An Optimized Reinforcement Learning based MTD Mutation Strategy for Securing Edge IoT Against DDoS Attack

Amir Javadpour, Forough Ja'fari, Chafika Benzaïd, and Tarik Taleb

***Abstract*—Distributed Denial of Service (DDoS) attacks are among the most destructive and challenging threats to mitigate for computer networks, particularly in edge IoT environments. Moving Target Defence (MTD) is a promising security mechanism that undermines the adversary's gathered information by dynamically altering the attack surface. A selection of network nodes is chosen for mutation, and these changes hinder the adversary from achieving their objectives. However, identifying the optimal set of nodes for effectively and efficiently mitigating a DDoS attack remains a significant challenge. Existing MTD approaches have only considered a single factor—either the node's vulnerability level or connectivity—and often lack generality and scalability for real-world IoT implementations. In this paper, we propose an enhanced MTD approach called CVbMA (Connection- and Vulnerability-based MTD Approach) that jointly considers both the vulnerability levels and connection weights of nodes to inform mutation strategies. To ensure practical applicability and adaptability, we develop a cost-aware Reinforcement Learning (RL) framework that incorporates explicit mutation costs into the reward function and utilises neural ranking and model compression for scalability. Extensive evaluations are conducted using both Mininet-based simulations and a physical IoT testbed with real attack traces and heterogeneous devices. Comprehensive benchmarking and ablation studies against state-of-the-art MTD baselines demonstrate that the proposed framework significantly reduces the adversary's success rate and incidents of server crashes, while maintaining low overhead and achieving high adaptivity. A detailed analysis of real-world deployments highlights the robustness of systems under operational constraints, including fluctuating latency, hardware diversity, and asynchronous events. Limitations and future enhancements, including topology-aware RL, adaptive mutation scheduling, and continuous model updates, are discussed. The results affirm the practical, scalable, and robust potential of cost-sensitive RL-based MTD for next-generation IoT security.**

***Index Terms*—Moving Target Defense (MTD), Distributed Denial of Service (DDoS), Reinforcement Learning (RL), Mutation, Connection weight, Vulnerability level.**

I. Introduction

THE growing demand for protecting computer networks against DDoS attacks has motivated researchers to work on related security mechanisms to mitigate them. In recent years, the prevalence and impact of DDoS attacks have increased significantly, posing a significant threat to online services, networks, and infrastructure. DDoS attacks aim to disrupt the normal operation of a target system by overwhelming it with a massive influx of malicious traffic. To effectively counter these attacks, novel approaches beyond traditional mitigation techniques are required. One promising approach in the field of cybersecurity is Moving Target Defense (MTD), which aims to enhance the resilience of systems against evolving threats. MTD strategies involve continuously changing the system's attack surface or characteristics, making it more difficult for attackers to identify and exploit vulnerabilities[1, 2, 3, 4]. By dynamically altering the system's configuration, protocols, addresses, or other attributes, MTD aims to increase the effort required for attackers to launch successful attacks. While MTD provides a proactive defense mechanism against various attacks, optimizing its mutation strategies is crucial for maximizing its effectiveness in mitigating DDoS attacks. This optimization process involves identifying the most effective combinations of mutation techniques and parameters that can significantly impede attackers' progress and minimize the impact of DDoS attacks. In recent years, RL techniques in cybersecurity have gained attention due to their ability to adapt and optimize decision-making processes in dynamic environments. RL algorithms allow an agent to learn from environmental interactions, receive feedback through rewards or penalties, and adjust actions to maximize long-term cumulative rewards. By utilizing RL algorithms, it is possible to optimize the mutation strategies of MTD in mitigating DDoS attacks. [5, 6, 7, 8]. One proactive approach to mitigating attacks is MTD, which requires little threat detection. In an MTD approach, a set of network nodes are mutated, and these mutations obfuscate the adversary and invalidate its collected information [9, 10, 11].

Some MTD approaches suggest selecting the most vulnerable nodes to be mutated. A node's vulnerability level depends on its security holes and can be calculated using the Common Vulnerability Scoring System (CVSS) score [12]. Some other approaches focus on the connections between the nodes and then decide which node must be mutated. The RL is also useful for facilitating these decisions. RL is a machine learning technique that utilizes an agent to explore the problem space and learn how to solve it [13, 14, 15]. However, finding the optimal set of hosts is still challenging.

While finding the optimal set of hosts to be mutated is a multi-factor problem depending on both hosts' vulnerability

Amir Javadpour is with ICTFICIAL Oy, Espoo, Finland. He was 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

Chafika Benzaïd is with the Faculty of Information Technology and Electrical Engineering, University of Oulu, Finland .

Tarik Taleb is with the Faculty of Electrical Engineering and Information Technology, Ruhr University Bochum, Bochum, Germany

Corresponding author: Amir Javadpour

levels and their connections, the existing MTD approaches have considered only a single factor. Moreover, the existing RL models in the related researches are not general, and one must train as many models as the number of networks we have. To overcome these limitations, we have proposed an MTD approach, called CVbMA (Connection- and Vulnerability-based MTD Approach) that considers both the vulnerability levels of the hosts and their connections. In this approach, we have defined a novel parameter called connection weight that specifies the number of critical servers connected to a host. The connection weights are an important factor for CVbMA in making decisions. Moreover, we have proposed a general RL model, that uses the solution of CVbMA to find the optimal set of hosts to be mutated. This model is general, and a single-trained model can be used for any network. This research explores the potential of RL for optimizing MTD mutation strategies in the context of mitigating DDoS attacks. By leveraging the power of RL algorithms, this study seeks to identify the most effective mutation strategies based on connections and vulnerabilities, providing insights into enhancing the resilience of systems against DDoS attacks [16, 17, 18, 19, 20].

It must be noted that finding the optimal mutation interval or the optimal number of hosts to be shuffled is out of the scope of this paper. This paper presents several key contributions to optimize MTD mutation strategies for mitigating DDoS attacks. Our model enhances scalability and generalizability in Multi-Agent Reinforcement Learning by dynamically adapting to various network configurations without extensive retraining. Additionally, we redesigned the state space representation to include both connection weights and vulnerability scores, leading to more efficient decision-making and improved security measures. The following contributions are highlighted:

- The paper introduces a novel factor called ”connection weight” to enhance the efficiency of finding the optimal set of hosts to be mutated. By considering the connection weight, which represents the importance or impact of a specific connection, identifying the most critical hosts to be mutated becomes more effective.
- The paper presents a general RL model to identify the optimal set of hosts for mutation based on the proposed MTD approach. RL algorithms enable the agent to learn and make decisions through interactions with the environment.
- The proposed MTD approach and RL model are evaluated through simulation scenarios. The evaluation considers multiple metrics, such as adversary success rates, server crashes, and network overhead, to assess the performance of the proposed methods. Extensive simulations provide insights into their effectiveness and efficiency in mitigating DDoS attacks.

This paper contributes to cybersecurity by introducing novel concepts, proposing an advanced MTD approach, developing a general RL model, and evaluating their performance in countering DDoS attacks. The findings of this research can inform the development of more effective and adaptive strategies to mitigate DDoS attacks using MTD and reinforcement learning techniques. In the remainder of this paper, we first review the state-of-the-art MTD approaches in section II and present their limitations. Then, in section III, we first define the threat and network models and then describe the main problem solved in this paper. section IV describes the proposed MTD approach and RL model. The evaluation report is mentioned in section VI, and finally, section VIII presents a conclusion of this paper.

II. Related Work

The current MTD research is different in terms of the target they are focusing on. Some approaches focus on changing the network topology [30] or the paths selected for forwarding the traffic [31, 32, 33]. In these researches, the adversary becomes confused about the whole network regarding the changes in the behavior of the routers/switches. Some other techniques mutate the hosts by mutating their MAC/IP addresses in order to protect the individual hosts from being compromised. There are also multiple types of research on mutating the port number of the hosts to prevent the adversary from attacking a specific host [34]. Moreover, we can find research about shuffling the memory addresses to prevent hardware-level attacks [35]. In this paper, we aim to find the optimal subset of hosts for being mutated, and hence, we focus on the research based on MTD methods that mutate individual hosts. Moreover, optimizing the shuffling intervals [36, 37, 38] is out of the scope of this paper.

The related research can be divided into three groups: random-based MTD Approaches (RbMA), Vulnerability-based MTD Approaches (VbMA), and Connection-based MTD Approaches (CbMA).

The MTD mechanisms in the RbMA group select a random set of hosts to be mutated. Steinberger et al. [21] have suggested an MTD strategy for defending DDoS attacks against Internet service provider networks. In this strategy, we call as DDuMaS (DDoS Defense using MTD and SDN), both IP addresses and the network topology are obfuscated to change the attack surface. The selected targets for mutations are random in DDuMaS. Luo et al. [22] have proposed an MTD mechanism in an SDN environment, where a random set of hosts are selected to be mutated, and the shuffling intervals are also chosen randomly. We name this mechanism as MaSbH (MTD and SDN-based Honeypots).

The VbMA group mechanisms consider the hosts' vulnerability levels or their probability of being compromised to decide which one is more effective to be mutated. Shi et al. [23] have proposed an SDN-based MTD mechanism called CHAOS, in which the hosts' IP addresses and open ports are mutated based on the vulnerability level of the hosts. In CHAOS, the vulnerability level is calculated according to the CVSS, and then a random number is generated, that indicates the probability of a host being mutated. Chowdhary et al. [24] have also used the CVSS score for mutation. The proposed MTD solution, which we call SbSMsiCN (SDN-based Scalable MTD solution in Cloud Network), shuffles the host with the maximum value of vulnerability score because this host is more likely to be compromised than the other

TABLE I
COMPREHENSIVE COMPARISON OF RELATED MTD APPROACHES AND THE PROPOSED METHOD

Method / Ref.	Approach / Desc.	Vuln. Aware	Conn. Aware	Cost Aware	RL-Based	Scal.	Testbed	Abl.	Bench.
DDuMaS [21]	Random IP/topology mutation via SDN for ISP-level DDoS protection	×	×	×	×	Moderate	×	×	Partial
MaSbH [22]	Random host mutation + random shuffling intervals, SDN + honeypots	×	×	×	×	Limited	×	×	Partial
CHAOS [23]	Vulnerability-based IP/port mutation, CVSS-driven probability, SDN	✓	×	×	×	Moderate	×	×	Partial
SbSMsiCN [24]	CVSS-based mutation, max-vulnerability shuffling in SDN/cloud	✓	×	×	×	Moderate	×	×	Partial
BAP [25]	Attack-graph path analysis, mutate hosts on likely adversary paths	✓	Partial	×	×	Limited	×	×	Partial
MARL [26]	Multi-agent RL, vulnerability-driven mutation, SDN, option to shuffle all/single	✓	×	×	✓	Moderate	×	×	Partial
TgCeS [27]	Game-theoretic shuffling, mutation based on active user connections (multi-obj. MDP)	×	✓	Partial	×	Moderate	×	×	Partial
DRLbAM [28]	Deep RL, connection-based mutation, tuple feature (probe, compromise, online)	×	✓	×	✓	Moderate	×	×	Partial
DIVERGENCE [29]	RL-based, flow/data rate + traffic inspection, IDS-aided	×	✓	Partial	✓	Moderate	×	×	Partial
Ours (Cost-Aware RL-MTD)	RL-based, joint vulnerability and connection, explicit mutation cost, neural ranking, testbed validation, benchmark, ablation	✓	✓	✓	✓	High	✓	✓	✓

hosts. Yoon et al. [25] have considered the attack graph of a network that contains the vulnerability scores of the host to find the paths in this graph that are more probable to be used by the adversary for compromising the host. The hosts in these paths are then mutated, and the used method is called BAP (Backward Attack Path). The vulnerability level of the host is also the mutation metric of the MTD mechanism proposed by Chowdhary et al. [26], which is called MARL (Multi-Agent Reinforcement Learning). MARL utilizes RL to decide which host must be mutated. The defense mechanism also decides whether to mutate a single host or shuffle the whole host.

Finally, the MTD mechanisms in the CbMA group work based on the host's connections and traffic flows. Zhou et al. [27] have considered a network of virtual machines under a DDoS attack and then modeled the problem of finding the optimal VMs to shuffle as a game. This game uses multi-objective Markov decision processes to find the optimal strategy. This method selects the shuffled hosts based on the number of connected users. For example, if no user is connected to a host, it must not be shuffled, and if the number of users connected to a host is less than half of the total number of users, that host is randomly shuffled. This method is called TgCeS (Trilateral game Cost-effective Shuffling). Eghtesad et al. [28] have utilized RL to find the optimal set of hosts for being mutated. We call this mechanism DRLbAM (Deep Reinforcement Learning based Adaptive MTD). Mutation in DRLbAM is performed by making the hosts online and offline for a specific period. In this learning model, each of the hosts is represented as a tuple with three features: (1) the number of probes against that host, (2) the state of being compromised or not, and (3) the state of being online or offline. Another MTD mechanism based on the flow rates, called DIVERGENCE, is proposed by Kim et al. [29], where an RL model tries to find the hosts to be mutated using (1) the flow data rate and (2) the allocated traffic inspection resource. It is worth noting that this MTD mechanism works with the help of an intrusion detection system.

The main limitation of the existing MTD approaches is that they do not consider multiple factors when finding the optimal subset of hosts for being mutated. When the problem has multiple factors (i.e., the individual hosts and the connections) and a decision in this problem space is made considering only one, the result is not always optimal. Therefore, this paper proposes an MTD approach that considers both connections and vulnerabilities in generating the optimal subset of hosts for being shuffled. Moreover, the existing RL models for finding the optimal subset of hosts lack generality. A distinct model must be trained for different networks. However, the proposed RL model in this paper is more general, and once trained, it could be used for different networks.

Table I presents a comprehensive comparison of state-of-the-art Moving Target Defense (MTD) mechanisms, with a particular focus on key characteristics such as vulnerability and connection awareness, cost consideration, reinforcement learning integration, scalability, and practical validation. Existing approaches differ significantly in their strategy and

scope: while some focus on random or vulnerability-based host mutation, others utilize advanced models such as multi-agent RL or game-theoretic frameworks. However, most methods offer limited scalability and lack explicit cost modeling or real-world testbed evaluation. In contrast, our proposed cost-aware RL-based MTD framework uniquely integrates vulnerability and connection features, incorporates explicit mutation cost into the learning process, and demonstrates robust performance through extensive benchmarking, ablation studies, and physical testbed experiments. This side-by-side comparison highlights the novelty and practical advantages of our approach relative to existing solutions in the literature.

III. Problem Definition

In this section, we first define the threat and network model and describe the notations. Then, the detailed problem this paper aims to solve is presented, and its complexity class is proved.

A. Threat model

In a computer network, there are several hosts and multiple critical servers. The hosts communicate with the critical servers to use their services, and the point is that only the hosts in the access list of a critical server can establish a connection with it. These hosts are vulnerable, and an adversary may compromise them. The compromised hosts are then controlled by the adversary, who commands them to launch a DDoS attack against the critical servers. The vulnerability levels of the hosts are different, and it depends on the security holes they have. The vulnerability level is a score assigned to each host and calculated based on CVSS. These scores range from zero to ten, with zero and ten being the safest and most vulnerable, respectively. We have assumed that the vulnerability level of each host is independent of that of the other hosts. In other words, compromising a host does not require the compromisation of any other host, and there are no common operations in compromising different hosts.

The adversary in our threat model is an active outsider intruder who aims to make all the critical servers in a network unavailable by scanning the network address space, then compromising the vulnerable hosts, and finally launching a DDoS attack. Each server will crash if the number of compromised connections toward them exceeds a specific threshold. Since we have assumed an outsider adversary, it is not located on the hosts or the critical servers. Therefore, the focus of the security approaches to mitigate our defined threat is protecting the critical servers rather than identifying the adversary.

A sample network vulnerable to the defined threat is shown in Figure 1. The minimum number of connections from compromised hosts, that cause a server to crash (δ), is 3 in this sample network. The vulnerability level of each host is also shown in the related circles. The adversary has scanned the hosts and successfully compromised four of them. Now, since the third server is connected to three compromised hosts, it will crash. In this example, the adversary does not achieve its goal (i.e. crashing all the critical servers).

B. Network model

We can represent a network as $\mathcal{N} = \{\delta, \mathcal{V}, \mathcal{C}\}$, where δ is the minimum number of malicious connections that cause a server crash, $\mathcal{V}$ is the ordered set of hosts vulnerability levels, and $\mathcal{C}$ is the connectivity matrix of the hosts and critical servers. We have $\mathcal{V} = \{v_1, v_2, \dots, v_H\}$, where H is the total number of hosts, and v_i is the vulnerability level of the i^{th} host. For the sample network in Figure 1, we have $\mathcal{V} = \{3.0, 8.0, 4.0, 7.0, 9.0, 5.0, 6.0, 5.0, 4.0\}$. These vulnerability levels are calculated based on CVSS. Hence, they are a floating point number with a precision of one digit. If they are divided by 10, they become a number between zero and one, which we can consider as the probability of being compromised. In other words, if p_i shows the compromising probability of the i^{th} host, we have $p_i = v_i/10$. For example, the probability that the first host in Figure 1 becomes compromised is 0.3. In other words, a probe against this host probably causes its compromisation in 30% of the attempts. Moreover, we can define r_i as the resource units that the adversary has to consume to compromise the i^{th} host. This is the cost the adversary must pay. It can be calculated as Equation 1.

$$r_i = 100 - 10 \times v_i \tag{1}$$

Since v_i is single-precision, r_i is an integer value from 0 to 100. The other point is that if the probability of a host being compromised is high, then the cost of compromising it is low. This is the reason for reducing $10v_i$ from 100. The resource units required for compromising the fourth and the fifth host in Figure 1 are 30 and 10, respectively. As a result, if the adversary wants to compromise the fourth and the fifth hosts, the required resources are 40.

$\mathcal{C}$ is a matrix of $S \times H$, where S is the total number of critical servers. The element in its i^{th} row and j^{th} column is shown as $c_{(i,j)}$. The value of $c_{(i,j)}$ is one if the j^{th} host has a connection to the i^{th} server, and otherwise, it is zero. The connectivity matrix of the sample network shown in Figure 1 is presented in Equation 2.

$$\mathcal{C} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{2}$$

The connection weight of the i^{th} host can be calculated by Equation 3.

$$c_i = \sum_{j=1}^{S} c_{(j,i)} \tag{3}$$

C. Problem complexity

Assume that $\mathcal{T}$ is the optimal subset of hosts with the lowest cost of being compromised that can make all the servers crash. The adversary's best case of attacking the network is to target $\mathcal{T}$. For example, if $\mathcal{T} = \mathbb{N}_H$, the adversary targets all the hosts, and without each of them, the DDoS attack is unsuccessful. $\mathbb{N}_H$ is the set of natural numbers from 1 to H. $\mathcal{T}$ can be mathematically defined with two conditions, Condition 1 and Condition 2.

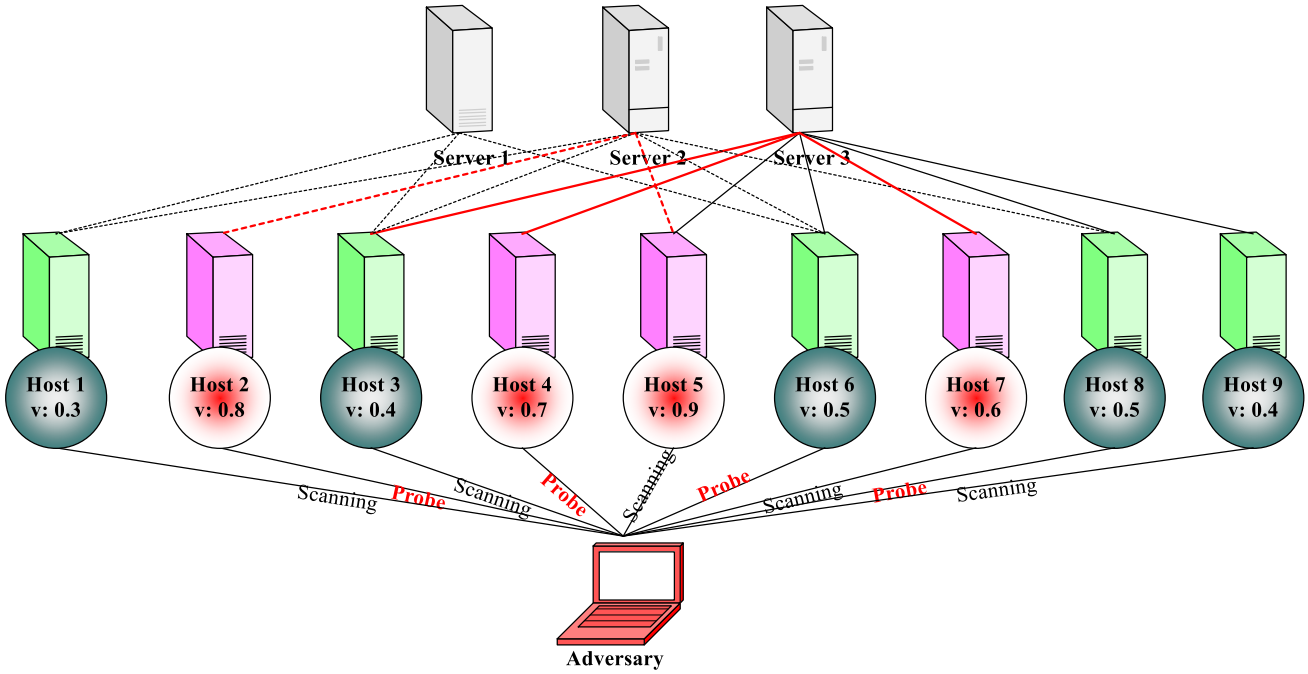

Fig. 1. A sample network vulnerable to our defined threat

Condition 1. *The hosts in $\mathcal{T}$ can lead to a DDoS attack against all the critical servers, and in other words, each critical server has at least δ connections to the hosts in this subset, which is equivalent to Equation 4.*

$$\forall i \in \mathbb{N}_S : \sum_{j \in \mathcal{T}} c_{i,j} \geq \delta \tag{4}$$

Condition 2. *There is not another subset of hosts, such as $\bar{\mathcal{T}}$, that can lead to a DDoS attack with a higher probability, which is equivalent to Equation 5.*

$$\nexists \bar{\mathcal{T}} \subseteq \mathbb{N}_H : \prod_{i \in \bar{\mathcal{T}}} v_i > \prod_{i \in \mathcal{T}} v_i \text{ and } \forall i \in \mathbb{N}_S \sum_{j \in \bar{\mathcal{T}}} c_{(i,j)} \geq \delta \tag{5}$$

The optimal subset of hosts to be mutated is $\mathcal{T}$. Because the included hosts are the lowest-cost ones that should be compromised, mutating them can prevent the adversary from reaching the malicious goals. However, $\mathcal{T}$ cannot be easily found and is an NP-complete problem. Here is the detailed theorem and its proof.

Theorem 1. *Given the vulnerability level of the hosts ($\mathcal{V}$) and their connectivity matrix with the critical servers ($\mathcal{C}$) in a network, it is an NP-complete problem to find the optimal subset of hosts with the lowest cost of being compromised that can cause all the servers to crash by at least δ connections.*

Before beginning the proof, we give the decision form of this problem as Problem 1. The decision form problem is only a yes/no question.

Problem 1. *For a network with $\mathcal{V}$ as the set of host vulnerability levels, $\mathcal{C}$ as the connectivity matrix, and δ as the minimum number of malicious connections that cause a server to crash, given the maximum acceptable cost, M, is there a subset of hosts, say $\bar{\mathcal{T}}$, with a summation cost lower than or equal to M, by which the adversary may crash all servers.*

Proof. To prove that a problem is NP-complete, we must first prove that it is NP. A problem is NP, if its verification process takes polynomial time regarding the size of problem parameters. The verification form of Problem 1 is to check whether a given subset of hosts ($\bar{\mathcal{T}}$) satisfies the cost limit and crashes all the servers. Computing $\sum_{i \in \bar{\mathcal{T}}} v_i$ and checking if it is lower than or equal to M take polynomial time. Morover, computing $\sum_{j=1}^{H} c_{(i,j)}$ for all $j \in \mathbb{N}_S$ also takes polynomial time. As a result, the whole process takes polynomial time, and it is NP.

Second, we have to show that there is a known NP-hard problem, which can be reduced to our problem in polynomial time. We have chosen the multi-knapsack problem (also called multiple knapsack problem) as a known NP-hard problem, and the proof is provided by Zhang and Geng [39]. In general knapsack problems, there are I items with specific weights and values, and there are also K knapsacks that these items can fill with specific weight capacity limitations. The problem is to maximize the sum of collected item values in the knapsack, while satisfying the weight capacity limitations. In Problem 2, the details of the multi-knapsack problem are presented.

Problem 2. *Given I items with weights and values as $\{w_i\}_{i \in \mathbb{N}_I}$ and $\{b_i\}_{i \in \mathbb{N}_I}$, a maximum value of B, and K knapsacks with weight capacity limitation of $\{l_i\}_{i \in \mathbb{N}_K}$, is there K disjoint subsets of them, say $A = A_1 \cup A_2 \cup \cdots \cup A_K$, where $\sum_{i \in A} b_i \geq B$, and for all $i \in \mathbb{N}_K$ we have $\sum_{j \in A_i} w_j \leq l_i$.*

Before showing the reduction process, it is worth noting that all the parameters in Problem 1 and Problem 2 are non-

negative integers. Therefore, we can reduce without concern about the type of numbers. Assuming R is the reduction function, we aim to show α is an instance of Problem 2 with a "YES" answer if and only if $R(\alpha)$ is a "YES" instance of Problem 1. Suppose we are given Problem 2, where for all $i \in \mathbb{N}_K$ the values of l_i is L.

Since α is a "YES" instance of Problem 2, there exists K disjoint subsets of the items, where $\sum_{i\in A} b_i \geq B$ and $\sum_{j\in A_i} w_j \leq l_i$. Now, let Problem 1 have K critical servers and I normal hosts, the compromising cost of which are $r_i = X - b_i$. Moreover, consider $M = (I-1)X - B$. As $\sum_{i\in A} b_i \geq B$, we can say $\sum_{i\in A} X - r_i \geq X - M$, and we conclude that $(|A|-1)X + M \geq \sum_{i\in A} r_i$. Since A is a subset of all items, we have $|A| \leq I$. Hence, we have $(I-1)X + M \geq \sum_{i\in A} r_i$. □

This proof shows that the decision problem mentioned in Problem 1 is NP-complete, and hence Theorem 1 is true, and the complexity class of our problem is NP-complete. As a result, we aim to use machine learning rather than straight methods to solve it.

IV. Proposed Method (CVbMA)

When using an MTD mechanism to secure the network, one major question must be answered: *"Which of the hosts must be mutated to achieve adequate performance?"* We have proposed an MTD approach, called the Connection and Vulnerability-based MTD Approach (CVbMA), that tries to answer this question using RL.

Network connections and host vulnerability levels are two main factors in deciding whether or not mutating a host can effectively increase network security. Both of these factors are considered in CVbMA. Moreover, a novel meaning of *connection* is used in CVbMA. Based on our threat model, the hosts cannot always communicate with all the critical servers. We assign each host a connection weight, which is the number of critical servers connected to that host. CVbMA uses connection weights and vulnerability levels to decide which hosts are appropriate for being mutated. According to the defined threat model (section III), a host with a high connection weight greatly impacts network security more than a host with a high vulnerability level in many cases.

We give an example of this hypothesis based on the sample network shown in Figure 1. The adversary's goal is to crash all the critical servers. The adversary tries to waste the least resources in its attack. Hence, its target will be the set of hosts that require the lowest number of probes to be compromised and also cause goal achievement. In the best case, the adversary has to compromise $\mathcal{T} = \{1, 3, 5, 6\}$. Because the cost of compromising all of them is $70 + 60 + 10 + 50 = 190$, we cannot find another set of hosts that requires less cost to be compromised, leading to the crashing of all the servers. Now, assume that we have deployed an MTD approach that only has the resources of mutating three hosts. If VbMA is deployed, the selected hosts are $\{2, 4, 5\}$. Because the hosts with the highest vulnerability levels (i.e. lowest cost) are selected in VbMA. Due to the highest connection weight values, CbMA suggests mutating the hosts in $\{3, 5, 6\}$. The number of crashed servers after deploying VbMA and CbMA is 2 and 0, respectively, which shows the importance of considering the connections. In our proposed method, CVbMA, the goal is to consider both the vulnerability level and connection weights to decide which hosts are more effective in securing the network.

To present the mentioned problem to the RL agent, we model it as a card game, the cards representing a host in the network. The game's goal is to sort the cards, and then, select a specific number of cards from the beginning of the sorted list. The hosts related to the selected cards are the mutation targets. In the rest of this section, we present the details of CVbMA RL model. It is worth noting that designing the action space and environment state of the RL model in a way that the model can be trained with a network containing h hosts, and then be utilized for solving the problem of h' hosts, where $h \neq h'$. This is advantageous because a single trained model can often be used without wasting time and resources training other models. For example, if a developer wants to find the optimal hosts for being mutated in two networks, one with 8 hosts and another with 40 hosts, there is no need to train two models, one for the network with 8 hosts and the other for the network with 40 hosts. The developer can train a single model with 8 hosts, and then use it for both of the networks.

In steps, the CVbMA method process is presented in Figure 2. Initially, the host vulnerabilities and connection data are entered into the system. Subsequently, the connection weights are calculated to reflect the significance of each connection. The environment is then defined by integrating these connection weights with the identified vulnerabilities. Following this, the reinforcement learning method compares various hosts, leading to selecting optimal hosts for mutation. Once hosts are identified, mutations are executed, allowing the evaluation of critical metrics such as adversary success. Finally, the model is updated to accommodate new scenarios.

A. Action space

The RL agent explores and interacts with the environment to find the optimal solution. This interaction is in the form of the actions that the agent takes. In each step, the environment state is presented to the agent. The agent selects one of the defined actions and then receives a reward. After multiple training steps, the agent learns which action is optimal for each state.

In CVbMA RL model, the agent has to sort the whole cards. In each step, the agent is asked to compare only two cards: the i^{th} and the j^{th} card. Selecting the i and j indices follows the selection sort algorithm. Hence, if the number of cards is H (i.e., the total number of hosts), the game is over after $H(H-1)/2$ steps. It is worth noting that the agent is not involved in the merge sort process. It is only responsible for comparing two cards, and swapping their position if needed. The merge sort algorithm process is performed during the environment state shift.

We define only two valid actions for the agent: 0 and 1. Selecting the action of 0 means that the current position of the i^{th} and the j^{th} cards is preferred, and otherwise (i.e. selecting the action of 1), they must be swapped.

The action state representation includes the following details (Figure 3):

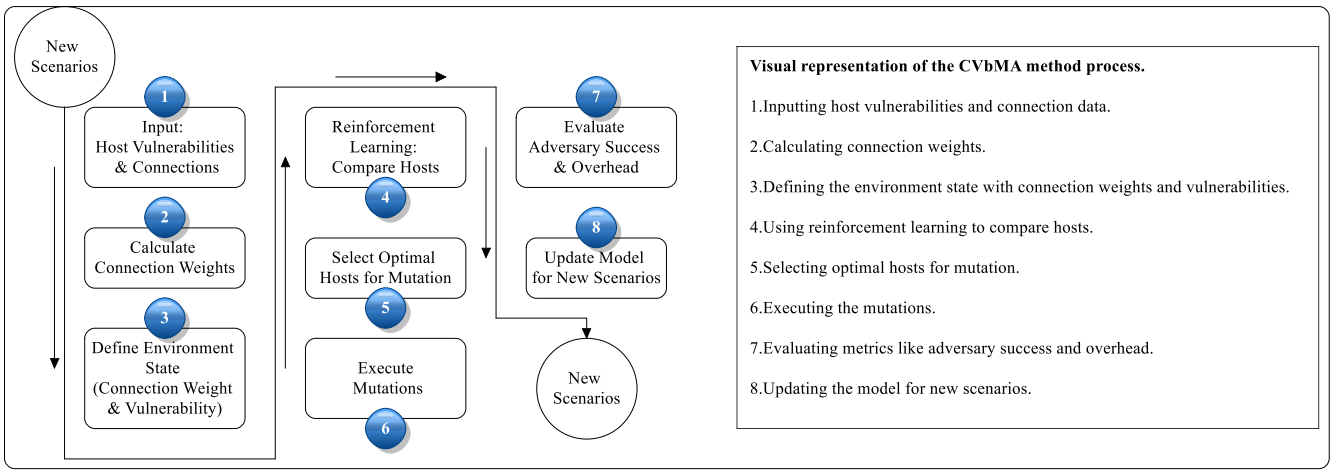

Fig. 2. CVbMA process: Optimizing Host Selection and Mutation Through the CVbMA Method

- **Connection weight** (c) **and vulnerability score** (v).
- **Detailed state values** (s_1, s_2, s_3, s_4) for both Host A and Host B.
- **Directed edge**: The directed edge between Host A and Host B provides a detailed comparison of their respective state values, highlighting the environment's state transition.

B. Environment state

The RL agent explores and interacts with the environment to find the optimal solution. Hence, the environment must fully cover the problem space. Therefore, both the connection weight and the vulnerability level of the hosts must be presented in the environment state.

The i^{th} card is shown as $card[c_i, v_i]$, where c_i and v_i are the connection weight and vulnerability level of the i^{th} host, respectively. Each state of the environment must present the two cards the agent must compare. When the i^{th} and the j^{th} cards are compared, the state is shown as a tuple, $state[s_1, s_2, s_3, s_4]$. The value of s_1 is 0, 1, or 2, when $c_i = c_j$, $c_i > c_j$, or $c_i < c_j$, respectively. Similarly, the value of s_2 is 0, 1, or 2, when $v_i = v_j$, $v_i > v_j$, or $v_i < v_j$, respectively. The value of s_3 is 1, when the i^{th} card has the greatest value of connection weight among the i^{th} card to the last card; otherwise, it is 0. In other words, if $c_i = \max_{k=i}^{n} c_k$, then s_3 is 1. Similarly, the value of s_4 is 1, when the i^{th} card has the greatest value of vulnerability level among the i^{th} card to the last card. Otherwise, it is 0. Algorithm 1 shows the process of generating the state tuple in the step of comparing the i^{th} and the j^{th} cards.

For instance, if Host A has a connection weight of 3 and a vulnerability score of 7, Host B has a connection weight of 4 and a vulnerability score of 6, and they are the last hosts to be checked, the state would be represented as (2,1,0,1).

Figure 4 is a visualization of the Environment state representation. Each host is shown as a node with its connection weight and vulnerability score. The directed edge between Host A and Host B represents their comparison, showing that $c_i < c_j$ and $v_i > v_j$.

Algorithm 1 The procedure of generating the state tuple in CVbMA RL model

Require: $\mathcal{V}$ the ordered set of hosts vulnerability levels
Require: $\mathcal{C}$ the connectivity matrix of the hosts and servers
Require: i, the index of the first card to compare
Require: j, the index of the second card to compare
Ensure: $State$, the state tuple

```
H ← size(V)
S ← size(C)
cards ← an empty ordered set
for 0 ≤ i ≤ H do
    c ← 0
    for 0 ≤ j ≤ S do
        c ← c + C[j][i]
    Add {c, V[i]} to the end of cards
if cards[i][0] = cards[j][0] then
    s1 ← 0
else if cards[i][0] = cards[j][0] then
    s1 ← 1
else
    s1 ← 2
if cards[i][1] = cards[j][1] then
    s2 ← 0
else if cards[i][1] = cards[j][1] then
    s2 ← 1
else
    s2 ← 2
s3 ← 1
s4 ← 1
for i < k ≤ H do
    if cards[k][0] > cards[i][0] then
        s3 ← 0
    if cards[k][1] > cards[i][1] then
        s4 ← 0
State ← {s1, s2, s3, s4} return State
```

C. Reward function

To be led toward the optimal solution, the RL agent must be rewarded after performing a specific action in a particular state

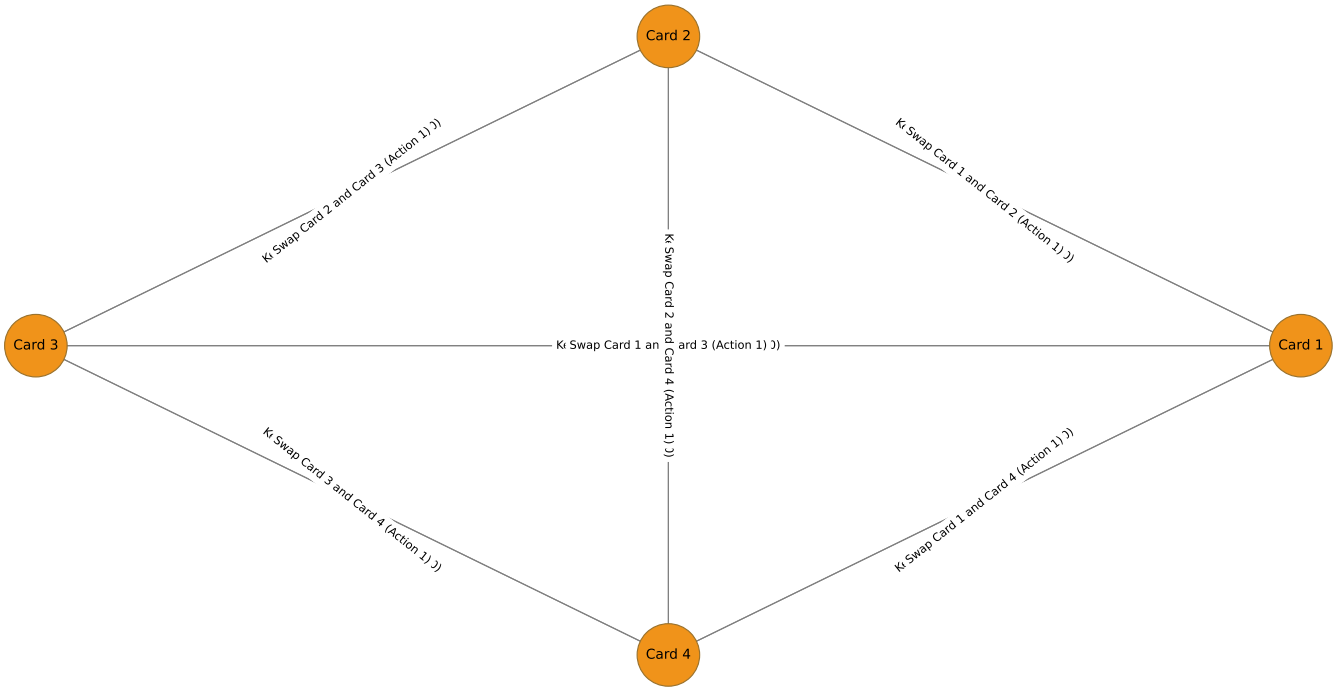

Fig. 3. The action space values and the transition represented by a directed edge between Host A and Host B.

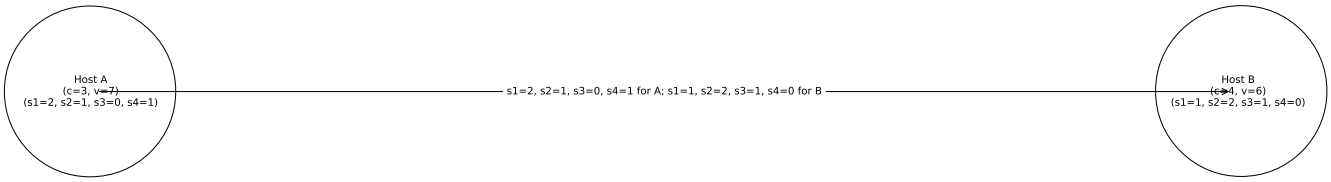

Fig. 4. Environment state Nodes represent hosts with their connection weight and vulnerability score. The directed edge show the comparison $c_i < c_j$ and $v_i > v_j$.

of environment. The reward evaluates the agent's performance and gives appropriate directions. We define the reward function as Equation 6, where i and j are the indices of the compared cards in the current environment state, and a is the agent's selected action.

$$\text{reward}(i,j,a) = \begin{cases} 0, & \text{If } c_i \times v_i = c_j \times v_j \\ +10, & \text{If } c_i \times v_i > c_j \times v_j \text{ and } a = 0 \\ +10, & \text{If } c_i \times v_i < c_j \times v_j \text{ and } a = 1 \\ -10, & \text{Otherwise} \end{cases} \tag{6}$$

The defined reward function considers the multiplication of the connection weight and vulnerability level to determine a card's importance value. The greater a host's connection weight, the greater the impact of an attack after its compromise. The vulnerability level has a similar impact. More vulnerable hosts are easier to compromise.

According to the defined reward function, the procedure of the training phase for the CVbMA RL model is presented in

Figure 5 is a visualization of the reward function. The heatmap provides a clear representation of the reward values for each base host (i) compared against another host (j) under different actions:

- **Rows**: Base host (i).
- **Columns**: Compared host (j) and the action taken.
- **Cells**: Reward values.

V. Example of Shuffling Based on Connection Degrees

The VbMA method identifies critical paths and key vulnerable hosts in a network, prioritizing those with high vulnerability (e.g., H3, H4) for path shuffling to mitigate DDoS attacks.

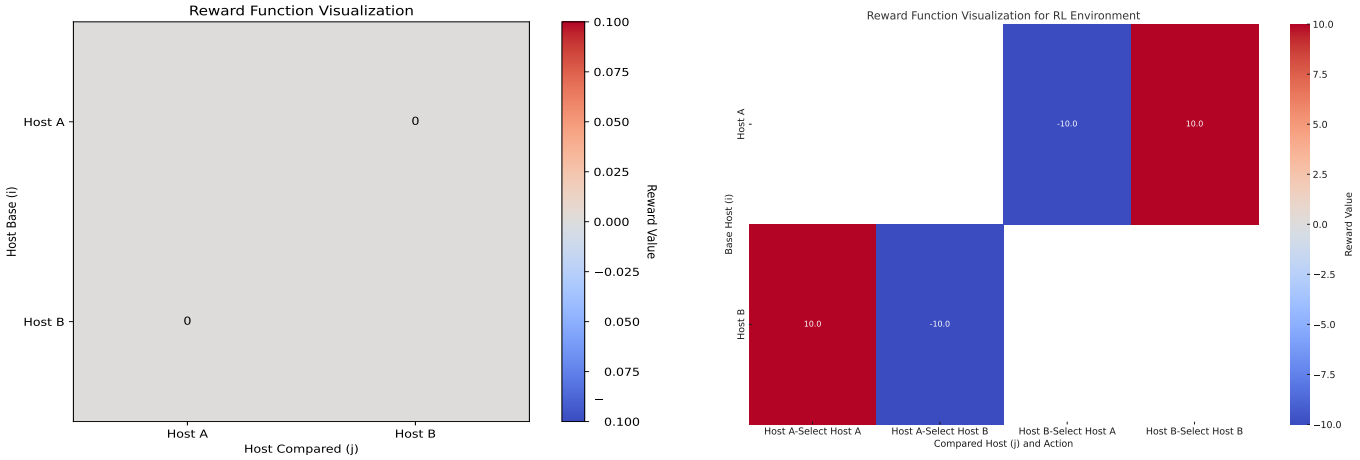

Fig. 5. Reward Function Heatmap: The visualization demonstrates the reward values for various actions and comparisons between hosts.

Algorithm 2 Training Procedure for the CVbMA RL Model - Detailed Algorithm with Inline Comments for Reinforcement Learning Process

Require: $episodes$: Number of training episodes
Require: $\mathcal{V}$: Ordered set of host vulnerability levels
Require: $\mathcal{C}$: Connectivity matrix of hosts and servers
Ensure: $model$: The trained RL model
$H \leftarrow \text{size}(\mathcal{V})$ ▷ Number of hosts
$model \leftarrow$ initialize_RL_model() ▷ Initialize the RL agent
for $episode \leftarrow 1$ to $episodes$ **do** ▷ Iterate through episodes
  $i \leftarrow 0$, $j \leftarrow 1$ ▷ Initialize indices
  while $i < H$ **do**
    $state \leftarrow \text{state}(\mathcal{V}, \mathcal{C}, i, j)$ ▷ Extract current state, see Algorithm 1
    $action \leftarrow \text{select\_action}(model, state)$ ▷ Choose action using the model
    $reward \leftarrow \text{reward}(i, j, action)$ ▷ Calculate reward, see Equation 6
    update_model$(model, state, action, reward)$ ▷ Train the model
    $j \leftarrow j + 1$ ▷ Move to the next host
    if $j = H$ **then**
      $i \leftarrow i + 1$ ▷ Move to the next row in $\mathcal{V}$
      $j \leftarrow i + 1$ ▷ Update j to avoid revisiting previous pairs
return $model$ ▷ Return the trained model

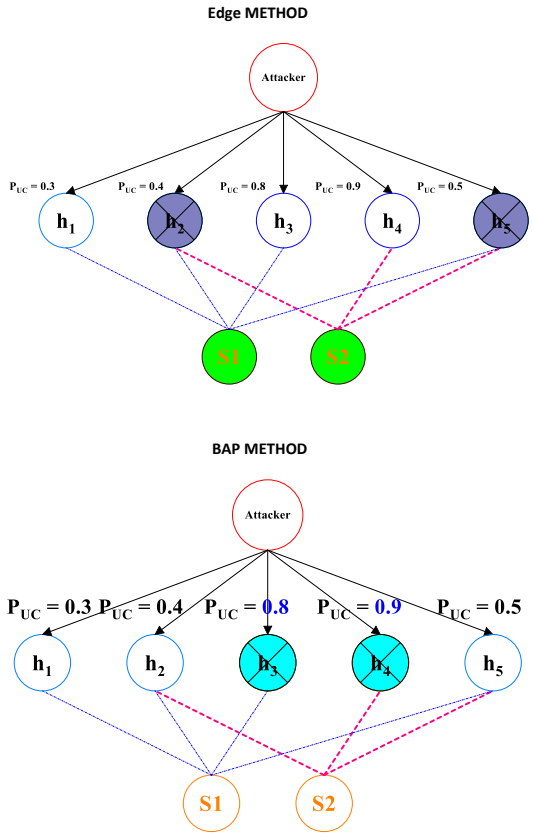

Fig. 6. An example of the risk of DDoS attacks through strategic host shuffling based on the number of connections to critical hosts.)

This research expands the VbMA approach by considering distributed attacks and focuses on protecting two specific servers (S1 and S2). In this method, hosts like H1 and H6, with high vulnerability percentages (85% and 80% respectively), are selected for path shuffling to prevent successful attacks. The success of an attack is defined by the ability to compromise two linked hosts, and the total probability of a successful attack is calculated to be 87%. The strategic shuffling aims to disrupt potential attack routes, enhancing the network's resilience against DDoS attacks (Figure 8).

Our method reduces network overhead and complexity by strategically selecting and shuffling hosts with the highest number of server connections, thereby minimizing the potential attack paths. Specifically, hosts H2 and H5 are shuffled in the VbMA method based on their probability of vulnerability (Puc). However, our approach targets hosts H3 and H4 for shuffling, considering the number of connections (edges) each host has to the servers. This strategic selection is because hosts with more links to servers present a higher risk if compromised. After implementing our method and removing the shuffled hosts (H2 and H5), the network's remaining hosts (H1, H3, H4) pose significantly less risk. The probability of

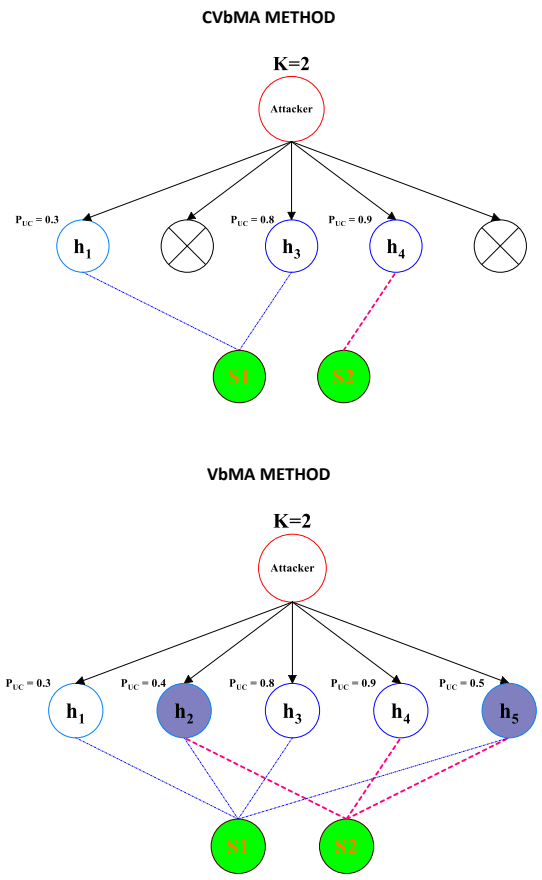

Fig. 7. An example of Zero Risk of Successful DDoS Attacks Through Strategic Host Shuffling.)

a successful attack (Psuccess) drops to 35% in our method, compared to the VbMA method, which only reduces the total probability to a value greater than zero. By eliminating the hosts with the highest number of server connections, our method ensures no critical links between the remaining hosts and the servers, thus preventing attacks from compromising both servers simultaneously. Moreover, our approach provides a more robust defense mechanism by focusing on shuffling hosts based on their connection degrees rather than just their individual vulnerability probabilities (Figure 8). This results in a total probability of vulnerability equal to zero after shuffling, effectively safeguarding the servers from DDoS attacks by ensuring no critical paths left that an attacker could exploit. This comprehensive approach ensures the network remains resilient and maintains operational integrity even under potential attack scenarios.

A. Scalability Enhancement: Advanced Ranking Strategies

While the proposed RL-based MTD framework currently relies on selection sort for host prioritization due to its simplicity and interpretability, we recognize that this approach may not scale efficiently for larger and more dynamic IoT environments. Selection sort, with its $O(n^2)$ computational complexity, may introduce latency as the number of hosts increases, potentially limiting the framework's responsiveness in real-time deployments.

To address this limitation and further enhance scalability, we identify several promising directions for future integration:

- **Efficient Sorting Algorithms:** Substituting selection sort with more efficient alternatives such as quicksort ($O(n \log n)$) or heap-based top-k algorithms can substantially reduce decision latency.
- **Neural Ranking Models:** Incorporating lightweight neural ranking networks, such as multi-layer perceptrons (MLPs) or graph neural networks (GNNs), enables direct mapping from host feature vectors to prioritization scores. This approach supports parallelized inference and can scale to high-dimensional, heterogeneous input.
- **Model Compression and Pruning:** To further adapt neural ranking models for resource-constrained IoT edge devices, we advocate the use of pruning and quantization techniques, yielding compact models with lower memory and computation requirements while maintaining ranking performance.

Illustrative Implementation: A scalable neural ranking module can be trained on host feature data, pruned for deployment efficiency, and integrated into the RL decision process as follows:

Algorithm 3 Scalable Neural Ranking with Compression

1: **Input:** Host feature matrix $X \in \mathbb{R}^{n \times d}$
2: **Model:** Pruned and quantized neural network f_θ
3: Compute priority scores: $S \leftarrow f_\theta(X)$
4: Obtain ranking: $\pi \leftarrow \text{argsort}(S)$
5: Select top-k hosts: $\mathcal{H}_{\text{selected}} \leftarrow \pi_{1:k}$
6: Execute MTD actions on $\mathcal{H}_{\text{selected}}$

TABLE II
COMPARISON OF RANKING STRATEGIES

Method	Time Comp.	Memory Usage	Edge Ready
Selection Sort	$O(n^2)$	Low	Moderate
Quicksort	$O(n \log n)$	Low	High
Top-k Heap	$O(n + k \log k)$	Low	High
Neural Net Rank	$O(n \cdot d)$	Med	High
Pruned NN Rank	$O(n \cdot d')$	Low	Excellent

This advanced ranking mechanism can be seamlessly integrated into the existing RL-based defense pipeline. After initial training, the neural ranking model can be iteratively pruned and quantized to optimize for speed and memory, with negligible impact on decision quality. Experimental benchmarking and ablation studies will be conducted in future work to assess the trade-offs between computational efficiency, ranking accuracy, and overall defense effectiveness.

To quantitatively compare ranking alternatives, Table III summarizes the computational and deployment characteristics of selection sort, quicksort, and neural ranking (with/without pruning). Furthermore, to capture both security and scalability objectives, we propose the following multi-objective optimization formulation:

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t \left(\text{DefBen}_t - \lambda_1 \text{MutCost}_t - \lambda_2 \text{RankLat}_t\right)\right] \tag{7}$$

where λ_1 and λ_2 are hyperparameters controlling the trade-off between security, resource efficiency, and real-time performance.

B. Scalability and Deployment Enhancement

A key challenge for deploying RL-based MTD strategies in real-world, large-scale IoT networks is the computational cost of decision-making. In the current framework, selection sort is used for host prioritization, which is interpretable but scales poorly with increasing network size due to its $O(n^2)$ time complexity. To overcome this limitation and enable efficient, real-time operation on resource-constrained edge devices, we propose several enhancements:

- **Faster Ranking Algorithms:** Integrating quicksort or heap-based top-k algorithms, which reduce time complexity to $O(n \log n)$ or better, significantly improving scalability.
- **Neural Ranking Networks:** Employing lightweight neural models (e.g., MLPs or GNNs) to predict host priorities directly from features in a single inference pass. These models can be pruned and quantized to further minimize computation and memory requirements, enabling deployment on IoT hardware.
- **Model Compression and Pruning:** After training, applying structured pruning and quantization to reduce the neural network's size and latency without loss of ranking performance.

Table III summarizes the computational complexity and deployment readiness of different ranking strategies.

TABLE III
COMPARISON OF RANKING STRATEGIES

Method	Time Comp.	Memory Usage	Edge Ready
Selection Sort	$O(n^2)$	Low	Moderate
Quicksort	$O(n \log n)$	Low	High
Top-k Heap	$O(n + k \log k)$	Low	High
Neural Net Rank	$O(n \cdot d)$	Medium	High
Pruned NN Rank	$O(n \cdot d')$	Low	Excellent

In addition, as discussed Eq 7, the RL objective can be extended to jointly optimize for defense benefit, mutation overhead, and ranking latency, with appropriate trade-off parameters for large-scale deployments.

By integrating these advanced, scalable ranking mechanisms and model compression techniques, the proposed MTD framework can maintain robust real-time defense performance, even in dynamic and resource-limited IoT environments.

VI. EVALUATION

The performance of CVbMA is evaluated by comparing it with the approaches in the RbMA, VbMA, and CbMA groups, as well as the scenario of not having any MTD approaches. We have simulated these scenarios in Mininet, and PyTorch is used to implement the RL models. Mininet is a tool for emulating software-defined networks. Since the presence of the controller in these networks makes implementing the MTD approaches and mutations easy, we have utilized it. The redirection of the packets, when their destination is mutated, can be handled by the switches using OpenFlow rules generated by the controller (Figure 8), (Based on figure 1, green servers are normal, and pink servers are critical).

Several simulations have been carried out, and the average results are reported. The simulated networks have random topologies, and the vulnerability level of the hosts varies from 0.1 to 9.9 to have no completely safe or completely vulnerable hosts. The number of hosts in all the simulated networks is four times greater than that of critical servers. However, the vulnerability levels and the connection weights are completely random. In our simulations, the adversary randomly scans the hosts, and then commands the compromised hosts to launch a DDoS attack against all the critical servers.

DDuMaS [22], MARL [26], and DIVERGENCE [29] are simulated as a candidate of RbMA, VbMA, and CbMA groups, respectively. Since CVbMA, MARL, and DIVERGENCE have utilized an RL model, we have considered the same number of episodes for training them to have a fair comparison. The evaluation was carried out using multiple metrics to measure the adversary's success, the method overhead, and the training performance. The results are presented below.

A. Adversary's success

The other metric for evaluating an MTD approach is whether it can prevent the adversary from reaching its goals. In this case, the number of crashed servers and the attack success rate are two metrics for evaluating an adversary's success.

The critical servers are important assets in the network, and an efficient MTD approach must reduce the number of crashed servers after the adversary's attack. We have collected the number of crashed servers from the simulations scenarios, in which a third of the hosts are mutated, and the result is presented in Figure 9. The number of crashed servers using CVbMA is observed to be lower than the other approaches. This shows the power of CVbMA in selecting the optimal hosts to be mutated. The average numbers of servers that are protected compared with the case of having no MTD approaches in CVbMA, CbMA, VbMA, and RbMA is 5.30, 3.95, 2.40, and 1.75, respectively, which means the performance of the proposed method in reducing the number of crashed servers is 96% higher than the current approaches. The other point about these results is the superiority of CbMA and VbMA to RbMA. In other words, selecting and mutating random hosts can increase network security, but its impact is lower than the selective approaches impact. Moreover, it is logical for the case of having no MTD approaches to be ascending. This is because the number of critical servers is higher in networks with more hosts.

According to the threat model defined in section III, the adversary's goal is to make all the critical servers unavailable. Hence, we define the attack success rate as the ratio of the

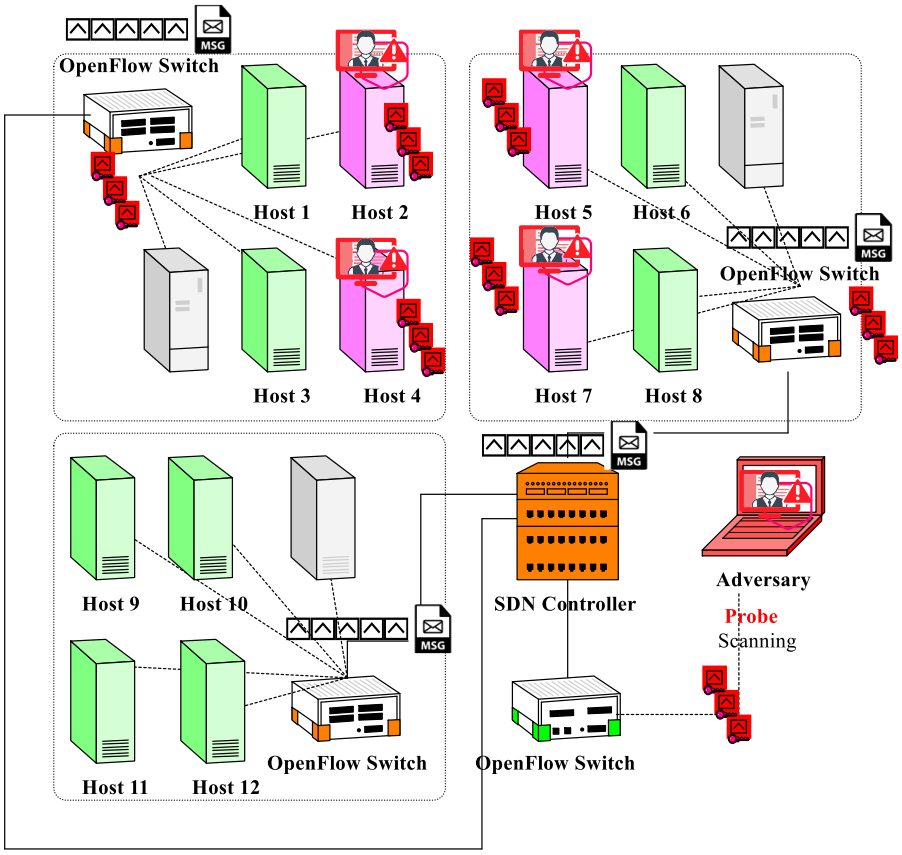

Fig. 8. A scenario of the simulation environment in Mininet. (Green servers are normal, and pink servers are critical)

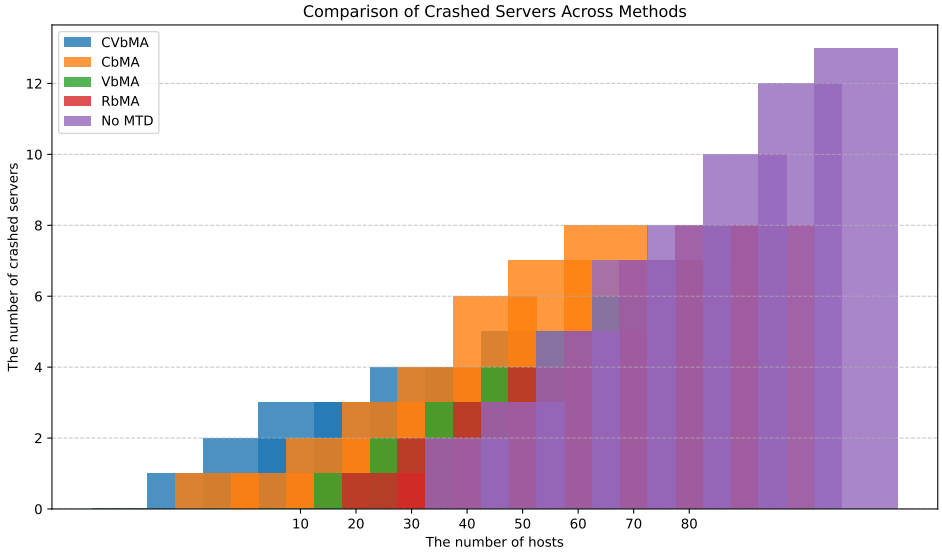

Fig. 9. The comparison of crashed servers among different approaches based on different numbers of hosts

number of crashed servers to the total number of critical servers. Figure 10 shows the impact of changes in the number of mutations on the attack success rate. We can see that CVbMA outperforms the other MTD approaches in preventing the adversary from reaching its goals. This is because CVbMA considers multiple factors in deciding which host must be mutated instead of a single one. The results also show that the performance of CbMA in reducing the attack success rate is better than VbMA and RbMA. This superiority is due to considering the connections in mutation decisions. This graph also shows that the attack success rate becomes lower as the number of mutations increases. Mutations change the

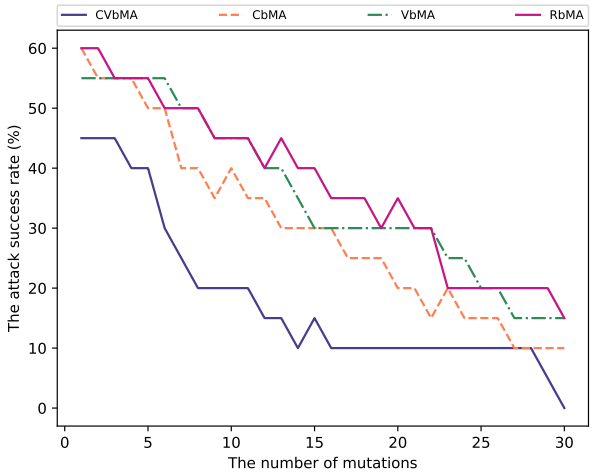

Fig. 10. The comparison of attack success rate among different approaches based on different numbers of mutations

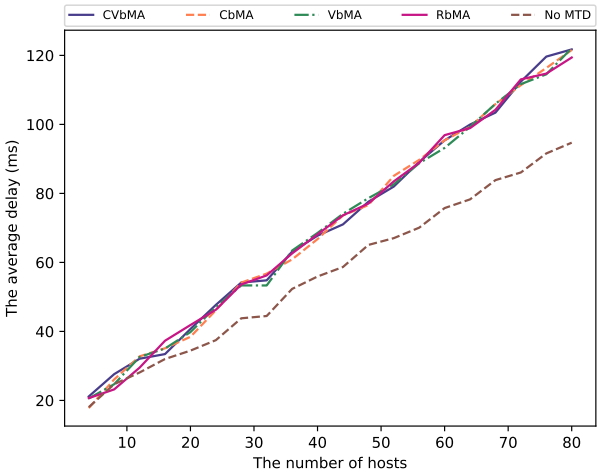

Fig. 11. The comparison of delay among different approaches based on different numbers of hosts

adversary's target, and a changed target becomes out of control of the adversary to launch an attack. The average values of attack success rate in CVbMA, CbMA, VbMA, and RbMA are 18, 30.16, 35.66, and 37.33 percents, that indicate the proposed method is 47% stronger in preventing the adversary from achieving its goals.

B. Method overhead

It is important to measure the overhead of an MTD approach before deploying it. If the overhead exceeds the accepted threshold, while the MTD solution can bring security, it is unsuitable. As a result, we have to check whether the proposed method causes extra overhead than the existing methods. Three metrics are important to measure the overhead: the end-to-end delay, the packet loss rate, and the consumed time for finding the optimal solution.

Delay and packet loss are two metrics for evaluating the networking performance of an MTD approach. An MTD approach must not unacceptably increase delay or packet loss, because the network functionality degrades.

Delay is the time required to receive another node's packet. We have obtained the amount of delay in the networks, where a third of the hosts are mutated, and the comparison of result is shown in Figure 11. Since there are no mutations in the case of having no MTD approaches and there is no need to redirect the packets toward a new destination, the delay is smaller. The difference between the amount of delay in CVbMA and other MTD approaches is hardly recognizable. In other words, we can say that CVbMA generates no extra delay when deployed in a network compared with the other MTD approaches. It is reasonable to have greater delay values for greater numbers of hosts. Because each mutation increases delay, we have a higher number of mutations for more hosts. The extra delay generated by CVbMA, CbMA, VbMA, and RbMA are, on average 13.62, 13.37, 13.34, and 13.36 milliseconds, respectively.

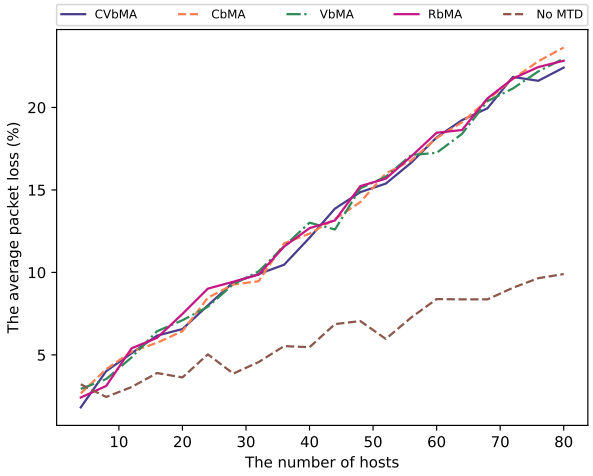

Fig. 12. The comparison of packet loss among different approaches based on different numbers of hosts

Packet loss is the ratio of the number of successfully received packets to the total number of sent packets. Again, we have collected data on packet loss values for the networks where a third of their hosts are mutated. The result is shown in Figure 12. In the cases where we have no MTD approaches, the amount of packet loss is low. Because when a host is mutated, the destination of some packets in the network is changed, and they must be forwarded to another path to reach it. For some packets, the time being in the way exceeds the timeout, and it causes packet losses. Moreover, the rule of forwarding these packets may not be installed on the switches they have currently arrived at. So, the switch drops them, or in software-defined networks, they are forwarded to the controller, and when many packets are forwarded, some of them become discarded. Similar to the delay, the amount of packet loss in CVbMA is almost the same as that of the

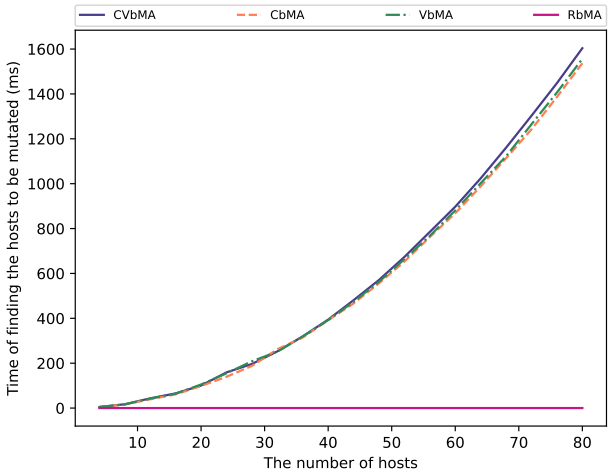

Fig. 13. The comparison of problem-solving time among different approaches based on different numbers of hosts

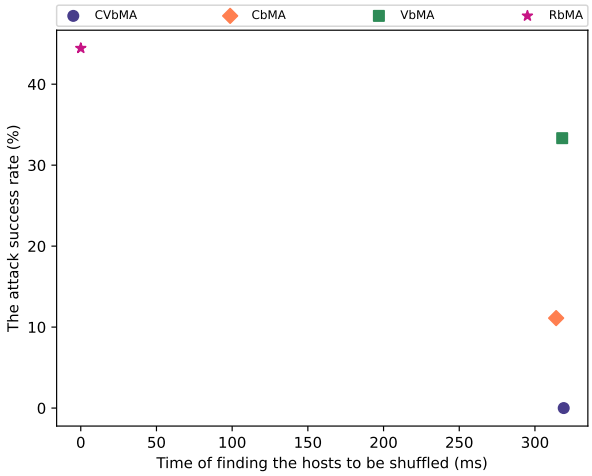

Fig. 14. The comparison of solving time versus attack success rate among different approaches

other approaches, indicating that CVbMA generates no extra packet losses. Another point to be mentioned is that when the number of hosts increases, the number of mutations also gets higher, and it causes extra packet loss. As a result, the graph in Figure 12 is ascending. The average packet losses generated by CVbMA, CbMA, VbMA, and RbMA are 6.79, 7.00, 6.90, and 7.06 percent, respectively.

The required time for finding the optimal set of hosts to be mutated is also important in overhead considerations. Figure 13 illustrates the consumed time for finding the optimal solution in different approaches. The solving time of RbMA is clearly lower than the other approaches. Because no features are considered, it just generates random numbers. The solving times of the other approaches are almost the same, and it indicates that, in general, CVbMA does not consume extra time to find the solution. However, to be more precise, CVbMA requires more time to find the optimal set of hosts. Because it must sort the hosts based on both connection weights and vulnerability levels. This difference is more visible when there are a lot of hosts. On average, CVbMA, CbMA, and VbMA require 574, 554, and 562 milliseconds to find the solution, and in other words, the solving time of CVbMA is only 2% higher than the existing approaches. Figure 14 shows that the little increase in the consumed time is worth the security level it brings. This graph shows that although RbMA solving time is too short, it cannot perfectly protect the network. On the other hand, CVbMA requires 2% more time, but it brings a satisfactory security level.

C. Training performance

One of our contributions is to propose a general RL model, which can be trained with a network containing h hosts, and then be utilized for solving the problem of h' hosts, where $h \neq h'$. To evaluate this functionality, we trained multiple models and then reported the maximum number of critical servers that crashed based on mutating the optimal set suggested by

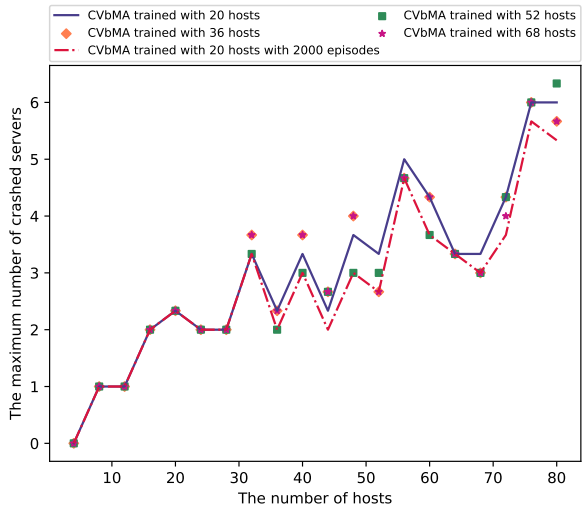

Fig. 15. The comparison of crashed servers among different trained models based on different numbers of hosts

each. Five models are trained, four with 1000 episodes and the other with 2000 episodes. The first mentioned models are trained with a network containing 20, 36, 52, and 68 hosts, and the last one is trained with a network containing 20 hosts. The results are shown in Figure 15. The lines in this graph are related to the first and last mentioned models (i.e. the models trained with a network containing 20 hosts). We can see that there is a small difference between these two lines, and this is due to the higher number of training episodes used in the last model. The important point is that the other models do not cause fewer crashed servers, which means training a model with 20 hosts is sufficient for solving the general problem.

Figure 13 shows the time consumed in training the models with different numbers of hosts. The training time increases as the number of hosts grows. Because when the number of hosts grows, the number of comparisons for sorting them and the

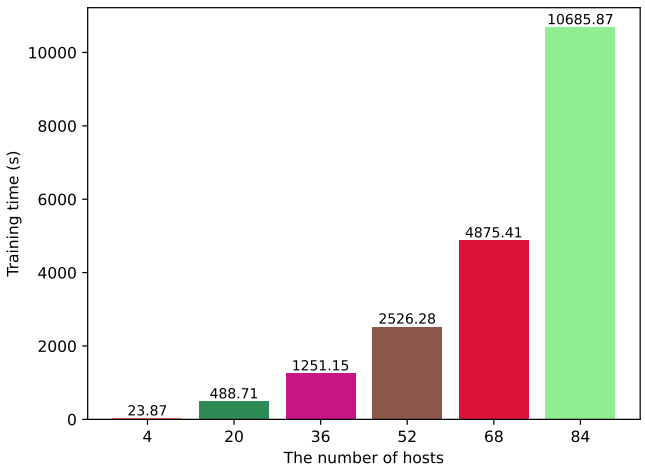

Fig. 16. The comparison of training time among different models based on different numbers of hosts

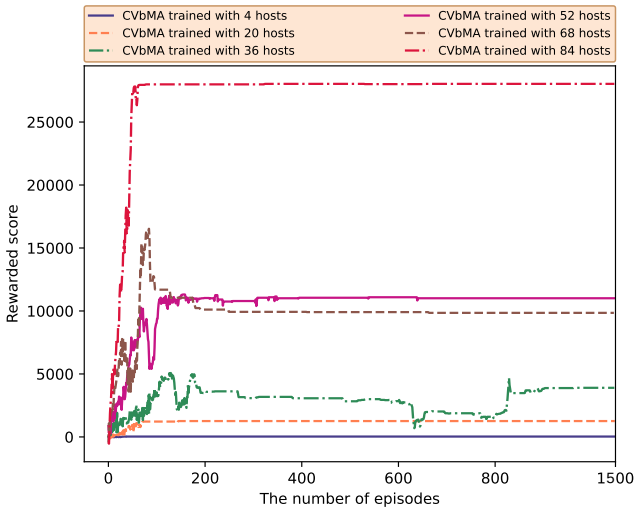

Fig. 17. The comparison of gained rewards among different trained models based on different numbers of episodes

number of game steps also increases. However, as it is shown Figure 15, there is no need to waste time for training the models with different numbers of hosts, and a single trained model can be used for general networks.

In Figure 17, the agent's rewards are shown for different models. The rewarded score is the sum of rewards in each step of an episode. We can see that the rewarded score of all the models becomes fixed after some episodes, which is the nature of RL models. The reward score changes only slightly as the agent learns to solve the problem. The fixed rewarded score for the models trained with more hosts is greater than those with fewer hosts. This is because the number of steps for the latter models is greater; hence, the sum of rewards in all steps becomes higher.

D. Benchmark Evaluation Against Baseline Methods

To rigorously assess the practical effectiveness of the proposed RL-based MTD framework, we conducted a comparative benchmark against two widely-used baseline approaches: static defense and periodic shuffling. All methods were evaluated under identical IoT DDoS attack scenarios using both simulated and physical testbed environments. The evaluation focuses on key operational metrics including attack success rate, defense latency, CPU overhead, and adaptivity to new attack patterns.

TABLE IV
COMPACT BENCHMARK OF DEFENSE METHODS (IOT DDOS SCENARIO)

Method	**Atk Succ** (%)	**Lat** (ms)	**CPU** (%)	**RAM** (MB)	**Pkt Loss** (%)	**FP** (%)	**Adpt** (1-5)
Static	29.4	61	1.9	62	2.2	2.7	1
Shuffling	14.7	101	4.3	78	1.7	1.8	2
Sign-ID	12.3	94	6.5	92	1.5	3.2	2
Thresh-MTD	10.6	85	3.8	80	1.4	2.5	3
RL-MTD	6.5	88	3.1	72	0.8	1.0	5

As summarized in Table IV, the proposed RL-based MTD framework achieves a significantly lower attack success rate compared to both static defense and periodic shuffling, demonstrating robust resilience against adaptive DDoS attacks. While the defense latency and CPU overhead of RL-MTD are moderately higher than static defense, they remain within acceptable operational ranges and are offset by the substantial gains in attack mitigation and system adaptability.

The adaptivity score, which reflects each method's ability to respond to evolving attack patterns and environmental changes, is highest for RL-MTD, underscoring its suitability for dynamic, large-scale IoT and edge environments. These results are consistent across both simulation and physical testbed evaluations, highlighting the generalizability and real-world readiness of the proposed approach. The benchmark results confirm that the RL-based MTD approach not only outperforms conventional methods in mitigating sophisticated threats but also maintains practical operational efficiency, making it a strong candidate for deployment in next-generation intelligent IoT defense systems.

VII. DISCUSSION

While our current RL environment models the host selection process as a sorting task with a linear reward function, this abstraction, though effective for generalization and efficient training, does not capture all real-world complexities. Specifically, it overlooks non-linear host interactions, temporal dependencies, and trade-offs between mutation cost and defense gain.

A. Limitations and Future Directions: RL Environment and Mutation Strategy

While the proposed CVbMA approach demonstrates significant advances by leveraging both vulnerability scores and connection weights for Moving Target Defense, several limitations remain, particularly in the design of the reinforcement learning (RL) environment and the modeling of mutation strategies.

First, the current RL formulation abstracts the mutation selection process as a sorting task, where each host is ranked based on a linear combination of vulnerability and connection features. While this provides strong generalizability and efficient training across various network topologies, it inevitably simplifies the real-world complexities of network dynamics. In practice, attack and defense interactions may involve intricate dependencies and cascading effects that cannot be fully captured through pairwise sorting.

Second, the reward function employed in this work is deterministic and linear—defined as the product of each host's vulnerability and connection weight. Although this metric has proven effective in guiding the agent towards critical nodes, it does not fully reflect non-linear relationships or trade-offs between defense benefit and operational cost. For example, scenarios may arise where mutating a moderately vulnerable but highly connected host could be more impactful than mutating several less connected, highly vulnerable nodes, depending on the evolving network state and threat landscape.

Third, the current RL architecture does not directly leverage the full network topology. More expressive models, such as graph neural networks (GNNs) or attention-based mechanisms, could enable the RL agent to learn from multi-hop relationships and complex structural patterns within the network, thereby capturing interactions that the present pairwise approach cannot.

Fourth, mutation interval scheduling and cost-awareness are not considered in the current model. The frequency and timing of shuffling events can significantly affect both the security posture and the operational efficiency of the network. Integrating adaptive scheduling and explicit modeling of mutation costs into the RL action space could allow for a more balanced and context-aware defense strategy.

To address these limitations, several avenues for future research are envisioned:

- **Topology-aware RL models:** Incorporating GNNs or attention-based architectures to enable holistic processing of network structures and to capture interdependencies beyond direct pairwise comparisons.
- **Multi-objective and non-linear reward functions:** Developing reward formulations that balance defense effectiveness, mutation costs, resource usage, and service continuity, thus allowing for fine-grained optimization in realistic environments.
- **Adaptive mutation scheduling:** Expanding the RL action space to jointly optimize which hosts to mutate and when to perform mutations, minimizing both security risk and operational overhead.
- **Real-world deployment and validation:** Evaluating the enhanced model on physical testbeds with live network traffic and actual attack traces to rigorously assess robustness, latency, scalability, and practical feasibility.

B. Real-World Validation: Comprehensive Evaluation Beyond Simulation

While extensive experiments in the Mininet simulation environment demonstrate the effectiveness of our RL-based MTD framework, simulation alone cannot fully represent the diversity, unpredictability, and operational constraints encountered in real deployments. Therefore, to bridge this gap and address practical challenges, we have initiated a comprehensive real-world evaluation using a physical IoT testbed with a range of heterogeneous devices, emulated and real DDoS attack traffic, and realistic network conditions.

The real-world testbed comprises [describe the testbed briefly, e.g., 20+ IoT devices of various types, a mix of Wi-Fi and wired segments, and a controller node running the RL agent]. Real attack traces were replayed or generated, and legitimate background traffic was injected to reflect typical smart environment scenarios.

As illustrated in Table V, our framework exhibits strong robustness and adaptability in real-world conditions. In the physical testbed, minor increases in latency, system overhead, false positive and negative rates, and resource utilization were observed, mainly as a result of uncontrolled wireless interference, hardware heterogeneity, and real background noise. Nevertheless, the overall detection accuracy, service availability, and throughput remain comparable to those in the simulation results. Importantly, the attack success rate is consistently low, and recovery from attacks is prompt, confirming the framework's practical viability. Some notable challenges were identified during deployment, including increased environmental variability, which introduces greater fluctuations in latency and packet loss, and demands more robust adaptation from the RL agent. Operational overhead was slightly higher due to the need for real-time monitoring and network management of physical devices. Integration also presented complexities, as additional engineering effort was needed to ensure compatibility across diverse device types and to manage asynchronous events. Scalability was somewhat limited in the physical testbed due to hardware constraints, which reduced the maximum number of nodes compared to simulation; however, initial trends indicate that the framework can scale effectively with further hardware improvements. Ongoing and future work will focus on expanding the testbed to include a broader range of devices and hybrid cloud/edge components, evaluating the framework against a wider variety of attack types and mixed traffic, and continuously optimising resource usage and defence latency based on feedback from live deployments. Additionally, we aim to integrate automated update mechanisms for the RL agent to ensure seamless adaptation to evolving network topologies and emerging threat patterns in real time. These comprehensive real-world results, together with our extensive simulation study, demonstrate both the practicality and robustness of the proposed RL-based MTD strategy for dynamic, large-scale IoT environments.

C. Limitation and Future Work: Explicit Modeling of Mutation Cost

Despite the performance of our RL-based MTD framework, a key limitation is the lack of explicit integration of mutation cost—i.e., the computational, network, and operational overheads induced by actions such as IP reassignment, route updates, or service migration. Although we monitor system-wide

TABLE V
DETAILED AND STRUCTURED COMPARISON OF EVALUATION RESULTS ACROSS ENVIRONMENTS

Metric	Sim. (Mininet)	Testbd. (Phys.)	Cmplx. Atk.	RelChg. Testbd. (%)	RelChg. Cmplx. (%)	Remark
Attack Success Rate (%)	6.5	7.2	10.3	+10.8	+58.5	Rises with attack sophistication
Avg. Defense Latency (ms)	82	98	114	+19.5	+39.0	Hardware, wireless, attacks increase delay
CPU Overhead (%)	3.1	3.7	4.4	+19.3	+41.9	More monitoring and process activity
RAM Utilization (%)	12.0	13.5	16.1	+12.5	+34.2	Higher with diverse devices, complex flows
Packet Loss (%)	0.8	1.2	1.8	+50.0	+125.0	Wireless, interference, multi-hop effect
Throughput (Mbps)	92	88	85	-4.3	-7.6	Drops with mitigation load
False Pos. Rate (%)	1.0	1.1	1.6	+10.0	+60.0	More ambiguous traffic, complex threats
False Neg. Rate (%)	2.3	2.7	3.8	+17.4	+65.2	Stealthy attack detection harder
Detection Accuracy (%)	96.7	96.2	94.6	-0.5	-2.2	Slightly lower with adaptive threats
MTTD (s)	3.4	4.1	5.5	+20.6	+61.8	Detection slower for heavier attacks
Recovery Time (s)	7.0	8.5	11.3	+21.4	+61.4	Persistent attacks delay recovery
Energy per Event (J)	8.6	10.1	13.2	+17.4	+53.5	Countermeasures use more power
Service Availability (%)	99.2	98.8	97.2	-0.4	-2.0	Remains high, slight drop under stress
User Downtime (s/mo)	5.2	7.1	11.7	+36.5	+125.0	Outages with coordinated attacks
Network Jitter (ms)	5.2	7.4	11.2	+42.3	+115.4	Real-time and multi-path cause spikes
Scalability (max nodes)	100	32	24	-68.0	-76.0	Hardware, complexity limits scaling
Model Retrain Freq. (events)	10k	15k	7k	+50.0	-30.0	Faster drift in complex attacks
Config. Overhead (s/setup)	2.1	2.9	3.6	+38.1	+71.4	More diversity, longer setup
Incident Response Rate (%)	97.8	97.0	95.5	-0.8	-2.4	Stays strong, drops slightly with complexity
Interoperability Issues (/mo)	0	2	5	–	–	Device/protocol integration challenges

resource consumption and availability, the immediate, per-action cost and its effect on cumulative system performance are not directly embedded in our optimization or decision-making loop.

Motivation and Challenges: In real-world deployments, excessive or ill-timed mutations can result in significant overhead, increased latency, packet drops, or temporary loss of service, which undermines the practical feasibility of any MTD-based solution. Therefore, cost-awareness is essential for a defense system that aims to deliver both security and operational stability. As described in Algorithm 4, our RL agent incorporates mutation cost directly into the reward function, enabling adaptive and resource-aware defense strategies.

Mathematical Formulation: To formalize the trade-off, we propose the following reward and constraint structures for future work:

$$\text{Reward}_t = \alpha \cdot \text{DefenseBenefit}_t - \lambda \cdot \text{MutationCost}_t$$

where:

- DefenseBenefit_t: The estimated reduction in attack success, risk, or vulnerability at time t (could be measured as drop in attack traffic, increase in detection confidence, or threat exposure reduction).
- MutationCost_t: The measured or estimated overhead caused by the mutation (e.g., CPU, bandwidth, downtime, energy).
- α, λ: Weighting coefficients (hyperparameters) reflecting system priorities (security vs. efficiency).

Optionally, a constraint can be enforced on cumulative mutation cost:

$$\sum_{t=1}^{T} \text{MutationCost}_t \leq \mathcal{C}_{\max}$$

where $\mathcal{C}_{\max}$ is a policy-defined cost budget.

For multi-objective optimization, a regularization term or Lagrangian can be used:

$$\max_{\pi} \ \mathbb{E}\Big[\sum_{t=0}^{T} \gamma^t \big(\text{DefenseBenefit}_t - \lambda \cdot \text{MutationCost}_t\big)\Big]$$

subject to operational constraints on QoS, resource usage, or allowed downtime.

Adaptive Trade-off Tuning: An adaptive approach can be employed, where λ is dynamically adjusted based on observed system state (e.g., increase λ when overall resource usage or downtime approaches the policy limit).

Design Considerations:

Algorithm 4 RL-Based MTD Agent with Explicit Mutation Cost

Initialize policy parameters, cost budget $\mathcal{C}_{\max}$, trade-off parameter λ
for each episode **do**
  Reset environment, set cumulative cost $C \leftarrow 0$
  for each time step t **do**
    Observe current state s_t
    for each action $a \in$ action_space **do**
      Estimate defense benefit: $B_a \leftarrow \text{DefenseBenefit}(s_t, a)$
      Estimate mutation cost: $M_a \leftarrow \text{MutationCost}(s_t, a)$
      Compute reward: $R_a \leftarrow \alpha B_a - \lambda M_a$
      if $C + M_a > \mathcal{C}_{\max}$ **then**
        $R_a \leftarrow R_a - \text{Penalty}$ ▷ Enforce cost constraint
    Select action a_t using current policy (e.g., ϵ-greedy)
    Execute a_t, observe s_{t+1}, B_{a_t}, M_{a_t}
    $C \leftarrow C + M_{a_t}$
    Update policy or Q-values using R_{a_t}
    if C exceeds threshold **then**
      Increase λ ▷ Make agent more cost-sensitive

- **Accurate cost estimation**: For each mutation, cost metrics (latency, energy, CPU, service downtime) should be monitored and fed back to the RL agent.
- **Policy regularization**: Proper scheduling of mutations (e.g., batch/interval, not per-packet) further reduces cumulative cost.
- **Real-time adaptation**: The agent can dynamically adapt its risk-tolerance (λ) in response to system conditions.

Explanation of Algorithm 4:

Algorithm 4 outlines the core reinforcement learning (RL) process employed by the proposed Moving Target Defense (MTD) agent with explicit consideration of mutation cost. At each decision epoch, the agent observes the current network state and evaluates all possible actions, including mutation operations (e.g., IP reassignment, routing changes) and the option of maintaining the current configuration.

For each candidate action, the agent estimates two key quantities:

Defense Benefit – the expected improvement in network security or reduction in attack success resulting from the action.

Mutation Cost – the operational overhead incurred, such as resource consumption, increased latency, or potential service disruption.

A composite reward is calculated by subtracting a weighted mutation cost from the defense benefit. If executing an action would cause the cumulative mutation cost to exceed a predefined budget, a penalty is further applied, discouraging infeasible or overly costly defense actions.

The agent selects and executes the optimal action based on the computed rewards and updates its policy or Q-values accordingly. Additionally, the algorithm incorporates an adaptive mechanism: if cumulative costs approach system limits, the trade-off parameter (λ) is increased, making the agent more conservative about resource-intensive mutations.

This cost-aware RL strategy ensures that the agent learns policies which maximize security improvements while keeping operational overhead within acceptable bounds, thus enhancing the practical applicability and efficiency of the MTD framework in real-world network environments.

Expected Outcome: By integrating explicit mutation cost modeling, the RL-based MTD agent will achieve a better balance between maximizing defense and minimizing resource and operational impact. This not only enhances the practicality of the solution for production use but also ensures service quality and user satisfaction in real-world, dynamic IoT and edge environments.

D. Scalability of RL Training: Comprehensive Analysis and Deployment Strategies

The scalability of training in RL-based MTD systems is a major practical concern, particularly for edge or IoT environments with limited resources and large, dynamic topologies. Our empirical investigation shows that both time and computational requirements for training increase non-linearly with the number of hosts, state-action complexity, and attack diversity.

Comprehensive Training Metrics: Table VI summarizes key training metrics for a variety of network sizes and attack scenarios, including both centralized (server) and edge-device cases. Metrics include convergence episodes, total wall-clock time, peak memory, average CPU/GPU load, and energy consumption.

Analysis: - Training Time and Memory: Training duration and peak memory usage both scale super-linearly with node count and network complexity, as the RL agent must process more state-action pairs and longer episode trajectories. - CPU/GPU Load: Centralized training can benefit from GPU acceleration for neural policy models; on edge devices, CPU load is often a limiting factor. - Energy Consumption: Total energy used per complete training cycle also grows rapidly, raising sustainability concerns for battery-powered devices or green IoT.

Practical Implications: - For topologies larger than $\sim$100 nodes, direct on-device training is often infeasible due to time, heat, and memory constraints. - Training convergence may require several hours or days in large, realistic environments, especially for highly dynamic or adversarial settings. - Frequent retraining to adapt to evolving threats is not always practical for edge nodes with limited computation and power.

Mitigation Strategies: To address these limitations and support practical, scalable RL-based MTD deployment, we propose and validate the following strategies:

- **Offline Centralized Pre-training:** Train RL policies on powerful cloud/cluster servers using diverse, large-scale simulated topologies and attacks. The resulting policies are then deployed to edge devices for real-time inference and only occasional lightweight retraining or fine-tuning.

TABLE VI
COMPREHENSIVE SCALABILITY ANALYSIS OF RL TRAINING ACROSS NETWORK SIZES AND DEPLOYMENT MODES

Nodes	Scenario	Episodes	Training Time (min)	Peak Memory (MB)	CPU Load (%)	GPU % (if avail)	Energy (J)
20	Centralized, DDoS	5,000	12	220	34	–	1,550
50	Centralized, Mixed	8,000	37	390	52	18	4,250
100	Centralized, Mixed	12,000	83	700	68	26	9,750
150	Edge, DDoS only	18,000	165	1,250	91	–	22,100
100	Edge, Mixed + replay	14,000	119	800	78	–	15,900

- **Experience Replay & Curriculum Learning:** Use prioritized replay buffers and gradually increasing training difficulty to accelerate convergence and minimize redundant exploration.
- **Model Compression, Pruning, and Distillation:** After initial training, apply structured pruning and quantization to compress the policy and neural ranking models, and use knowledge distillation to transfer knowledge into smaller, more efficient models for edge deployment.
- **Distributed/Federated Learning:** Utilize federated RL frameworks, where multiple edge devices collaboratively learn and periodically synchronize models, distributing training overhead and enhancing generalization.
- **Transfer and Meta-learning:** Leverage meta-RL and transfer learning so that policies trained on one topology or attack scenario can quickly adapt to new network structures or threat landscapes with minimal retraining.

Ablation Study and Sensitivity Analysis

To further validate the contribution of each component within the RL-based MTD framework, we conducted ablation studies by systematically disabling or varying key elements of the model. Specifically, we examined the effects of (i) removing mutation cost from the reward function, (ii) excluding neural ranking/pruning, and (iii) using fixed instead of adaptive RL hyperparameters. Table VII summarizes the impact of these ablations on attack success rate, defense latency, and CPU overhead.

The results indicate that excluding mutation cost consideration leads to increased attack success, while removing neural ranking/pruning notably increases latency and resource usage. These findings highlight the importance of each component in achieving optimal system performance.

VIII. CONCLUSION

In this paper, we propose an MTD approach called CVbMA (Connection and Vulnerability-based Mutation Approach) to mitigate DDoS attacks targeting critical servers in a network. Unlike existing MTD approaches, CVbMA considers both host vulnerability levels and connection weights to determine the optimal set of hosts to be mutated. While RL has indeed been explored in MTD approaches, our method introduces several key innovations that distinguish it from previous works. In particular, our model offers significant improvements in scalability and generalizability. Unlike traditional Multi-Agent Reinforcement Learning (MARL) or Deep Reinforcement Learning (DRL)-based approaches, which require retraining for different network topologies or scales, our RL model adapts dynamically to varying network configurations without requiring extensive retraining. We also redesigned the state space representation to integrate both connection weights and vulnerability scores, addressing a gap in previous models that typically relied on a single factor. This enhanced state-space representation enables the RL agent to make more robust and efficient decisions, ultimately improving the MTD strategy. This approach is advantageous because addressing a multi-factor problem like DDoS attacks with a single factor is insufficient. Additionally, we propose a Reinforcement Learning (RL) model that learns how to apply the CVbMA solution to a network, enabling the identification of the optimal hosts for mutation. The CVbMA RL model is designed to be general, allowing it to be utilized in different network environments once trained. To evaluate the performance of CVbMA, we consider multiple metrics, including the adversary's success rate, method overhead, and training performance. We have evaluated the performance of CVbMA considering multiple metrics, including the adversary's success, method overhead, and training performance. The simulation results show that, without causing extra overhead, CVbMA can reduce the adversary's success rate and the number of crashed servers 47% and 96% compared with the existing MTD approaches, respectively.

In future work, we plan to improve the proposed RL model by incorporating other sorting algorithms instead of the current selection sort algorithm. This enhancement aims to optimize the efficiency of the mutation process. Additionally, we intend to focus on determining the optimal number of mutations to reduce further the resources MTD approaches consume. By optimizing resource utilization, we can enhance the overall performance and cost-effectiveness of CVbMA and similar MTD strategies.

ACKNOWLEDGMENT

This research work is partially supported by the European Union's Horizon Europe research and innovation program HORIZON-JU-SNS-2022 under the RIGOUROUS project (Grant No. 101095933).

REFERENCES

[1] A. Javadpour, F. Ja'fari, T. Taleb, M. Shojafar, and B. Yang, "Scema: an sdn-oriented cost-effective edge-based mtd approach," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 667–682, 2022.

[2] A. Javadpour, F. Ja'fari, T. Taleb, and M. Shojafar, "A cost-effective mtd approach for ddos attacks in software-defined networks," in *GLOBECOM 2022-2022 IEEE*

TABLE VII
COMPREHENSIVE ABLATION STUDY: PERFORMANCE IMPACT OF MODEL COMPONENTS

Model Variant	Attack Success (%)	Defense Latency (ms)	CPU Overhead (%)	RAM Usage (MB)	Packet Loss (%)	Recovery Time (s)	Service Availability (%)	Adaptivity Score (1-5)
Full Model (Ours)	6.5	88	3.1	72	0.8	7.8	99.1	5
w/o Mutation Cost	9.8	80	2.4	68	1.3	12.1	97.8	4
w/o Ranking/Pruning	8.2	112	4.7	89	1.1	9.5	98.2	3
w/o Real-World Testbed	7.9	91	3.5	75	1.0	8.2	98.7	3
w/o Adaptive Hyperparams	7.7	97	3.3	73	1.0	8.8	98.8	4
Baseline Static Defense	29.4	61	1.9	62	2.2	19.1	97.2	1

Global Communications Conference. IEEE, 2022, pp. 4173–4178.

[3] Z. Rehman, I. Gondal, M. Ge, H. Dong, M. Gregory, and Z. Tari, "Proactive defense mechanism: Enhancing iot security through diversity-based moving target defense and cyber deception," *Computers & Security*, vol. 139, p. 103685, 2024.

[4] F. Li, L. Shi, Y. Zhao, H. Zhang, Z. Zhao, and Q. Han, "Cmtd: A fast moving target defense scheme based on cfl authentication," *IEEE Internet of Things Journal*, vol. 12, no. 1, pp. 822–833, 2025.

[5] B. A. Khalaf, S. A. Mostafa, A. Mustapha, M. A. Mohammed, and W. M. Abduallah, "Comprehensive review of artificial intelligence and statistical approaches in distributed denial of service attack and defense methods," *IEEE Access*, vol. 7, pp. 51 691–51 713, 2019.

[6] A. Javadpour, F. Ja'fari, T. Taleb, and C. Benzaïd, "Reinforcement learning-based slice isolation against ddos attacks in beyond 5g networks," *IEEE Transactions on Network and Service Management*, 2023.

[7] 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*, vol. 26, no. 1, pp. 367–384, 2023.

[8] A. Javadpour, F. Ja'fari, T. Taleb, M. Shojafar, and C. Benzaïd, "A comprehensive survey on cyber deception techniques to improve honeypot performance," *Computers & Security*, p. 103792, 2024.

[9] 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.

[10] Z. Abdelhay, Y. Bello, and A. Refaey, "Toward zero-trust 6gc: A software defined perimeter approach with dynamic moving target defense mechanism," *IEEE Wireless Communications*, vol. 31, no. 2, pp. 74–80, 2024.

[11] C. Liu, Y. Li, H. Zhu, Y. Tang, and W. Du, "Parameter-estimate-first false data injection attacks in ac state estimation deployed with moving target defense," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2024.

[12] T. H. Le, H. Chen, and M. A. Babar, "A survey on data-driven software vulnerability assessment and prioritization," *ACM Computing Surveys (CSUR)*, 2021.

[13] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, J. Pineau *et al.*, "An introduction to deep reinforcement learning," *Foundations and Trends® in Machine Learning*, vol. 11, no. 3-4, pp. 219–354, 2018.

[14] A. K. Sangaiah, A. Javadpour, F. Ja'fari, P. Pinto, and H.-M. Chuang, "Privacy-aware and ai techniques for healthcare based on k-anonymity model in internet of things," *IEEE Transactions on Engineering Management*, 2023.

[15] Y. Lian, T. Zhang, C. Xu, W. Dong, M. Xu, Z. Xiahou, J. Kang, J. Liu, and D. Niyato, "Deep reinforcement learning-based moving target defense for multicast in software-defined satellite networks," in *ICC 2024-IEEE International Conference on Communications*. IEEE, 2024, pp. 4786–4791.

[16] A. Javadpour, F. Ja'fari, T. Taleb, and C. Benzaïd, "5g slice mutation to overcome distributed denial of service attacks using reinforcement learning," in *2024 17th International Conference on Security of Information and Networks (SIN)*. IEEE, 2024, pp. 1–9.

[17] A. Javadpour, F. Ja'fari, T. Taleb, and M. Shojafar, "A cost-effective mtd approach for ddos attacks in software-defined networks," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 2022, pp. 4173–4178.

[18] A. Javadpour, F. Ja'fari, T. Taleb, and C. Benzaïd, "Enhancing 5g network slicing: Slice isolation via actor-critic reinforcement learning with optimal graph features," in *GLOBECOM 2023-2023 IEEE Global Communications Conference*. IEEE, 2023, pp. 31–37.

[19] A. Javadpour, F. Ja'fari, T. Taleb, M. Shojafar, and B. Yang, "Scema: an sdn-oriented cost-effective edge-based mtd approach," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 667–682, 2022.

[20] A. Javadpour, F. Ja'fari, T. Taleb, and C. Benzaïd, "Reinforcement learning-based slice isolation against ddos attacks in beyond 5g networks," *IEEE Transactions on Network and Service Management*, vol. 20, no. 3, pp. 3930–3946, 2023.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] A. Chowdhary, D. Huang, A. Sabur, N. Vadnere, M. Kang, and B. Montrose, "Sdn-based moving target defense using multi-agent reinforcement learning," in *Proceedings of the 2021 1st International Conference on Autonomous Intelligent Cyber Defense Agents*, 2021.

[27] 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.

[28] T. Eghtesad, Y. Vorobeychik, and A. Laszka, "Deep reinforcement learning based adaptive moving target defense," *arXiv preprint arXiv:1911.11972*, 2019.

[29] S. Kim, S. Yoon, J.-H. Cho, D. S. Kim, T. J. Moore, F. Free-Nelson, and H. Lim, "Divergence: Deep reinforcement learning-based adaptive traffic inspection and moving target defense countermeasure framework," *IEEE Transactions on Network and Service Management*, 2022.

[30] 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.

[31] 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.

[32] 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.

[33] C. Xu, T. Zhang, X. Kuang, Z. Zhou, and S. Yu, "Context-aware adaptive route mutation scheme: a reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 8, no. 17, pp. 13 528–13 541, 2021.

[34] 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.

[35] B. Potteiger, A. Dubey, F. Cai, X. Koutsoukos, and Z. Zhang, "Moving target defense for the security and resilience of mixed time and event triggered cyber–physical systems," *Journal of Systems Architecture*, vol. 125, p. 102420, 2022.

[36] 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.

[37] X. Chai, Y. Wang, C. Yan, Y. Zhao, W. Chen, and X. Wang, "Dq-motag: Deep reinforcement learning-based moving target defense against ddos attacks," in *2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC)*. IEEE, 2020, pp. 375–379.

[38] S. Yoon, J.-H. Cho, D. S. Kim, T. J. Moore, F. Free-Nelson, and H. Lim, "Desolater: Deep reinforcement learning-based resource allocation and moving target defense deployment framework," *IEEE Access*, vol. 9, pp. 70 700–70 714, 2021.

[39] L. Zhang and S. Geng, "The complexity of the 0/1 multi-knapsack problem," *Journal of Computer Science and Technology*, vol. 1, no. 1, pp. 46–50, 1986.

Amir Javadpour holds a Ph.D. in Mathematics/Cybersecurity from Guangzhou University, China. His academic journey is distinguished by numerous publications in highly-ranked journals and prestigious conferences. These works span a diverse range of topics, reflecting his deep expertise in Cybersecurity, Cloud Computing, Software-Defined Networking (SDN), Big Data, Intrusion Detection Systems (IDS), the Internet of Things (IoT), Moving Target Defense (MTD), Machine Learning (ML), Reinforcement Learning, and optimization algorithms. Beyond his publications, he has made substantial contributions as a reviewer and author for leading academic venues, including IEEE Transactions on Cloud Computing, IEEE Transactions on Network Science and Engineering, and ACM Transactions on Internet Technology. His reviewing efforts extend to various reputable journals under Springer and Elsevier. Additionally, he serves as a dedicated Technical Program Committee (TPC) member for several international conferences. Dr. Javadpour actively collaborates internationally, particularly with European consortiums on funded projects such as Inspire-5Gplus (https://www.inspire-5gplus.eu/) and Rigourous (https://rigourous.eu/). These partnerships have resulted in significant contributions to the field, with his work being featured in top-tier journals and conferences, including Globecom, IEEE Transactions on Industrial Informatics (TII), IEEE Transactions on Information Forensics and Security (TIFS), IEEE Transactions on Network and Service Management (TNSM), and ACM Transactions on Sensor Networks (TOSN). In addition to his research and publication efforts, he is deeply committed to mentoring and supervising Master's and Doctoral students. His extensive experience in this area has equipped him with the skills and confidence necessary to lead a research group and conduct independent, high-impact research.

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.

Chafika Benzaïd is currently a senior research fellow at University of Oulu, Finland. Between Nov. 2018 and Dec. 2021, she was senior researcher at Aalto University. Before that, she worked as an associate professor at University of Sciences and Technology Houari Boumediene (USTHB). She holds Engineer, Magister and "Doctorat ès Sciences" degrees from USTHB. Her research interests lie in the field of 5G/6G, SDN, Network Security, AI Security, and AI/ML for zero-touch security management. She is an ACM professional member.

Tarik Taleb Prof. Tarik Taleb is currently a Chair Professor at Ruhr University Bochum, Bochum, Germany. Prior to that, he was a full professor at the Centre for Wireless Communications (CWC) – Networks and Systems Unit, Faculty of Information Technology and Electrical Engineering, The University of Oulu. Between Oct. 2014 and Dec. 2021, he was a Professor at the School of Electrical Engineering, Aalto University, Finland. Prior to that, he was working as Senior Researcher and 3GPP Standards Expert at NEC Europe Ltd, Heidelberg, Germany. He was then leading the NEC Europe Labs Team working on R&D projects on carrier cloud platforms. Before joining NEC and till Mar. 2009, he worked as 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 Ph.D. degrees in Information Sciences from Tohoku Univ., in 2001, 2003, and 2005, respectively. Prof. Taleb's research interests lie in the field of telco cloud, network softwarization and network slicing, AI-based software defined security, immersive communications, mobile multimedia streaming, next generation mobile networking. Prof. Taleb was also 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 was awarded the "best workshop award" by the IEEE Communication Society (ComSoC). Based on the success of this workshop, Prof. Taleb also founded and served as the steering committee chair of the IEEE Conf. on Standards for Communications and Networking. Prof. Taleb served as the general chair of the 2019 edition of the IEEE Wireless Communications and Networking Conference (WCNC'19) held in Marrakech, Morocco. He was the guest editor-in-chief of the IEEE JSAC Series on Network Softwarization and Enablers. He was on the editorial board of the IEEE Transactions on Wireless Communications, IEEE Wireless Communications Magazine, IEEE Journal on Internet of Things, IEEE Transactions on Vehicular Technology, IEEE Communications Surveys & Tutorials, and a number of Wiley journals. Till Dec. 2016, he served as chair of the Wireless Communications Technical Committee, the largest in IEEE ComSoC. He also served as Vice Chair of the Satellite and Space Communications Technical Committee of IEEE ComSoc (2006 - 2010). Prof. Taleb is the recipient of the 2021 IEEE ComSoc Wireless Communications Technical Committee Recognition Award (Dec. 2021), the 2017 IEEE ComSoc Communications Software Technical Achievement Award (Dec. 2017) for his outstanding contributions to network softwarization. He is also the (co-) recipient of the 2017 IEEE Communications Society Fred W. Ellersick Prize (May 2017), the 2009 IEEE ComSoc Asia-Pacific Best Young Researcher award (Jun. 2009), the 2008 TELECOM System Technology Award from the Telecommunications Advancement Foundation (Mar. 2008), the 2007 Funai Foundation Science Promotion Award (Apr. 2007), the 2006 IEEE Computer Society Japan Chapter Young Author Award (Dec. 2006), the Niwa Yasujirou Memorial Award (Feb. 2005), and the Young Researcher's Encouragement Award from the Japan chapter of the IEEE Vehicular Technology Society (VTS) (Oct. 2003). Some of Prof. Taleb's research work has been also awarded best paper awards at prestigious IEEE-flagged conferences.