5G Slice Mutation to Overcome Distributed Denial of Service Attacks Using Reinforcement Learning

Amir Javadpour*, Forough Ja'fari§, Tarik Taleb¶, Chafika Benzaïd‡

*ICTFICIAL Oy, Espoo, Finland
‡Faculty of Information Technology and Electrical Engineering, University of Oulu, Oulu, Finland
§Department of Computer Engineering, Sharif University of Technology, Iran
¶Faculty of Electrical Engineering and Information Technology, Ruhr University Bochum, Bochum, Germany
*a.javadpour87@gmail.com (Corresponding Author) §azadeh.mth@gmail.com
¶tarik.taleb@rub.de ‡chafika.benzaid@oulu.fi

***Abstract*—5G slices are susceptible to indirect Distributed Denial of Service (DDoS) attacks, where overwhelming traffic directed to one slice can also disrupt other slices sharing the same infrastructure Many current mitigation methods rely on a detection phase, which may not be effective against unknown or sophisticated attacks. Moving Target Defense (MTD) is a security mechanism that invalidates the adversary's collected information, and it can be deployed without the detection phase. In this paper, we propose a Slice Mutation technique based on Reinforcement Learning (SMRL) that reduces the impact of DDoS attacks on 5G slices while keeping the number of allocated slices acceptable. SMRL proposes a general RL model that considers ternary and ranking numbers to improve learning performance. We tested SMRL on computer networks attacked by a real botnet called Mirai and assessed its performance using various measures, including a new functionality analysis method The results indicate that SMRL decreases the number of slices impacted by a DDoS attack and enhances the distribution of slices among infrastructure resources by 46% and 20%, respectively.**

***Index Terms*—5G networks, Moving Target Defense (MTD), Distributed Denial of Service (DDoS), Network security, Reinforcement Learning.**

I. Introduction

Network slicing divides a physical network into multiple virtual networks to handle user requests for custom resources. In 5G networks, each slice is logically separated from the other slices [1, 2, 3, 4]. However, they share a common substrate network, which can be the core network in 5G. This shared infrastructure makes the impact of Denial of Service (DoS) and Distributed DoS (DDoS) attacks, in which the adversary tries to send flooded traffic toward a specific target to render it unavailable, more destructive [5, 6, 7]. Consider a scenario where two slices, namely a and b, are assigned to a single physical node C, and the adversary intends to target slice a. In this case, when the adversary floods the network with traffic intended for slice a, the traffic is sent to the physical node C, which ultimately results in the degradation of the overall functionality of C. As a consequence, not only does the targeted slice a suffer, but slice b is also affected by the attack.

Most existing mitigation mechanisms for DoS/DDoS attacks rely on detecting malicious activities first. However, due to the increasing sophistication of attacker behavior and the rise of zero-day attacks, these detection methods often suffer from a high false negative rate, meaning many attacks go undetected. Consequently, relying solely on diagnostic-based solutions is not always sufficient. This highlights the importance of mitigation techniques such as Moving Target Defense (MTD) that do not depend on reconnaissance. MTD works by constantly changing the attack surface, making any information the attacker collects obsolete and reducing the chance of a successful attack.[8, 9]. In the case of network slices, it can be defined as mutating the slices to decrease the impact of DDoS attacks. To our knowledge, none of the existing research has considered slice mutation as a technique for securing 5G networks [10, 11, 12].

This paper proposes a novel MTD method based on slice mutation, utilizing Reinforcement Learning (RL) to find the optimal mutation scenario. The proposed MTD method is referred to as SMRL, which stands for Slice Mutation technique based on Reinforcement Learning. SMRL has two main goals: (1) reducing the impact of a indirect DDoS attacks and (2) keeping the number of successfully allocated slices acceptable. A general RL model that different core networks can use is proposed. This model considers the remaining CPU capacity and the number of allocated slices for each physical machine. However, these features are converted into ternary and ranking numbers to enhance the model's learning performance. The main contributions of this paper are:

- **Proposing a novel slice mutation method for protecting 5G slices against DDoS attacks and reducing their impact.**
  This paper introduces a new RL-based Slice Mutation (SMRL) technique to protect 5G network slices against distributed DDoS attacks. In traditional 5G infrastructure, the shared physical platform supporting multiple virtual slices makes the system vulnerable to indirect DDoS attacks. When one slice is targeted, the effect of the attack can spread to other slices with the same infrastructure, compromising their availability. To address this issue, the SMRL approach dynamically allocates slices in physical

resources through remapping, reducing the impact of attacks on untargeted slices. By constantly moving slices around, this method implements the MTD mechanism effectively, making it difficult for attackers to maintain their position. This approach doesn't rely on traditional detection techniques and provides an active layer of defense.

- **Proposing a general RL model that, once trained, can be utilized by different core networks.**
The RL model learns optimal mutation scenarios that balance two key goals: minimizing the impact of DDoS attacks and ensuring an acceptable level of slice allocation. What distinguishes this RL model is its generality, which allows it to be adapted and trained for different network environments without extensive modifications. The model uses a combination of triple representations for remaining CPU capacities and ranking numbers for the currently allocated slices on each physical machine to increase learning performance. This transformation simplifies the learning process and enables faster convergence and more effective decision-making. The ability of the RL model to adapt to different core networks makes it a flexible and scalable solution for mitigating DDoS risks in 5G and beyond.

- **Considering multiple metrics for evaluating the proposed method performance and introducing novel evaluation metrics.**
This paper also introduces new evaluation criteria designed to evaluate the effectiveness of the proposed SMRL method. Unlike conventional DDoS mitigation approaches that only focus on identifying or isolating attacks, SMRL is assessed on several fronts, including the proportion of affected slices, the proportion of requests accepted, and the distribution of requests across infrastructure resources. These metrics provide a more comprehensive view of system performance by measuring the impact of DDoS attacks and ensuring network efficiency and resource utilization are maintained. Simulation results performed on a network attacked by a real botnet (Mirai) show significant improvements made by SMRL.

The following sections of this paper delve into mitigating DDoS attacks against 5G slices. To begin with, a review of the existing research on the various mitigation methods is presented in section II. The threat model and network models that form the basis of the proposed solution are explained in detail in section III. The proposed MTD method and its intricate RL model are described in depth in section IV. Additionally, section V comprehensively analyzes the proposed solution's effectiveness and performance. Finally, the paper concludes with section VI, summarizing this research's key findings and contributions.

II. Related Work

This section reviews the research on securing the network slices against DoS/DDoS attacks. Thantharate et al. [13] have proposed a secure framework for 5G networks, in which the end users must first authenticate and then access the slices. This authentication process ensures that malicious users cannot access the critical slices. To ensure the security and safety of the network, we utilize advanced deep neural network learning algorithms to monitor the traffic behavior of each user. This allows for the detection of any anomalous patterns that may indicate a potential security threat. By continuously analyzing and learning from user traffic, our system can identify and respond to potential threats in real-time, keeping the network and its users safe and secure. Kuadey et al. [14] have also utilized a deep learning method to classify the network traffic toward the slices as normal and malicious. If a traffic flow is detected as a DDoS attack, it is blocked to prevent its access to the slices. Another learning model is proposed by Bousalem et al. [15], which detects the DDoS traffic and then enforces the related security policies using software-defined networking concepts. When a malicious user is detected, its traffic will be forwarded to a sinkhole-type slice to remove it from the critical slices. The DDoS attack against 5G slices is also detected by the prediction model proposed by Moudoud et al. [16]. This model collects the users' activity history and classifies them into legal, suspicious, and malicious based on the Markov stochastic process. The malicious activities are then blocked, and the suspicious ones are carefully monitored. Niu et al. [17] suggests blocking a slice that is the target of traffic load higher than a specific threshold.

The research we previously reviewed is based on detection and requires a detection phase to address DDoS attacks. These methods have limitations in dealing with zero-day (i.e., unknown) and sophisticated attacks because they rely on previously collected datasets. Our focus is on methods that depend on the detection phase.Sattar [18] have considered slice isolation to mitigate DDoS attacks. In allocating the slices on the physical network, the objective is to maximize their separation. ObSI (Optimization-based Slice Isolation) becomes ineffective in high-load conditions due to the limitations of available resources. When there are many slice requests, isolating each slice on separate physical machines becomes increasingly difficult because the system must handle more slices than its resources can fully support. As the load increases, multiple slices are forced to share the same physical infrastructure, undermining the isolation. If a DDoS attack targets one slice, the lack of isolation allows the attack to affect other slices on the same machine. Consequently, the optimization algorithm can only ensure proper isolation when there are fewer requests, but it cannot maintain this isolation under heavy load.

The mentioned limitations of detection-based and isolation-based works, which are summarized in Table I, motivate us to use more powerful techniques, such as MTD, to overcome them.

TABLE I

A SUMMARY OF THE RESEARCH TO MITIGATE DDOS ATTACKS AGAINST 5G SLICES.

Reference	Year	Main Idea	Limitation
Thantharate et al. [13]	2020	Authenticating the users and monitoring the traffic to detect anomalies	Dependency on the detection phase
Kuadey et al. [14]	2021	Classifying the traffic into normal and malicious, and blocking the malicious class	
Bousalem et al. [15]	2022	Detecting the malicious traffic and forwarding it to a sinkhole-type slice	
Moudoud et al. [16]	2020	Collecting users' activities log and blocking the malicious activities	
Niu et al. [17]	2022	Blocking the slice which is the target of high load traffic	
Sattar, Javadpour et al. [18, 19]	2019,2023	Performing slice isolation to reduce the impact of DDoS attacks	Non-scalability

III. PROBLEM DEFINITION

In a practical scenario, an adversary can gather information about which machine in the main network has the most allocated sectors through several detection techniques. A traffic analysis method involves monitoring data flow between end devices and core network machines to infer which machines carry more load and suggest more cuts. Additionally, an adversary can exploit vulnerabilities in network protocols or management interfaces, such as insecure Application Programming Interfaces (APIs), to extract resource allocation details. Social engineering or phishing attacks targeting administrators or system logs can also provide insights into the distribution of network slicing. In our specified threat model, the adversary is situated on an end device and has compromised several other end devices to form an army. Once the army is fully assembled, the adversary instructs them to send a large amount of traffic towards a machine in the core network. This attack is repeated over time. The adversary's target is the machine with the most allocated resources. This greedy behavior of the adversary is also noted in a publication by Niu et al. [17].

Now, we explain the network model to formulate the problem this paper aims to solve. We can model the network as $\mathcal{N} = \{\mathcal{P}, \mathcal{R}, \mathcal{A}\}$, where $\mathcal{P}$ and $\mathcal{R}$ are the set of physical machines and current active slice requests, and $\mathcal{A}$ is a matrix that indicates the allocation of each request on each physical machine. We have $\mathcal{P} = \{p_1, p_2, \dots, p_P\}$, where P is the total number of physical machines in the core network and p_i is the CPU capacity of the i^{th} physical machine. We also have $\mathcal{R} = \{r_1, r_2, \dots, r_R\}$, where R is the total number of active slices and r_i is the required CPU capacity of the i^{th} slice request. It is worth noting that we are not focusing on the bandwidth constraints in this paper. So, we have not considered the links features in the defined network model. $\mathcal{A}$ is a binary matrix of $P \times R$ (i.e., with P rows and R columns), where the element in the i^{th} row and the j^{th} column, $a_{(i,j)}$, is one if the j^{th} slice is located on the i^{th} physical machine. If a request is not allocated on any of the physical machines, we use the notation of ϕ in this matrix. Based on this definition, we have at most a single one in each column of $\mathcal{A}$.

A sample core network, α, with nine active slices is illustrated in Figure 1. For this network, we have $\mathcal{P} = \{7, 5, 4, 4, 5\}$ and $\mathcal{R} = \{3, 3, 2, 2, 1, 3, 2, 1, 1\}$. The $\mathcal{A}$ value is also presented in Equation 1.

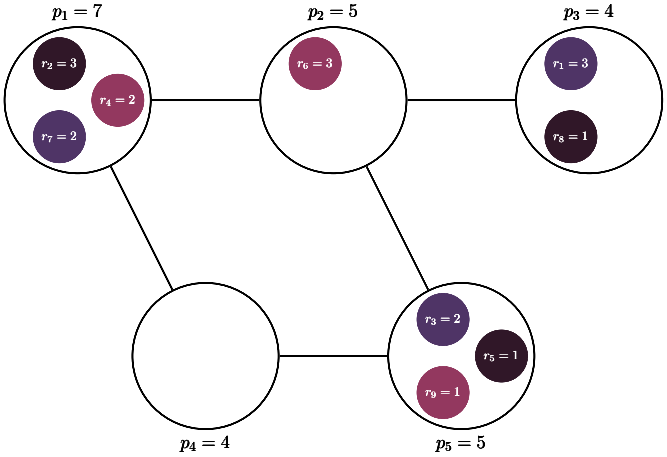

Fig. 1. A sample core network, α, with nine active slices.

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

Now, we define some functions related to this model. The $slic(\mathcal{N}, i)$ function returns the number of slices that are allocated on the i^{th} physical machine. We can calculate this function using Equation 2.

$$slice(\mathcal{N}, i) = \sum_{j=1}^{R} a_{(i,j)} \tag{2}$$

$max(\mathcal{N})$ is the highest number of slices that are allocated on a single machine. The calculation of this function is presented in Equation 3.

$$max(\mathcal{N}) = \max_{1 \leq i \leq P} slice(\mathcal{N}, i) \tag{3}$$

For calculating the consumed CPU capacity of the i^{th} physical machine, $con(\mathcal{N}, i)$ is defined. It is calculated by Equation 4.

$$con(\mathcal{N}, i) = \sum_{j=1}^{R} a_{(i,j) \times r_i} \tag{4}$$

TABLE II
SOME OF THE NUMERICAL VALUES OF THE DEFINED FUNCTION FOR FIGURE 1

function	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$
$slic(\mathcal{N}, i)$	3	1	2	0	3
$max(\mathcal{N})$			3		
$con(\mathcal{N}, i)$	7	3	4	0	4
$rem(\mathcal{N}, i)$	0	2	0	4	1
$aloc(\mathcal{N}, i, 7)$	1	1	0	1	0
$suc(\mathcal{N}, i)$	1	1	1	1	1
$sup(\mathcal{N})$			9		

$rem(\mathcal{N}, i)$ is a function that gets the physical machine index and returns its remaining CPU capacity. It can be calculated using the $con()$ function as Equation 5.

$$rem(\mathcal{N}, i) = p_i - con(\mathcal{N}, i) \tag{5}$$

Now, we define a binary function, $aloc(\mathcal{N}, i, j)$, that returns one if the j^{th} request can be allocated on the i^{th} physical machine. For calculating $aloc()$, we can use Equation 6.

$$aloc(\mathcal{N}, i, j) = \begin{cases} 1, & \text{If } a_{(i,j)} = 1 \\ 1, & \text{Else if } rem(\mathcal{N}, i) \geq r_j \\ 0, & \text{Otherwise} = 0 \end{cases} \tag{6}$$

We define another binary function, $suc(\mathcal{N}, i)$, that returns one if the i^{th} request is successfully allocated on a machine. We can use Equation 7 to calculate $suc(\mathcal{N}, i)$.

$$suc(\mathcal{N}, i) = \begin{cases} 1, & \text{If } \exists j : a_{(j,i)} = 1 \\ 0, & \text{Otherwise} = 0 \end{cases} \tag{7}$$

Finally, $sup(\mathcal{N})$ is the total number of active slices that are successfully located on a physical machine, which is calculated by Equation 8.

$$sup(\mathcal{N}) = \sum_{i=1}^{N} suc(\mathcal{N}, i) \tag{8}$$

Table II presents some of the numerical values of the defined functions for the illustrative example in Figure 1.

Based on defined notations and functions, which are summarized in Table III, the problem this paper aims to solve is *”How to dynamically minimize the value of $max(\mathcal{N})$, while keeping the value of $sup(\mathcal{N})$ at an acceptable level”*. Minimizing the value of $max()$ reduces the impact of DDoS attacks on the non-target slices, and keeping the value of $sup()$ acceptable is the primary requirement of the network slicing process.

IV. PROPOSED MTD METHOD

The proposed MTD method starts after the slices are initially allocated. This method aims to mutate the slices so that the physical machine hosting the maximum number of slices changes over time. We give an example to depict this goal according to Figure 1. In this example, four slices are allocated on the first physical machine. The adversary repeats the attack periodically. Hence, in the next launch, the first machine is again the target, and four slices become unavailable each time.

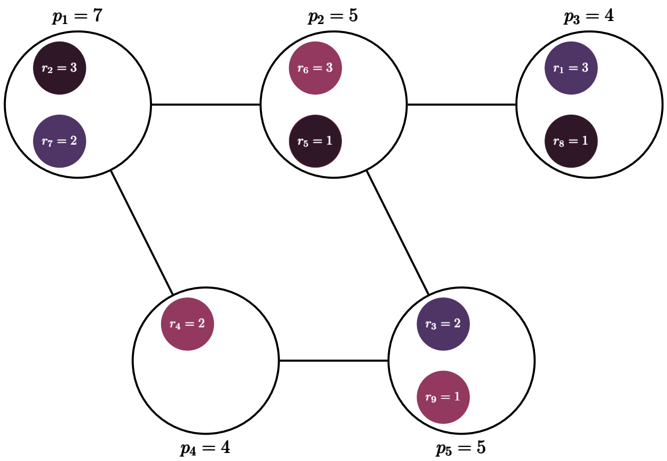

Fig. 2. The mutation of the slices in the sample core network, α

Now assume that the allocation of the slices changes over time as illustrated in Figure 2. The adversary discovers that the first machine contains the greatest number of slices. Before the adversary’s army received the attack command, the allocations had already changed, resulting in the DDoS attack causing the unavailability of only two slices.

The main challenge of the proposed method is resource limitation. The physical machines in the core network have specific CPU capacities, and the links are also bandwidth-limited [20]. Hence, not all the slice mutation scenarios are valid. The sum of the CPU capacities of the slices located on a machine must not exceed its CPU capacity. The same condition is required for the links. We are going to handle this challenge using RL models.

Our proposed MTD method, called SMRL (Slice Mutation technique based on Reinforcement Learning), suggests an RL model that extracts the features of the core network and the allocated slices on each machine and then finds the optimal mutation scenario for mitigating the DDoS attacks. In each mutation time interval, the RL model finds the best slice to be migrated to another machine. The RL model finds both the slice and the destination machines. In the remainder of this section, we explain the details of the proposed RL model.

A. Action space

Finding the optimal solution in an RL model is a kind of game for the agent. In each step of this game, one of the slices is highlighted, and the agent can decide whether to migrate it. In the case of migration, the agent also decides which physical machine is the optimal destination. There is no need to consider the actions space size as $P + 1$ since no migration occurs if the destination physical machine is the same as the current host of the slice [21]. Hence, the size of the action space is P. The game is over when the agent reaches the step of R. If the agent is in step i, where $1 \leq i \leq R$, and

TABLE III
THE NOTATIONS AND FUNCTION USED IN MODELING THE NETWORK IN THIS PAPER

Notation/Function	Description
$\mathcal{N}$	The network model, containing the core network and its active slices
$\mathcal{P}$	The set of physical machines
P	The total number of physical machines
p_i	The CPU capacity of the i^{th} physical machine
$\mathcal{R}$	The set of currently arrived slice requests
R	The total number of active slices
r_i	The required CPU capacity of the i^{th} request/slice
$\mathcal{A}$	The binary matrix of allocation states
$a_{(i,j)}$	The element in the i^{th} row and j^{th} column, which is one if the j^{th} slice is allocated on the i^{th} machine
$slic(\mathcal{N}, i)$	The number of slices allocated on the i^{th} machine (Equation 2)
$max(\mathcal{N})$	The highest number of slices that are allocated on a single machine (Equation 3)
$con(\mathcal{N}, i)$	The consumed CPU capacity of the i^{th} machine (Equation 4)
$rem(\mathcal{N}, i)$	The remained CPU capacity of the i^{th} machine (Equation 5)
$aloc(\mathcal{N}, i, j)$	The condition of the j^{th} slice about being allocated on the i^{th} machine (Equation 6)
$suc(\mathcal{N}, i)$	The condition of the j^{th} slice about being successfully allocated (Equation 7)
$sup(\mathcal{N})$	The total number of active slices that are successfully allocated (Equation 8)

selects the action of j, it means that the i^{th} slice must migrate to the j^{th} physical machine [22].

B. Environment states

The environment in an RL model represents the problem space to the agent. The agent is provided with the environment data and then explores it to find the best solution. The environment contains multiple states, and the transition between the states depends on the agent's action. In the proposed RL model, we represent each physical machine of the core network with two main features: (1) the remaining CPU capacity and (2) the number of slices that are currently mapped on it [23, 24]. It is worth noting that since we have not considered the limitations of bandwidth capacities (section III), the features related to the links are not presented in formulating the environment states.

SMRL transforms the raw features into ranking or ternary numbers to improve the agent's learning performance. The agent does not have to know the numerical values for the remaining CPU capacities. The only requirement is that the current machine can host the slice. We only need to use three numbers: two to indicate that the slice is currently located on this machine, one to indicate that the slice can be removed from the machine, and two to indicate invalid migration. For example, the sample core network, α, has $\mathcal{P} = \{7, 5, 4, 4, 5\}$. The agent must allocate the first slice (r_1) in the first step. This slice can be located on any of the machines. So we can pass $\{1, 1, 1, 1, 1\}$. If the value of r_1 was 6, we passed $\{1, 1, 0, 0, 1\}$ to the agent. These ternary values help reduce the state size; accordingly, the agent can be trained more effectively. Moreover, SMRL converts the number of slices located on a physical machine to its rank. For example, in Figure 1, the first and fifth machines have the greatest value of $slic()$. So, they are ranked as 0. The third and second machines are in the second and third places, respectively. So we can pass $\{0, 3, 2, 4, 0\}$ to the agent. Using this transformation makes the proposed RL model general. In other words, once a single model is trained, any core networks with P physical machines can utilize it. This model is dependent on neither the values of CPU capacities nor the slice features.

Algorithm 1 The procedure of generating the environment state

```
Require: $\mathcal{N}$, the network model and its parameters
Require: $request$, the index of the current slice request
Ensure: $state$, the current environment state
  $state \leftarrow \{\}$
  for $1 \le i \le P$ do
      if $a_{(i,request)} = 1$ then
          Add 2 to $states$
      else if $rem(\mathcal{N}, p_i) \ge r_{request}$ then          ▷ Equation 5
          Add 1 to $state$
      else
          Add 0 to $state$
  $ranks \leftarrow \{\}$
  for $1 \le i \le P$ do
      $max \leftarrow \phi$
      for $1 \le j \le P$ do
          if $j \in ranks$ then
              continue
          $ms \leftarrow slic(\mathcal{N}, max)$
          if $max = \phi$ or $slic(\mathcal{N}, j) > ms$ then          ▷ Equation 2
              $max \leftarrow j$
      Add $max$ to $ranks$
  for $1 \le i \le P$ do
      for $1 \le j \le P$ do
          if $ranks[j] = i$ then
              $ls \leftarrow slic(\mathcal{N}, ranks[j-1])$          ▷ Equation 2
              if $j > 1$ and $ls = slic(\mathcal{N}, ranks[j-1])$ then
                  Add the last member of $state$ to $state$
              else
                  Add $j$ to $state$
              break
  return $state$
```

The procedure of generating the environment states is presented in Algorithm 1. In the first loop of this algorithm, the ternary numbers are generated. The second loop sorts the physical machines based on their allocated slices, and the last loop calculates the rank of each machine.

Algorithm 2 The procedure of training the agent in SMRL

```
Require: $\mathcal{N}$, the network model as the environment
Require: $episodes$, the number of training episodes
Ensure: $model$, the trained model
  $model \leftarrow$ initialize the RL model
  for $1 \leq e \leq episodes$ do
      $moves \leftarrow 0$
      $mutation \leftarrow$ FALSE
      while $moves < R$ do
          $state \leftarrow$ generate the environment state      ▷ Algorithm 1
          $action \leftarrow$ the optimal action found by $model$
          $r \leftarrow moves$
          $p \leftarrow action$
          if $state[p] = 0$ then
              $reward \leftarrow$ -30
          else if $state[p]$ = 1 then
              if $state[P + p] = 0$ then
                  $reward \leftarrow$ -10
              else
                  $reward \leftarrow 0$
                  $mutation \leftarrow$ TRUE
          else
              $reward \leftarrow 0$
          if $moves = R - 1$ then
              if $mutation$ then
                  $reward \leftarrow reward$ + 1
          Allocate the $r^{th}$ request on the $p^{th}$ physical machine
          Update $model$ based on $state$, $action$, and $reward$
          $moves \leftarrow moves$ + 1
  return $model$
```

C. Reward function

The agent is led toward the optimal solution by the awards received after each action in each state. There are different conditions for the actions as follows.

- Migrating a slice to a machine with insufficient capacities is unsuitable. Hence, if the action leads to this situation, the value of the reward is a big negative number (i.e., -30).
- Since we are performing a mutation scenario, having at least one migration in each time interval is suitable. So, we give a big positive number for cases with at least one mutation.
- It is not preferred to migrate a slice to a machine with the highest number of allocated slices (i.e., the highest value of $slic()$). As a result, we give a reward equal to a small negative number for the actions resulting in this situation.

Algorithm 2 explains the procedure of training the agent in detail.

V. Evaluation

To evaluate the performance of SMRL, we simulated thousands of networks, and the average results are reported in this section. We used PyTorch to implement the RL model and Python to simulate the networks. The performance of SMRL is compared with that of ObSI [18, 22, 25, 26], the only comparable method, which is a mitigation method for DDoS attacks against 5G slices with no requirement of the detection phase. The core networks in the simulation scenarios have different numbers of physical machines, varying from 5 to 25. The CPU capacity of these machines is randomly selected, with a maximum limit of 40. The slice requests also require a random amount of resources with a maximum of 6 CPUs. The number of active slices at each time also varies between 2 and 100. Since there are 20 different core networks, we have trained the proposed RL model with 20 different conditions, all of them with 5000 training episodes.

There are 100 end devices in the simulated networks, and they are connected to the core network through a single switch. We have implemented the botnet of Mirai to compromise the end devices, and the launched DDoS attack follows the defined threat model (section III). The main adversary is located on a random end device, and five random end devices are initially compromised as the first bots. Then, these bots scan the network and probe the other victim's devices. Once the authentication credential of an end device is found, it is reported to Mirai's loader component. The loader loads a malicious script onto the victim's device and turns it into a bot. Once the botnet army is established, the adversary commands them to launch a DDoS attack against the physical machine with the highest number of slices. The DDoS command is sent randomly every 60-120 seconds. In our comparison with ObSI, we evaluated the performance of SMRL based on its effectiveness in mitigating DDoS attacks without relying on detection mechanisms. While ObSI considers additional constraints such as bandwidth and end-to-end delay, our focus was specifically on assessing the resilience of slice mutation in enhancing network security against these attacks. We have not implemented a variant of ObSI that incorporates these constraints; instead, we analyzed SMRL in its current form. Furthermore, it is essential to note that ObSI assumes slices consist of multiple Virtual Network Functions (VNFs) and discusses both intra- and inter-slice isolation. In contrast, SMRL focuses primarily on slices composed of a single VNF, leading us to compare them against the inter-slice scenarios presented in ObSI mainly. We acknowledge the differences in assumptions and constraints between the two methods, which are crucial for interpreting the comparison results.

We have considered three evaluation metrics for assessing SMRL's performance: the affected slices ratio, the request acceptance ratio, and the distribution of the allocated requests. The remainder of this section presents the analysis of these metrics.

A. Affected slices ratio

The affected slice ratio is the number of slices that become unavailable after a DDoS attack compared to the total number of successfully allocated slices. One of the SMRL's goals is to minimize the value of $max(\mathcal{N})$, and the affected slices ratio is an excellent metric to evaluate it. Figure 3 reports the simulation results regarding the affected slices ratio. The first point about this graph is its descending order. The growth in active slices results in fewer successfully allocated slices. So, the core network resources are assigned to fewer slices;

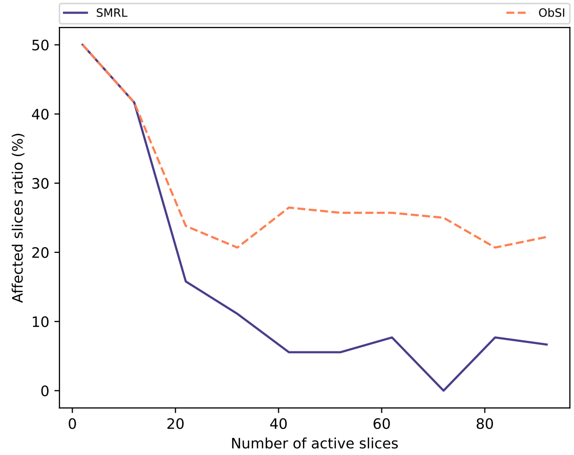

Fig. 3. Comparing the affected slices ratio of SMRL and ObSI methods.

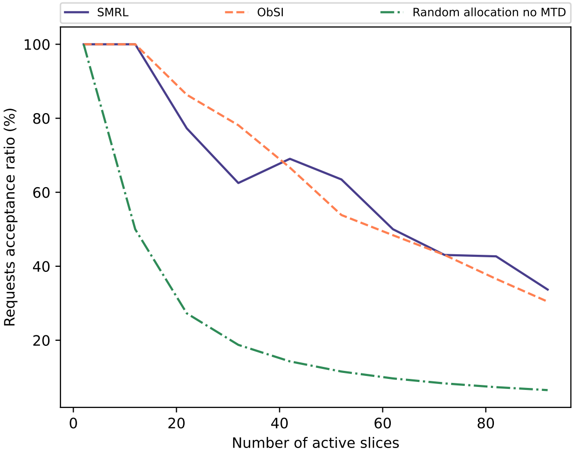

Fig. 4. Comparing the requests acceptance ratio of SMRL and ObSI methods and having random allocations without mutations.

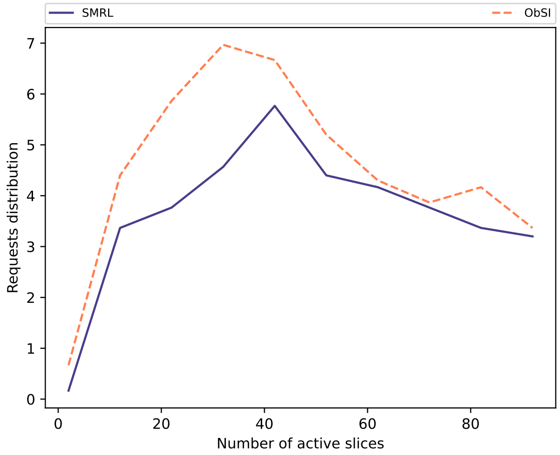

Fig. 5. Comparing the requests distribution of SMRL and ObSI methods.

hence, the number of slices allocated on a single machine becomes smaller. The other point is the significant difference in the ratio of affected slices between SMRL and ObSI. On average, SMRL has improved this metric by 46%. This is due to changes in the slice allocations. The fixed allocation in ObSI can only achieve a limited isolation level. However, the use of SMRL results in modified allocations over time, leading to different $max()$ values.

B. Requests acceptance ratio

The second goal of SMRL is to retain an acceptable number of successfully allocated slices (i.e., $suc(\mathcal{N})$). This goal can be evaluated using the requests acceptance ratio, the ratio of successfully allocated slices to the total number of active (i.e., currently arrived) slices. A comparison of SMRL and ObSI regarding acceptance ratio is presented in Figure 4. The graph displays also the outcome of a scenario where no MTD approaches are used, and slices are randomly allocated. The graph is in descending order because as the number of active slices increases, the core network's ability to cover them diminishes. The acceptance ratio is low if the slices are randomly allocated without mutations. On the other hand, both SMRL and ObSI have tried their best to allocate as many slices as possible. Hence, there is no significant difference between their performance in terms of acceptance ratio. The average acceptance ratios of SMRL and ObSI are 64.17% and 64.34%, respectively. We have reported the results of the acceptance ratio only to check whether SMRL decreases the acceptance ratio too much.

C. Requests distribution

A good allocation solution distributes the slices among all the physical machines so that a single machine is neither too idle nor too busy handling the slices. For this reason, we can analyze the request distribution. We define the request distribution as the variance of $slic()$ for all the machines. This is calculated based on Equation 9, where $\overline{slic}$ is the average value of $slic()$ for all i.

$$\text{Requests Distribution} = \frac{\sum_{i=1}^{P} (slic(\mathcal{N}, i) - \overline{slic})^2}{P - 1} \tag{9}$$

For example, the request distributions of Figure 1 and Figure 2 are 1.7 and 0.2, respectively. The lowest values of variance indicate a more complete distribution. Therefore, a reasonable allocation solution has lower request distribution values than others. Figure 5 illustrates the comparison of requests distribution between SMRL and ObSI. When the number of active slices is too high, many are not successfully allocated, and the remaining ones are easier to handle. So, the request distribution in this case is low. On the other hand, if the number of requests is too low, they also become easier to handle, and the same situation happens. In the other cases, the allocation solution has to handle many slices, and it is hard to keep their distribution perfect. As a result, this graph has a bell curve. Moreover, we can see that SMRL achieves a lower value of request distribution in all the scenarios, which means SMRL is more powerful than ObSI in distributing the slices among all the machines. The average results show that SMRL has improved the request distribution by about 20% compared to ObSI.

VI. Conclusion and Discussion

Network slicing is one of the key processes in 5G and beyond. The slices support the users' custom requirements for infrastructure resources. The slices are vulnerable to DDoS attacks that make them unavailable. In this paper, we have proposed a slice mutation technique based on RL (SMRL) that first randomly allocates the slices on the physical machines and then, in each time interval, performs a mutation to achieve two goals: (1) reducing the impact of a DDoS attack against a specific slice on the other slices, and (2) keeping the number of successfully allocated slices acceptable. The proposed RL model utilizes ternary and ranking numbers to represent the remaining CPU capacities and the number of currently allocated slices for each physical machine, respectively. Using these numbers, the proposed RL model is generally to be utilized by multiple core networks. We have evaluated the goal achievement of SMRL by three metrics: affected slices ratio requests acceptance ratio, and requests distribution. The average results obtained from the simulations say that SMRL has reduced the affected slices and requests distribution by 46% and 20%, respectively, while keeping the acceptance ratio unchanged. This paper's future work will consider the links and bandwidth capacities for representing the core networks and the slice requests. Moreover, we are eager to analyze the impact of different mutation intervals on the performance of SMRL. Integrating honeypot and encryption as a service (EaaS) techniques with the SMRL model offers several promising directions for future research in securing network slices in 5G and beyond [3, 27, 28, 29, 30]. Combining EaaS with this approach increases security by encrypting end-to-end communications, ensuring that even if attackers intercept traffic, they cannot access sensitive information. Additionally, encrypted honeypots can simulate high-value targets and divert attention away from operational breaches while maintaining strong data protection. A mathematical model can also be developed to optimize the placement of honeypots in mutated slices and determine the effect of honeypot deception on the overall network resilience. Finally, incorporating EaaS and honeypots into core network demonstrations, where honeypots simulate different bandwidth capacities and cut requests, provides another layer of deception and security, especially against DDoS attacks. This hybrid approach strengthens the defense of the network-slicing environment while maintaining acceptable performance levels.

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] S. Zhang, "An overview of network slicing for 5g," *IEEE Wireless Communications*, vol. 26, no. 3, pp. 111–117, 2019.

[2] C.-C. Liu and L.-D. Chou, "5g/b5g network slice management via staged reinforcement learning," *IEEE Access*, 2023.

[3] 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.

[4] A. Javadpour, F. Ja'Fari, T. Taleb, and C. Benzaïd, "A mathematical model for analyzing honeynets and their cyber deception techniques," in *2023 27th International Conference on Engineering of Complex Computer Systems (ICECCS)*, 2023, pp. 81–88.

[5] R. Khan, P. Kumar, D. N. K. Jayakody, and M. Liyanage, "A survey on security and privacy of 5g technologies: Potential solutions, recent advancements, and future directions," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 196–248, 2019.

[6] C. Benzaïd, T. Taleb, A. Sami, and O. Hireche, "A deep transfer learning-powered edos detection mechanism for 5g and beyond network slicing," in *GLOBECOM 2023 IEEE Global Communications Conference*, 2023, pp. 3963–3979.

[7] M. C. Hlophe and B. T. Maharaj, "An sdn controller-based network slicing scheme using constrained reinforcement learning," *IEEE Access*, vol. 10, pp. 134 848–134 869, 2022.

[8] W. Soussi, M. Christopoulou, G. Xilouris, and G. Gür, "Moving target defense as a proactive defense element for beyond 5g," *IEEE Communications Standards Magazine*, vol. 5, no. 3, pp. 72–79, 2021.

[9] 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.

[10] C. Benzaïd, T. Taleb, A. Sami, and O. Hireche, "Fortisedos: A deep transfer learning-empowered economical denial of sustainability detection framework for cloud-native network slicing," *IEEE Transactions on Dependable and Secure Computing*, 2023.

[11] 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.

[12] 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.

[13] A. Thantharate, R. Paropkari, V. Walunj, C. Beard, and P. Kankariya, "Secure5g: A deep learning framework towards a secure network slicing in 5g and beyond," in *2020 10th annual computing and communication workshop and conference (CCWC)*. IEEE, 2020, pp. 0852–0857.

[14] N. A. E. Kuadey, G. T. Maale, T. Kwantwi, G. Sun, and G. Liu, "Deepsecure: Detection of distributed denial of service attacks on 5g network slicing—deep learning approach," *IEEE Wireless Communications Letters*, vol. 11, no. 3, pp. 488–492, 2021.

[15] B. Bousalem, V. F. Silva, R. Langar, and S. Cherrier, "Deep learning-based approach for ddos attacks detection and mitigation in 5g and beyond mobile networks," in *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*. IEEE, 2022, pp. 228–230.

[16] H. Moudoud, L. Khoukhi, and S. Cherkaoui, "Prediction and detection of fdia and ddos attacks in 5g enabled iot," *IEEE Network*, vol. 35, no. 2, pp. 194–201, 2020.

[17] Y. Niu, G. Feng, Y. Li, Z. Liang, S. Zheng, Y. Zhao, and J. Zhang, "mmtc slice mapping under ddos attack in 5g ran," in *2022 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*. IEEE, 2022, pp. 588–591.

[18] e. a. Sattar, "Towards secure slicing: Using slice isolation to mitigate ddos attacks on 5g core network slices," in *2019 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2019, pp. 82–90.

[19] 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.

[20] A. Javadpour, G. Wang, and S. Rezaei, "Resource management in a peer to peer cloud network for iot," *Wireless Personal Communications*, vol. 115, no. 3, pp. 2471–2488, 2020.

[21] W. F. Villota-Jacome, O. M. C. Rendon, and N. L. da Fonseca, "Admission control for 5g core network slicing based on deep reinforcement learning," *IEEE Systems Journal*, vol. 16, no. 3, pp. 4686–4697, 2022.

[22] 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.

[23] M. R. Raza, C. Natalino, P. Öhlen, L. Wosinska, and P. Monti, "Reinforcement learning for slicing in a 5g flexible ran," *Journal of Lightwave Technology*, vol. 37, no. 20, pp. 5161–5169, 2019.

[24] K. Suh, S. Kim, Y. Ahn, S. Kim, H. Ju, and B. Shim, "Deep reinforcement learning-based network slicing for beyond 5g," *IEEE Access*, vol. 10, pp. 7384–7395, 2022.

[25] A. Javadpour, F. Ja'fari, T. Taleb, H. Ahmadi, and C. Benzaïd, "Cybersecurity fusion: Leveraging mafia game tactics and reinforcement learning for botnet detection," in *GLOBECOM 2023-2023 IEEE Global Communications Conference*. IEEE, 2023, pp. 6005–6011.

[26] A. Javadpour, F. Ja'fari, P. Pinto, and W. Zhang, "Mapping and embedding infrastructure resource management in software defined networks," *Cluster Computing*, vol. 26, no. 1, pp. 461–475, 2023.

[27] A. Javadpour, F. Ja'fari, T. Taleb, C. Benzaïd, Y. Bin, and Y. Zhao, "Encryption as a service (eaas): Introducing the full-cloud-fog architecture for enhanced performance and security," *IEEE Internet of Things Journal*, pp. 1–1, 2024.

[28] A. Javadpour, F. Ja'fari, T. Taleb, C. Benzaid, L. Rosa, P. Tomas, and L. Cordeiro, "Deploying testbed docker-based application for encryption as a service in kubernetes," in *Proceedings of the 32nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2024)*, September 2024.

[29] A. Javadpour, F. Ja'fari, and T. Taleb, "Encryption as a service: A review of architectures and taxonomies," in *IFIP International Conference on Distributed Applications and Interoperable Systems*. Springer, 2024, pp. 36–44.

[30] 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.

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 featured in top-tier journals and conferences, including Globecom and IEEE Transactions.

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.

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.