's success rate (b) Compromised servers rate

Fig. 5. Comparing SCEMA with BAP and defenseless network (i.e. Normal) regarding the adversary's success and compromised servers rates

2) Adversary's success rate: Figure 5(a) illustrates the adversary's success rate in different network topologies. 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 by considering the number of edges as the main shuffling parameter. According to the average results, SCEMA can reduce the adversary's success rate 14.32% more than BAP.

3) Compromised servers rate: The ratio of the compromised servers is shown in Figure 5(b). Again, we see that the 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 has also a lower amount in SCEMA against BAP.

VI. 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 which have more connections to the critical servers. We provide a system architecture that implements SCEMA and simulated this system in Mininet to evaluate the metrics relating to the design goals. We observe that SCEMA has lower complexity than BAP, and its complexity is independent of the attack path. Thus, it is a cost-effective solution and can easily develop for large-scale networks. The results also show that the security level is also higher than BAP.

ACKNOWLEDGMENT

This work was supported in part by the Academy of Finland Project 6Genesis Flagship (Grant No. 346208) and the EU's Horizon 2020 research and innovation programme under the INSPIRE-5Gplus project (Grant No. 871808). The paper reflects only the authors' views. The Commission is not responsible for any use that may be made of the information it contains. Mohammad Shojafar is supported by Marie Curie Global Fellowship funded by European Commission with grant agreement MSCA-IF-GF-839255.

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, P. Pinto, F. Ja'fari, and W. Zhang, "Dmaidps: a distributed multi-agent intrusion detection and prevention system for cloud iot environments," *Cluster Computing*, pp. 1–18, 2022.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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