Computers & Security 97 (2020) 101976

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Cost-effective moving target defense against DDoS attacks using trilateral game and multi-objective Markov decision processes

Yuyang Zhou[a,b,c], Guang Cheng[a,b,c,*], Shanqing Jiang[a,d], Yuyu Zhao[a,b,c], Zihan Chen[a,b,c]

[a] School of Cyber Science and Engineering, Southeast University, Nanjing, China
[b] Key Laboratory of Computer Network and Information Integration, Ministry of Education, Nanjing, China
[c] Jiangsu Provincial Key Laboratory of Computer Network Technology, Southeast University, Nanjing, China
[d] National Key Laboratory of Science and Technology on Information System Security, Beijing, China

ARTICLE INFO

Article history:
Received 26 April 2020
Revised 22 June 2020
Accepted 21 July 2020
Available online 22 July 2020

Keywords:
Moving target defense
Trilateral game
Cost-effective
Multi-objective Markov decision processes
DDoS attacks

ABSTRACT

Moving Target Defense (MTD) has emerged as a game changer to reverse the asymmetric situation between attackers and defenders, and as one of the most effective countermeasures to mitigate DDoS attacks, shuffling-based MTD has gained ever-growing attention in cyber security. Despite the increased security, frequent shuffles would significantly bring heavy burden to the system. Moreover, most existing work has not adequately considered the impact of MTD techniques on the defender, and especially ignored that on legitimate users. Due to the lack of cost-effective shuffling methods, it is difficult to reach the optimal balance between the performance and overhead associated with the MTD deployment. Building on our preliminary work in this field, we propose a novel cost-effective shuffling method, which involves common users as a trilateral game for strategy generation and resists DDoS attacks with several MTD mechanisms. The novel game model extends our previous work to further describe the interaction among the attacker, the defender and users in detail, and we exploit Multi-Objective Markov Decision Processes to find the optimal MTD strategy by solving the trade-off problem between the effectiveness and cost of shuffling. By designing a trilateral game cost-effective shuffling algorithm, we capture the best MTD strategy and reach a balance between them in a given shuffling scenario. Simulation and experiments on an experimental software-defined network (SDN) indicate that our approach can effectively mitigate DDoS attacks with an acceptable overload, and exhibit better performance than other related and state of the art approaches.

1. Introduction

Distributed Denial-of-Service (DDoS) attacks (Wang et al., 2017), which are intended to prevent legitimate users from accessing specific network resources, have been considered as one of the largest unsolved and most serious threats to cyber security. Even though there are a number of DDoS defense mechanisms available (e.g., intrusion detection system), DDoS attacks can easily bypass traditional defenses by leveraging many IP addresses as the source of their packet flood or structuring attack packets to mimic legitimate traffic. Specifically, adversaries can take control of networks of compromised and remotely controlled hosts, known as botnets (Albanese et al., 2018), to send a large volume of illegitimate requests to the target servers, overwhelming their resources and degrading performance for users (Wright et al., 2016).

Under normal circumstances, cyber attacks are launched after reconnaissance efforts aimed at scanning the attack surface and collecting critical information about the target system (Sugrim et al., 2018). Unfortunately, cyber configurations are typically deterministic, static and homogeneous (Blakely et al., 2019), which brings adversaries asymmetric advantages over the defenders. In detail, adversaries may systematically probe target networks to gather additional information after the initial compromise, and launch attacks at their chosen time point to exploit the discovered vulnerabilities (Carvalho and Ford, 2014). On one hand, due to the passive position, it is difficult for a traditional defender to protect all resources from all possible DDoS attacks at all times. On the other hand, the traditional strategy (e.g., firewalls) relies on knowing the characteristics of attacks to defend the target system. It becomes inefficient and insufficient when facing more advanced DDoS attacks with unknown patterns or launched from botnets, which is common in today's cyber attacks.

* Corresponding author at: School of Cyber Science and Engineering, Southeast University, Nanjing, China.
E-mail addresses: yyzhou@njnet.edu.cn (Y. Zhou), gcheng@njnet.edu.cn (G. Cheng), sqjiang@njnet.edu.cn (S. Jiang), yyzhao@njnet.edu.cn (Y. Zhao), zhchen@njnet.edu.cn (Z. Chen).

https://doi.org/10.1016/j.cose.2020.101976
0167-4048/

In order to reverse the asymmetric situation between the attacker and the defender, Moving Target Defense (MTD) (Cai et al., 2016; Lei et al., 2018a) has recently emerged as a game changer in the field of cyber security. MTD regularly or strategically changes system attack surfaces to reduce attackers' understanding of the target system, forcing them to continually reassess and replan their cyber attack operations. Any discovered vulnerabilities may disappear after enough time has passed, thus reducing the chance of a successful exploit. Essentially, the shuffling-based MTD significantly increases attack cost/complexity, which makes it far more difficult for an adversary to launch a successful attack (Manadhata and Wing, 2004). Therefore, many MTD based methods have been recently proposed to help mitigate DDoS attacks. In general, these methods allow the defender to dynamically reconfigure clients or migrate them among a large pool of proxy servers, and adversaries can strike specific targets only if they know which ones are in use at the time.

Although there are some studies (Bardas et al., 2017; Hong and Kim, 2015) proposing evaluation and optimization of shuffling-based MTD strategies in the literature, they mostly focus on solutions in which the security of the system is greatly improved. We argue that the strategy of MTD mechanisms might not be optimal for a defender considering that in some cases the defense costs have been ignored when evaluating its utility. In addition, an MTD mechanism with a high frequency of shuffling has inevitable negative effects on the system, such as reducing the quality of service (QoS) on top of the extra costs associated (Zangeneh and Shajari, 2018), which may also affect the stable access to the system from legal users. Therefore, there is a need to design cost-effective solutions that can strengthen the system from the attacks while restricting defense overhead and bringing a minimal service degradation to common users.

In our previous work (Zhou et al., 2019), we modeled the game between the attacker and the defender and discussed the game process and game payoff to guide the defender to analyze the impact of different MTD strategies on the protected system. Then, the Multi-Objective Markov Decision Processes (Hahn et al., 2019) was exploited to solve the trade-off problem between the effectiveness and cost for MTD shuffling. Thus, a cost-effective shuffling method was proposed to reach the best strategy and resist DDoS attacks using MTD mechanisms, such as IP hopping, port hopping and migration. Although that preliminary work represents the first important step towards a comprehensive solution to the problem of cost-effectiveness balance when utilizing MTD against DDoS attacks, several limitations still exist. First, as shown in Zhou et al. (2019), the shuffling-based MTD mechanisms may introduce a costly overhead to legitimate users and cause a backlog of service requests, which has been ignored in respect of making shuffling decisions in our previous work. Hence, users ineluctably become the third party of this game, and their overhead must be taken into consideration when deploying MTD mechanisms. Second, the game strategies of both players in prior work have been designed in advance, which may not match the actual situation and bias the decisions. Third, the previously proposed shuffling algorithm lacks consideration of all conditions, and may make extreme decisions for several high load VMs, which will lead these users to have to suffer from attacks.

In this paper, in order to ensure the best cost-effectiveness balance of MTD shuffling, we introduce a novel trilateral game theory that can involve legitimate users as special participants of the game. We significantly revised our previous game model to capture the effects of different game strategies on the performance and overhead of shuffling-based MTD mechanisms, especially, the users' overhead has been fully discussed at the same time. In addition, to accurately reflect the decision-making situations of all game players, heuristic strategies have been used to replace those in previous work for all players. Finally, a novel cost-effective shuffling algorithm has been proposed to find the optimal strategy that balances the effectiveness and cost of shuffling-based MTD, and we also extend experiments with additional strategies and more kinds of attacks to prove the advantage of our method in resisting DDoS attacks with limited overhead. In summary, the main contributions of this work are listed as follows:

- We model the interaction among tripartite participants with heuristic strategies as a sequential game and then exploit Multi-Objective Markov Decision Processes (MOMDP) for solving the multi-stage stochastic optimization of decision problems.
- We propose an MTD shuffling scenario to increase the applicability of the proposed method, and it can help to further analyze and quantitatively evaluate the detailed payoffs of different game players.
- We present a novel Trilateral Game Cost-Effective Shuffling algorithm (TCS) to find the optimal strategy for a sequence of decisions and reach a trade-off between the effectiveness and the cost of MTD while guaranteeing legitimate users' access to the service.
- The proposal is compared with state of the art methods on an experimental SDN testbed. Simulation and experimental results have shown that our method can effectively shuffle with limited cost and performs well in resisting DDoS attacks.

The remainder of this paper is organized as follows. We discuss the related work in Section 2 and propose the threat model in Section 3. Model specification and detailed analysis of the game are presented in Section 4. Description of the shuffling scenario and algorithm is given in Section 5. The performance of our proposed method is evaluated via simulation and experiment in Section 6. Finally, we conclude the paper in Section 7.

2. Related work

In this section, we present the existing literatures on MTD shuffling mechanisms, MTD for DDoS attacks, evaluation for MTD, and game-based MTD strategies.

2.1. MTD shuffling mechanisms

The shuffling-based MTD approaches dynamically reconfigure the network attributes over time to ensure the security of the system. The goal of shuffling the attack surface is to prevent an attacker from collecting the information of target networks such as scanning for open-ports, and sending non-malicious traffic to uncover system topology or discover vulnerabilities. A number of shuffling-based MTD approaches have been proposed and can be classified into random, event-based, and hybrid mutation. Early researches on random mutations (Chang et al., 2018; Gillani et al., 2015) stipulate that each shuffle in a random mutation occurs after a set time interval which could be random or periodic. The time interval would be the only information needed in this case, which can be easily speculated once the attacker understands the shuffling rules. In contrast, the moves in event-based mutations (Crouse et al., 2015; Zhang et al., 2017) rely on extra information such as security policies and alerts to trigger shuffling mechanisms. Upon receiving an external stimulus, the attack surface would be shuffled in order to mitigate the event. Even though unnecessary overhead can be avoided in event-based mutations, it does not help the defender in terms of security when facing unknown attacks that can bypass the detection. Therefore, hybrid mutations offer a mixed approach which combines many aspects of random and event-based mutations. Several researchers have proposed hybrid MTD models (Huang and Ghosh, 2011; Kampanakis et al.,

2014) that shuffle either at certain intervals or through events. The combination of random and event-based mutations makes up for each other's deficiencies, and thus provides proactive defense against adaptive adversaries while keeping the overhead at a lower level. However, traditional MTD shuffling mechanisms mostly move from the point of view of the defender. Due to lack of description of the attacker's behaviors, it is uncertain whether the target network is incomplete or noisy for the attacker.

2.2. MTD for DDoS attacks

Due to the capabilities of elastic capacity provisioned in the cloud platforms, many MTD techniques have been implemented to mitigate DDoS attacks recently. Some researches (Jia et al., 2014; Wang et al., 2014) have focused on MTD techniques that deploy proxies between clients and servers, and reconfigure them to disrupt knowledge accumulated by adversaries, either proactively or in response to detected threats. Likewise, the goal of MTD-based DDoS mitigation methods is to force attackers to strike the wrong target with additional costs, and thereby protecting the real network assets. In order to redirect legitimate users to secure VMs and restrict attackers in quarantine ones, Venkatesan et al. (2016) proposed a new proxy assignment strategy to isolate compromised servers, thereby reducing the impact of attacks. Moreover, an SDN-based MTD mechanism to defend against a type of DDoS attacks called Crossfire was proposed by Aydeger et al. (2019), it reorganized the routes in such a way that the congested links are avoided during packet forwarding. In the context of detection and defense of DDoS attacks, Liu et al. (2018) combined the programmability of SDN and the flexibility of network function virtualization (NFV) for the improvement of system security. In addition, authors in Steinberger et al. (2018) proposed a collaborative and scalable DDoS mitigation approach that combines MTD and SDN to limit the effects caused by large-scale DDoS attacks, and then integrated the low cost solution into existing system infrastructure. Although MTD-based methods can improve the system's ability to resist DDoS attacks, in some extreme cases, they may exhibit weak performance but with much overhead.

2.3. Evaluation for MTD

Meanwhile, a rich line of research was proposed to evaluate MTD mechanisms by quantifying the changes on the attack surface and assessing the performance of the mutations (Alavizadeh et al., 2018; 2019). Combining various types of MTD assessments, Leeuwen et al. (2016) summarized prior approaches to the evaluation of MTD techniques and developed a single hybrid experiment for analysis of the various aspects of MTD approaches. To assess the effectiveness of MTD in a quantitative way, Bopche and Mehtre (2017) employed classical graph distance metrics such as maximum common subgraph (MCS) and graph edit distance (GED) to measure temporal changes in attack surface of dynamic networks. Hong et al. (2018) also proposed a temporal graph-based security model and developed a new set of dynamic security metrics to assess and compare their effectiveness. Different from those temporal models, Connell et al. (2018) proposed a method that involves a utility function to capture the trade-off between security and performance, and thereby evaluating the resource availability and security of MTD mechanisms. Moreover, system attack surface (SAS) (Xiong et al., 2019) and analytic hierarchy process (AHP) (Zhang et al., 2019) based evaluation models have also been proposed to quantify the effectiveness and costs of MTD. In order to thoroughly analyze the impact of MTD, most studies have shifted from qualitative to quantitative evaluation of MTD, however, abstract models always hinder the methods from being actually deployed.

2.4. Game-based MTD strategies

In order to learn MTD from the perspective of both sides, some researchers have adopted game theory (Cybenko et al., 2019; Tan et al., 2019; Wang et al., 2018) to model the interaction between the attacker and the defender and determine the selection of MTD strategy. Prakash and Wellman (2015) employed empirical and game theoretic techniques to examine the game process between both players. Although they showed that security alerts play an important role in effective move selection, the cost of the moves was ignored. Feng et al. (2017) proposed a Bayesian Stackelberg game that models the signaling strategies for the defender, and theoretically proved that defensive advantages can be established through strategic information disclosure. Similarly, a repeated Bayesian Stackelberg game was utilized by Wahab et al. (2019) to optimize the balance between detection load and the accuracy of multi-type attacks. Moreover, Markov Decision Process (MDP) based approaches have been utilized to analyze and further select policies by some researchers (Hu et al., 2017; Miehling et al., 2015). In Lei et al. (2017), a Markov game based optimal MTD strategy selection method was proposed to balance the defensive revenue and network service quality. In the model, the attack surface changes with the exploitation of network vulnerability, and the effectiveness of the method has been proved by case study. Building on this work, Lei et al. (2018b) then proposed an incomplete information Markov game approach which is more realistic for strategy generation. Although the proposed model has been examined via theoretical analysis and numerical study, the game process is built only between the attacker and defender as well.

2.5. Discussion

Based on the literature review given above, it can be argued that the vast majority of the works use shuffling-based MTD with the aim of coming up with a dynamic attack surface which could increase the system security. Different from that observation, we not only take advantage of shuffling mechanisms and SDN with the purpose of resisting DDoS attacks effectively, but also but also fully discuss the overhead in our solution. Although some researches have proposed various models to evaluate the effectiveness of MTD, the great mass of the proposed solutions do not meet the feasibility and universality, as they apply abstract model for assessment while ignoring the defense cost of MTD. Contrary to this, our work employs a specific shuffling scenario, which can help construct assessment models and determine the parameters, thus the effectiveness and cost of the proposal can be evaluated quantitatively.

Another observation is that nearly all the works consider only the attacker and the defender as the game players for solving the decision-making problems under the MTD game. Opposite to this observation, our solution takes the user overhead into consideration and treats legitimate users as special participators in a novel trilateral game. Similarly, the traditional MDP is not suitable for the novel game model, thereby the MOMDP has been exploited for the multi-objective optimization of strategies in our solution. In addition, it is also worth noting that in an effort to deal with the optimization problems, the great mass of approaches use theoretical analysis or numerical studies for examining the proposed systems under a limited number of metrics. On the bright side, our solution is evaluated on an SDN testbed through both simulation and experiments, and achieves better performance in comparison with state of the art methods.

Finally, one could say that the problem of best MTD strategy remains largely an open issue. Namely, some solutions are not able to describe this kind of interactions while others do, but the model they present is still not accurate. To cope with this shortcoming, the work at hand has put considerable effort into optimizing the cost-effective trilateral game model and developing quantitative evaluations that are able to conform to reality.

3. Threat model

In this section, we describe a threat model to characterize the behavior of attackers, the moving target defense mechanism and the reaction of common users. We assume that the network platform and the service provider are trusted, and meanwhile attackers come from external network. The rational and tactful adversary may also have multiple network resources to scan and probe before strategy execution, although they may not utilize all of it when attacking targets. We also assume that the defender might take advantage of some defense mechanism to prevent the target system from being compromised. In addition, common users, who play the role of the third party, may pay more attention to the performance of the system service rather than the actions and strategies of the other two participants.

3.1. Attacker behavior

A strategic and rational attacker, with the objective of maximizing the number of machines that can be compromised in the target network, always needs to obtain some sensitive parameters about the defenders before launching a successful attack. In order to gain knowledge of the protected system, an attacker may take the time, computing, and monetary resources to conduct reconnaissance operations that can be either passive (e.g., sniffing on an interface) or active (e.g., pinging a host). Once enough information about the defender has been obtained, the attack will be launched with characteristics that are systematically decided by the current system state as well as the defense actions. The whole procedure including probing and launching the attack incurs significant cost. For example, the cost of launching a DDoS attack will be related to the resources consumed by previous IP address scanning, stealthy port scanning and the amount of utilized clients when the attack happens.

3.2. Defense mechanism

In order to guard a system from being hacked or destroyed, the defender has to collect the information about the whole system and any suspicious reconnaissance behavior that may lead to risks. Using moving target defenses to safeguard the system, the defender needs to make shuffles to change the attack surface as well as take other necessary measures against an attacker. For each shuffle, it incurs a shuffle cost due to the utilized computing and network resources. We assume the occurrence of a DDoS attack can be easily detected and the defender can acquire the information about the compromised client within a very short time-frame. In detail, this paper focuses on shuffle based MTD techniques that can be implemented at network level, and the DDoS detection techniques are beyond the scope of this paper.

Therefore, the defense mechanism is defined as follows. Once one or several hosts in the protected system are compromised, in order to prevent the follow-up, the defender will shuffle the exploited hosts by the following defense types.

- Port hopping: Dynamic and continuous change of port number of a particular service.
- IP hopping: The defender changes the IP address of a virtual machine (VM) dynamically and incessantly.
- Migration: The defender migrates the applications or services under attack between VMs.

3.3. User behavior

The shuffle based MTD method periodically reconfigures the network system or migrates the compromised VMs, which may introduce a costly overhead to legitimate users and cause a backlog of service requests. Different from traditional passive defense, the overhead incurred by frequent shuffle is significant for the defender to make shuffling decisions. Meanwhile, high service latency or interruption during the whole shuffle process also forces users to choose other servers, which may lead to system overloads. Hence, it is essential to take the user behavior into consideration and we assume that the users have the following capabilities:

- Users can not know the behavior and decisions of the attacker and the defender.
- Users do not possess the defense skills and only care about the service state and system performance.
- Users would choose another one if the service delay exceeds their expected threshold value or an interruption happens.

3.4. Objective

The objective of this paper about trilateral cost-effective shuffling MTD method is to investigate the optimal way for a cyber defender to make decisions, while taking into account the shuffling/attack cost, effectiveness and service overhead among three parties, including the defender, the attacker and users. It is important to maximize the effectiveness and minimize the shuffling cost and users' overhead, while restricting the attacker's payoff and forcing them to terminate the attack. Moreover, it is possible for a defender to endure risks without shuffling, if the shuffling cost is high or the shuffle may cause service interruption while the effectiveness is low. We seek to examine what is the best way to make the shuffling decision and how to reach the best trade-off between cost and effectiveness.

4. MTD Game model

As discussed in Zhuang et al. (2014), one of the essential MTD problems is adaptation selection problem, also known as "how to adapt". However, a major drawback of many MTDs is that they only seek the adaptation of improving the effectiveness of the protected system, which may introduce the neglection of the adaptation time and costs. There could be multiple sequence of actions that could lead to the same system state, and the complex adaptations may not perform better than simple actions in some cases. Therefore, the strategy would require to take constraints such as time and costs into consideration, and the defender should make the best decision at each time in order to optimize both the effectiveness and cost of MTD.

In this sequential game, the defender adopts an MTD strategy by migrating the resource across the network to make it difficult for the attacker to identify the real location of the resource, while the attacker may observe the defender's actions by monitoring network traffic. Knowing this strategy (but not its realization), the attacker then determines against which VM to conduct DDoS attacks and which IP address to choose. The defender can detect the occurrence of a DDoS attack and partially obtain the attacker's actions by observation, while users may take actions including quitting current client and turning to another one when the service delay is high. Thus, the trilateral game in this paper is incoordinate in which the attacker and the defender play their best strategy to act against each other, and users could not understand the behaviors of the other two players and only make decisions according to

the service state. Likewise, the cost of each player would be generated from actions they take, while the user could not possess any personal reward due to passive participation in the trilateral game.

Definition 1. S represents a finite set of states, including all possible attack surfaces that the protected system could experience and let $S_t(v)$ be the state of the specific VM v at the time step t, where $t \in \{0, \ldots, T\}$ and T is the time horizon of the game. In detail, there are two possible values for every $S_t(v)$, where $S_t(v) = 0$ represents a secure VM whereas $S_t(v) = 1$ means this VM is under DDoS attack at this time.

Definition 2. O represents the status of the protected system by attacker's observation. Likewise, $O_t(v)$ represents the state of the observed VM v at the time step t, and it shares the same value as $S_t(v)$ can take. However, different from the defender's global perspective of the protected system, we assume that the attacker can obtain the state information of target VM by some methods (e.g. monitoring the traffic). Since the attacker may not explore all VMs at one time, O_t only contains the status of partial VMs, which can be represented as $O_t \subseteq S_t$.

Definition 3. $\mathbb{A}$ is the finite set of actions that all players need to take so as to maximize their perspective payoffs during the game. It consists of <A, D, U, t>, which is attacker actions, defender actions, user actions and time step t. Since players take actions in the process of the trilateral game, their actions during the time steps of the game can be presented as $\mathbb{A} = \{(A_0, D_0, U_0), (A_1, D_1, U_1), \ldots, (A_T, D_T, U_T)\}$. $A_t(v)$ represents the set of attacker actions on the VM v at time step t, such as sniffing, scanning or striking a VM. Likewise, $D_t(v)$ and $U_t(v)$ separately represent the sets of actions applied on VM v that defender and user take at time step t. In addition, $A_t(v)/D_t(v)/U_t(v) = 0$ represents that they have no action, otherwise they have done at least one thing on the target VM when $A_t(v)/D_t(v)/U_t(v) = 1$.

Definition 4. $f_E^{A/D/U}(v)$ is an effectiveness function that determines the effectiveness value E for different game players that they can obtain according to their actions on the VM v and the states of the VM before and after the specific action. It can be represented as $f_E^{A/D/U}(v) : S \times A/D/U \times S \to E$, where $A/D/U$ is the set of actions that attacker/defender/user takes in the game.

Definition 5. $f_C^{A/D/U}(v)$ is a cost function that maps an action on VM v to a cost value C for the game players. It can be represented as $f_C^{A/D/U}(v) : A/D/U \to C$, and similarly, $A/D/U$ separately represents a set of actions that different game players can take.

4.1. Game process

At the beginning of the game, S_0, A_0, D_0 and U_0 need to be initialized with $\varnothing$. Based on our assumption, the defender is fully aware of system state at every time step, whereas the attacker only knows the state O_t through observation. Thus, we also set $O_0 = \varnothing$. In addition, as discussed in Section 3.2, the defender may need to analyze the system to obtain the subsequent states when the reconnaissance is in progress, but can acquire the information about the occurrence of a DDoS attack in a short time.

At each time step $t + 1 \in \{1, \ldots, T\}$, the attacker can choose any VM $v \in V$ to conduct a DDoS attack with a success probability $p(v)$, and $p(v, v')$ from VM v to v' if the attacker has taken control of v. Simultaneously, the defender decides which VMs to shuffle to prevent the attacker from further intruding, and users may suffer from high service delay due to shuffles.

After the initialization, the game proceeds in discrete time steps, $t + 1 \in \{1, \ldots, T\}$, with both the attacker and the defender aware of the current time. As a special participator in the trilateral game, the user takes actions without knowing other players' behaviors, and may choose to stay on current VM if the service is still available. The following sequence of game events among three players occurs at each time step $t + 1$.

(1) The defender observes S_t, while the attacker observes O_t based on the state of the protected system.

(2) The defender and attacker select their actions A_{t+1} and D_{t+1} according to their respective strategies.

(3) The user takes an action U_{t+1} based on the performance of system service after other players' strategies have been implemented.

(4) The system transits to its next state S_{t+1} according to the transition function (Algorithm 1).

(5) All players evaluate their rewards (the user does not possess any reward) and costs for the time step, respectively.

(6) All players enter the next time step unless the time step T has arrived.

In the STF (Algorithm 1), Line 1 judges whether the secure VM has been monitored by the attacker and attack actions at next time are given in Line 2. Then, MTD will actually work in Line 4 if the defender shuffles these target VMs while users keep access to them, otherwise, there is a probability that the VM will crash and even the attack may spread to other VMs, which can be seen in Line 6 and Line 7. Moreover, if there exists no observation or several VMs have already been under DDoS attacks, we set the system state unchanged for the next time step in Line 10, and a previous judgement has been set in Line 11. Therefore, Line 12 indicates that the VMs can be well protected when the shuffling-based MTD has been deployed on them. On the contrary, they will be at a risk

Algorithm 1 State transition function (STF).

Input:
The system state at time step t, S_t;
The observation by attacker at time step t, O_t;
The defender action at time step $t + 1$, D_{t+1};
The attacker action at time step $t + 1$, A_{t+1};
The user action at time step $t + 1$, U_{t+1};
Output:
The system state probability distribution at time step $t + 1$, S_{t+1} with probability p;

1: **if** $O_t(v) \subseteq S_t(v) = 0$ **then**
2: $A_{t+1}(v) \leftarrow 1$;
3: **if** $v \in D_{t+1}(v) \cap v \notin U_{t+1}(v)$ **then**
4: $S_{t+1}(v) \leftarrow 0$;
5: **else**
6: with probability $p(v)$, $S_{t+1}(v) \leftarrow 1$;
7: with probability $p(v')$, $S_{t+1}(v') \leftarrow 1$;
8: **end if**
9: **else**
10: $S_{t+1}(v) \leftarrow S_t(v)$;
11: **if** $v \in D_{t+1}(v) \cap v \in A_{t+1}(v) \cap v \notin U_{t+1}(v)$ **then**
12: $S_{t+1}(v) \leftarrow 0$;
13: **else**
14: **if** $v \notin D_{t+1}(v) \cap v \in A_{t+1}(v)$ **then**
15: with probability $p(v)$, $S_{t+1}(v) \leftarrow 1$;
16: **else**
17: **for** $v \notin D_{t+1}(v) \cap v \notin A_{t+1}(v)$ **do**
18: with probability $p(v', v)$, $S_{t+1}(v) \leftarrow 1$;
19: **end for**
20: **end if**
21: with probability $p(v')$, $S_{t+1}(v') \leftarrow 1$;
22: **end if**
23: **end if**
24: **return** S_{t+1} with probability distribution p

without MTD mechanisms, which is shown in Line 15. Even for the VM that both the defender and attacker ignore, we assume that there may also be a system breakdown caused by another VM in Line 18, or by itself in Line 21. Finally, the system state probability distribution at time step $t+1$, including status information of all individual VMs, is returned in Line 24.

4.2. Game payoff

As discussed in Section 4.1, S_t is a system state at time step t, when the attacker plays A_t, the defender plays D_t, the user plays U_t, and the previous system state is S_{t-1}. Moreover, effectiveness and cost functions have directly relative to system states and players' actions according to their definitions. In order to simplify the operation of them, therefore, we denote H_t by the game history, which can be defined as follows:

Definition 6. H is a finite set which represents the history of the game process, including the system states S of all VMs and players' actions $\mathbb{A}$ at different time steps. Since the time horizon of the game is T, let H_t be the game history ranging from initiation to the time step t, which can be presented as $H_t = \{(S_0, \mathbb{A}_0), ..., (S_t, \mathbb{A}_t)\}$, where $t \in \{0, \ldots, T\}$.

After all players have taken actions in the game, each of them will get either a negative or a positive return (the user could only obtain a negative return because of shuffle overheads). It is the quantitative assessment of each player's action which represents the game payoff. In traditional MTD, both the defender and the attacker need to take the payoff into consideration when they make defense or attack decisions. However, for a trilateral MTD game, the defender not only takes the defense payoff into account, but also needs to give a full consideration to the user's overhead. Each player then receives a payoff function and acts to increase their own expected payoffs. Meanwhile, the participant of the third side will greatly expand the solution space of the game, so that the trilateral game is more complex than the traditional interaction between the defender and the attacker.

With respect to H_t, the players' payoff values of two objectives, which include goal effectiveness values (the effectiveness value could be 0 for the user) and action costs, can be separately represented as follows:

$$P^d_{t+1} = E^d(H_{t+1}) - C^d(H_{t+1}) \tag{1a}$$

$$P^a_{t+1} = E^a(H_{t+1}) - C^a(H_{t+1}) \tag{1b}$$

$$P^u_{t+1} = -C^u(H_{t+1}) \tag{1c}$$

where P^d_{t+1}, P^a_{t+1}, and P^u_{t+1} are the payoff values of different game players' actions at time step t, and E and C separately represent the effectiveness and cost functions that are defined in Definition 4 and 5.

In Eqs. (1a) and (1b), since the defender and attacker may shuffle or strike several VMs at one time, and the evaluation of payoffs relies on the system states before and after taking actions, the payoff values for them should be calculated at time step $t+1$ with the game history H_{t+1}. Although the user's payoff value is still calculated based on H_{t+1}, there is a significant difference that Eq. (1c) only contains the cost function. Common users passively participate in the trilateral game and focus on the QoS rather than the system security, therefore, their actions bring no effectiveness to the protected system during the game process and make P^u_{t+1} always a negative number.

4.3. Game strategy

As discussed above, a strategy for the game players is a policy by which the player chooses when to execute its actions on what VMs, as a function of its game history H_T and the current time. However, the space of available game strategies is vast even with a single action type. In order to avoid direct exploration to the strategy space and analyze the game process more meticulously, we therefore focus on heuristic strategies that are defined by regular structures and patterns of behavior over time. We select a set of such strategies through an iterative process of strategy exploration and game analysis, and systematically depict actions of all players during the game process. Heuristic strategies in MTD can be divided into three types: time-based, event-based and payoff-based.

Time-based strategies, also known as periodic strategies, are generated deterministically during a specific time interval, or probabilistically according to a renewal process. For example, the attacker periodically scans and probes the attack surface to obtain the critical information about the target system. Strategies triggered by events may apply actions to the VM based on observations of this VM, or a combination of observations across the protected system. For instance, the user chooses another VM owing to the service interruption of the required VM. Due to the passivity of common users, payoff-based strategies only exist between the defender and the attacker, and usually occur in the end of current time step. The defender and the attacker may evaluate their payoffs of actions during this time step, and make action decisions (which VM to shuffle or attack) for the coming time step.

Since the attacker is rational and the interaction among three players is more complex, we therefore focus on payoff-based strategies to describe the heuristic selection strategies of the defender and the attacker. In addition, although the user's payoff can be calculated by the Eq. (1c), users themselves would not pay close attention to the payoffs but take actions due to sudden events such as high service latency or interruption. In order to make the user's strategies conform to the actual situation, event-based heuristic strategies would be considered to depict the user's actions during the trilateral game.

4.3.1. Attacker strategy

We consider two forms of heuristic attacker strategy defined by the payoff-based selection strategies. For the attackers, at time step $t+1$, based on O_t, they need to consider only VM $v \in V$ that can change the target system state at time step $t+1$. As mentioned in the Section 4.1, in our game, the attacker only knows the initial system state S_0, where $O_0(v) = S_0(v) = 0$ for each $v \in V$. Hence, we denote the *potential attack target* at time step $t+1$ by $\alpha(O_t)$ which represents this set of VMs and consists of two parts as follows:

(1) Target on VM v directly to launch an attack.

(2) Target on another VM v' with probability to reach v.

Based on the two parts of VMs discussed above, we obtain $\alpha(O_t)$ defined as follows:

$$\alpha(O_t) = \{v \in V | O_t(v) = 0\} \cup \left\{v' \in V | O_t(v') = 0, p(v', v) > 0\right\} \tag{2}$$

Since the attacker is rational in our assumption, the attacker selects heuristic strategies based on potential assessment of the payoff with $\alpha(O_t)$. Intuitively, the value of an attack payoff quantitatively represents what the attacker can obtain by this attack at the time step. Different from the game payoff calculated in the end of each time step, the potential payoff is considered as an expected return for the next time step and a guidance for the following game strategy.

The main idea of this payoff-based game strategy is to choose the attack target by which the attacker's potential payoff could be

maximized based on previous system state at each time step. However, due to lack of knowledge about the defender's action at this time step, the potential payoff the attacker calculates is biased for their unilateral action.

4.3.2. Defender strategy

Among the payoff-based strategies we also explore two different criteria for the selection of protected VMs. Since the defenders could not fully understand the true system states at each time step, it is crucial for them to reason through the possible system states based on their observations before committing to a defensive action.

The defender needs to take both their observation and their assumptions about the attacker strategy into consideration to form an understanding of the current system state. Similarly, we denote by $\beta(S_t)$ the *potential defend target* at time $t+1$ as follows:

(1)Target on VM v according to the system state S_t from the defender's observation.

(2)Target on VM v which is not in D_t.

According to the above analysis, we obtain $\beta(S_t)$ defined as follows:

$$\begin{aligned}\beta(S_t) = &\{v \in V | S_t(v) = 1\} \\ &\cup \{v \in V | S_t(v) = 0 \cap v \notin D_t(v)\}\end{aligned} \tag{3}$$

Meanwhile, before making decisions, rational defenders also need to assess the potential game payoffs of imminent actions with $\beta(S_t)$. The potential assessment of the game payoff for the defenders represents the quality of the strategy to fight against attackers' malicious actions for the next time step. Essentially, from the defenders' point of view, the higher the value of the defense payoff is, the safer the protected system will be.

4.3.3. User strategy

As special participators in the trilateral game, common users are considered to select strategies that generate actions based on the service performance and quality when accessing the system. As discussed in Section 3.3, users can not acquire the behavior and decisions of the attacker and the defender, and only pay attention to the service state and system performance. Different from the defender and the attacker selecting payoff-based heuristic strategies, users take actions along with performance fluctuations caused by system events, rather than evaluating their payoffs to make decisions. Considering that users pay more attention to service quality and other performance factors, we explore two different types of heuristic strategies at time step t by $\delta(S_t)$ for the user as follows:

(1)Keep the connection to the current VM v that provides required service.

(2)Turn to another online VM v' that provides the same service.

Therefore, we also conclude the definition of $\delta(S_t)$ as follows:

$$\delta(S_t) = \begin{cases} \delta(S_{t-1}) & \text{if } 0 \leqslant \mathbb{t}_t < \mathbb{T} \\ v' \in V \setminus \delta(S_{t-1}) & \text{else} \end{cases} \tag{4}$$

where $\mathbb{t}_t$ represents the time overhead at time step t induced by the attacker and defender, and $\mathbb{T}$ is the thresholding that common users can endure during their access to the system services.

The event-based strategies require users to make decisions based on the occurrence of a particular event, so that users select strategies at the end of each time step, which means they take actions after the other two players have done that. Users would continue current connection with the VM if the system service remains in a good condition. In the face of high service delay or system interruption, however, they would disconnect current connection and access another VM on their own initiative.

5. Trilateral game cost-effective shuffling method

As discussed above, we give the description of the game model and describe the game process and game strategies among the attacker, the defender and users. However, the game may reach an equilibrium which is undesirable for the defender. Aiming at making the game more beneficial for the defender and users, and reaching the best trade-off between shuffling cost and defense effectiveness, we propose a trilateral game cost-effective shuffling method, which consists of threat model and game theory, to adopt different shuffling types under different conditions.

5.1. MTD shuffling scenario

Different from traditional static defense scenario, when a service or a VM is under DDoS attacks, the MTD defender controls and relocates the ports, IPs or VMs in use from extra resources. Nevertheless, additional overhead is incurred in the procedure of several shuffles, such as consumption of network resources and increment of service delay. Therefore, our goal is to balance the defense effectiveness and the user overhead whereas restricting the attacker's payoff at the same time by the optimal implementation of a shuffling-based MTD method.

To increase the applicability of our shuffling method and expound the details more clearly, we make some assumptions and propose the shuffling-based MTD scenario as follows.

Given: a set of q users and a group of n online VMs with r network segments and u ports of equal resources for m users, where $m \times n = q, r \leqslant n$

Output: three sequences of matrices $(X_0, X_1, \ldots, X_T)$, $(Y_0, Y_1, \ldots, Y_T)$, $(Z_0, Z_1, \ldots, Z_T)$, where $X_t \in \{0, 1\}^{r \times n}$, $Y_t \in \{0, 1\}^{u \times n}$ and $Z_t \in \{0, 1\}^{q \times n}$, such that

$$\sum_{i=1}^{n} x_{ij}^t \geqslant 1 \quad j = 1, \ldots, r; \tag{5a}$$

$$\sum_{j=1}^{r} x_{ij}^t = 1 \quad i = 1, \ldots, n; \tag{5b}$$

$$\sum_{i=1}^{n} y_{ij}^t \leqslant n \quad j = 1, \ldots, u; \tag{5c}$$

$$\sum_{i=1}^{n} z_{ij}^t = 1 \quad j = 1, \ldots, q; \tag{5d}$$

$$\sum_{j=1}^{q} z_{ij}^t = m \quad i = 1, \ldots, n; \tag{5e}$$

The matrix X_0 denotes the initial IP assignment, and the matrices $\{X_t | 0 < t \leq T\}$ represent the IP shuffling decision at time step t, where binary variable x_{ij}^t indicates that whether the ith VM is assigned to jth network segment. Hence, Eq. (5a) states that each network segment owns at least one VM, and Eq. (5b) ensures that each VM is assigned to only one network segment. Similarly, the matrices Y_t represent the port shuffling decision at time step t, and y_{ij}^t is a boolean value that indicates that whether the ith VM is assigned to jth port. Therefore, we can easily get Eq. (5c) which indicates that at most n VMs can share the same port number in this shuffling scenario. Moreover, as the VM migration is the third shuffling mechanism, the matrices Z_t denote the overall condition of VM migration at time step t and the binary variable z_{ij}^t represents that whether the jth user is assigned to ith VM. Based on Eq. (5d) and Eq. (5e), we can conclude that each user is assigned to only one VM and each VM can only be allowed to serve m users.

5.2. MTD shuffling objectives

Since the goal of moving target defense is to limit the number of crashed VMs with less defense cost, we quantify the payoff values of shuffle within T time steps by the difference between the effectiveness and cost in every time step multiplied by a discount value γ^t, where the discounted value ensures that the defender's payoff is decayed over time. The objective function to maximize the payoff of shuffling is as follows:

$$f_D = Maximize \sum_{t+1=1}^{T} \gamma^t \left(E^d(H_{t+1}) - C^d(H_{t+1})\right), where \tag{6a}$$

$$E^d(H_{t+1}) = \sum_{v \in V} STF((S_t(v), H_{t+1}) | S_t(v) = 1), and \tag{6b}$$

$$C^d(H_{t+1}) = \sum_{v \in D_{t+1}} \left(w_x \sum_{j=1}^{r} |x_{vj}^{t+1} - x_{vj}^{t}| + w_y \sum_{j=1}^{u} |y_{vj}^{t+1} - y_{vj}^{t}| + w_z \sum_{j=1}^{q} |z_{vj}^{t+1} - z_{vj}^{t}| \right) \tag{6c}$$

For Eq. (6), $H_{t+1} \subseteq H_T$ and it represents the game history from initialization to the time step $t+1$. Regarding the defender's shuffling effectiveness, Eq. (6b) represents the status transition from time step t to $t+1$. In terms of IP hopping, port hopping and migration cost in a shuffle, the cost function in Eq. (6c) represents the cost of shuffling from time step t to $t+1$, where w_x, w_y, w_z are the weights for different shuffling techniques assigned by the network operator.

In contrast, the shuffling-based MTD intends to make it more difficult for attackers to launch a successful attack and to force them to quit due to increasing attack cost. Thus, we also quantify the payoff values of an attacker according to the game history and discounted value, and the objective function to minimize the attack payoff is as follows:

$$f_A = Minimize \sum_{t+1=1}^{T} \gamma^t \left(E^a(H_{t+1}) - C^a(H_{t+1})\right), where \tag{7a}$$

$$E^a(H_{t+1}) = \sum_{v \in V} STF((S_t(v), H_{t+1}) | S_t(v) = 0), and \tag{7b}$$

$$C^a(H_{t+1}) = \sum_{v \notin D_{t+1} \cap v \notin A_{t+1}} (w_x + w_y) + \sum_{v \in D_{t+1}} w_z \tag{7c}$$

As seen in Eq. (7b) and Eq. (7c), the attacker's effectiveness value E_{t+1}^a and cost value C_{t+1}^a respectively represent the reward obtained from the VM crash and the cost caused during the whole attack stages. In detail, the effectiveness function of the attacker indicates that the target VM has been crashed at time step $t+1$. Moreover, the attacker's cost value in Eq. (7c) can be divided into two parts: the cost spends during the scanning and probing stages and the cost occurs when implementing the attack to the target VM. Hence, it can be calculated by the number of VMs, ports and IPs, which follow the strategies of the defender and attacker, using the assigned weights w_x, w_y, w_z as well.

Different from the defender and the attacker directly involved in the game, common users should have got normal and stable access to the VM. However, the attacks lead to the target VM crashes and the defender may shuffle this VM, which takes the service down for a time. On the one hand, users need to endure the service instability caused by the attack. On the other hand, the shuffling defense also prevents users from accessing services within a short period of time. In addition, if the service delay exceeds the tolerance limit, some users who are impatient may choose to access another VM before the defender makes a shuffle decision. Therefore, we quantify the cost values of legal users that caused by the MTD shuffle, and the objective function to minimize the users' cost is as follows:

$$f_U = Minimize \sum_{t+1=1}^{T} C^u(H_{t+1}), where \tag{8a}$$

$$C^u(H_{t+1}) = \eta \left(\sum_{v \in A_{t+1}} \mathbb{t}_a + \sum_{v \in D_{t+1}} \mathbb{t}_d \right) + \sum_{v \in U_{t+1}} \mathbb{t}_u \tag{8b}$$

As discussed in Section 4.2, the effectiveness value is assigned to be 0 for users in the game, so we only calculate the cost value of users in Eq. (8b). The cost function includes the time when the service delay increases caused by the attack, the service interruption due to the shuffle, and the waiting time for users to turn to other VMs, where $\mathbb{t}_a$, $\mathbb{t}_d$ and $\mathbb{t}_u$ separately represent the average overhead induced by each of these actions. It is worth noting that only the time caused by attacks and defenses will affect all online users on the current VM, and the overhead due to the user's spontaneous behaviors will be borne by the users who have taken actions. In addition, in an actual scenario, not all users are online at the same time and unnecessary shuffling costs are generated during each time step. In order to decrease the extra costs, we denote the number of online users in one VM by η to guide the defender to evaluate users' overhead and make his decision in a more cost-effective manner, where $0 \leq \eta \leq m$.

In our problem, we consider all the effectiveness and cost values from all game players with different weights w_1, w_2, and w_3 assigned by the network operator. Our objective function finds the best trade-off as follows.

Objective:

$$Maximize \quad w_1 \times f_D - w_2 \times f_A - w_3 \times f_U \tag{9}$$

5.3. Decision problem modeling

Markov Decision Processes (MDPs) are a popular model for performance analysis and system optimization, which provide a framework for inferring from the actions of decision making agents in an environment. Although plenty of research has been made in the area, there are still many sequential decision problems that are not well modelled by MDPs. One important reason for this is that in most cases of real world, decision problems have multiple objectives cause human action selection is driven by multiple objectives at the same time. For example, for a computer network we may want to maximize performance while minimizing power consumption (Roijers and Whiteson, 2017). The field of multi-objective decision making addresses how to formalize and solve decision problems with multiple objectives.

In the following, we exploit Multi-Objective Markov Decision Processes (MOMDP) to model the decision problem, which is a multi-stage stochastic optimization problem with the objective of maximizing the defender's payoff, minimizing users' overhead and minimizing the attacker's payoff. A MOMDP for these three objectives in our case is a tuple $(S, X, \pi, p, R, \gamma)$, where:

- S represents a finite set of states, which can be defined the same as Definition 1, and let S_t be the state of the system at time step t.
- X represents a finite set of decisions, and let X_t be the random variable of the decision at time t.
- π denotes a policy function that maps each state to a decision with the probability $p[X_t = x | S_t = s]$, where $x \in X$, $s \in S$.
- p denotes a state transition function that maps a decision and related states to a probability, which can be calculated with Algorithm 1, and let $p_{ss'}^x = p[S_{t+1} = s' | S_t = s, X_t = x]$, where $x \in X$, s, $s' \in S$.

• R is a reward function that maps a state and a decision to a reward vector of l dimensions with a decision distribution of π, where l represents the number of objectives.

• γ is the discount factor, and Let $\gamma_k \in (0, 1]$ be the discount factor of the kth objective.

In our problem, a policy is a strategy that decides the probability distribution of the solutions based on the state of the protected system at time step t. Solving the problem means finding the policy π^* that maximizes the total expected discounted reward value. Although many approaches have been studied, the simplest and most general of them is value iteration, which works very well in practice for MDPs. The flexibility of value iteration (Ashok et al., 2017) allows for several improvements and adaptations, further increasing its performance and accelerating processing of very large MDPs. To find the optimal shuffling strategy, thus we solve the following value function based on value iteration.

$$V^{\pi}_{t+1,k}(s) = \sum_{x\in X} \pi[x|s]\left(R^{\pi}_{k}(s,x) + \gamma_k \sum_{s'\in S} p^{x}_{ss'} \mathbb{E}[V^{\pi}_{t,k}(s')]\right) \tag{10a}$$

$$V^{\pi}_{t+1}(s) = \max \sum_{k=1}^{l} w_k V^{\pi}_{t+1,k}(s) \tag{10b}$$

In the value function, $V^{\pi}_{t+1,k}(s)$ represents the expected cumulative reward for the kth objective according to the policy π and the state s after t time steps. Then, $V^{\pi}_{t+1}(s)$ sums up the values of all objectives with different weights, where $\sum_{k=1}^{l} w_k = 1$, and $l = 3$ for our problem. In addition, all game players' decisions follows the strategies described in Section 4.3, and especially, the defender's shuffling decisions must be constrained by Eqs. (5a) to (5e). Finally, once the best policy π^* has been obtained, we can deduce the transition probability of the state, and then calculate the reward values of three objectives by Eqs. (6a), (7a), and (8a).

5.4. Trilateral game cost-effective shuffling algorithm

Although easy to implement, value iteration can be slow to converge in some cases. To shuffle more efficiently and accelerate the solution, in the following, we first present a novel trilateral game cost-effective shuffling algorithm (TCS) to consider the cost and effectiveness of shuffling, with the three objectives of maximizing the payoff that the defender may obtain, minimizing users' overhead and minimizing the payoff which the attacker can get. Specifically, in the initial assignment step, q users, r network segments and u ports are randomly assigned to n online VMs in our shuffling scenario, whereas the tth shuffling step iteratively reduces the number of the crashed VMs. Afterwards, the system state at time step t represents the assignment of users, network segments, ports in the system and the condition of crashed VMs through state transition function (Algorithm 1).

Thereout, TCS (Algorithm 2) is proposed to significantly reduce the unnecessary cost and is executed after the initial assignment at each time step. In TCS, Line 1 has a holistic view of the crashed VMs based on the current system state. Then, Line 2 judges whether the current VM has online users and no shuffling decisions are given in Line 3 if there is no user. Moreover, if there exist online users, we randomly set the new network segment and ports for the next time step in Line 6 and a previous threshold has been set in Line 7. Line 8 indicates that we have to migrate this part of users to another secure VM if there are few online users, and we have to relocate them to several VMs in Line 11 when the target VMs have no enough capacity. On the contrary, we turn the network segment and ports to absolutely different ones rather than VM migration in Line 15 if the average number of online users exceeds the half of maximum capacity. Finally, shuffling decisions of all VMs for the next time step are returned in Line 19.

Algorithm 2 Trilateral game cost-effective shuffling algorithm (TCS).

Input:
The VM states at time step t, $\{S_t(v_1), S_t(v_2), \ldots S_t(v_n)\}$;
A binary $r \times n$-matrix X_t;
A binary $u \times n$-matrix Y_t;
A binary $q \times n$-matrix Z_t;
The number of online users in each VM at time step t, $\{\eta_t(v_1), \eta_t(v_2), \ldots \eta_t(v_n)\}$;
Output:
A binary $r \times n$-matrix X_{t+1};
A binary $u \times n$-matrix Y_{t+1};
A binary $q \times n$-matrix Z_{t+1};

1: **for** $S_t(v_i) = 1, 1 \leqslant i \leqslant n$ **do**
2: **if** $\eta_t(v_i) = 0$ **then**
3: Set $\sum_{j=1}^{r} x^{t+1}_{i,j} = 0$, $\sum_{j=1}^{u} y^{t+1}_{i,j} = 0$, $\sum_{j=1}^{q} z^{t+1}_{i,j} = 0$;
4: **else**
5: **if** $0 < \eta_t(v_i) \leqslant [\frac{m}{2}]$ **then**
6: Randomly set $x^{t+1}_{i,j}, y^{t+1}_{i,j}$;
7: **if** Exist $0 < \eta_t(v_{i'}) \leqslant [\frac{m}{2}]$ and $S_t(v_{i'}) = 0$ **then**
8: Set $\sum_{j=1}^{q} z^{t+1}_{i',j} = \eta_t(v_i)$;
9: **else**
10: **for** $S_t(v_{i'}) = 0, 1 \leqslant i' \leqslant n$ **do**
11: Set $\sum_{i'=1}^{n}\sum_{j=1}^{q} z^{t+1}_{i',j} = \eta_t(v_i)$;
12: **end for**
13: **end if**
14: **else**
15: Set $x^{t+1}_{i,j} \cap x^{t}_{i,j} = 0, y^{t+1}_{i,j} \cap y^{t}_{i,j} = 0, z^{t+1}_{i,j} = z^{t}_{i,j}$;
16: **end if**
17: **end if**
18: **end for**
19: **return** all $x^{t+1}_{i,j} \in X_{t+1}, y^{t+1}_{i,j} \in Y_{t+1}, z^{t+1}_{i,j} \in Z_{t+1}$

In addition, the transition probability of TCS $p^{TCS}_{ss'}$ is represented as the STF function in Algorithm 1, where the value is correlated to the system state and players' actions, especially (m, n, q, r, u) in this shuffling scenario. The rewarding values R^{TCS}_1, R^{TCS}_2, and R^{TCS}_3 of each state s with policy TCS represent the payoffs of each shuffle for different players in state s as follows:

$$R^{TCS}_1(s) = \sum_{s'\in S} p^{TCS}_{ss'} |s - s'| - TCS(s)(w_x r + w_y u + w_z q) \tag{11a}$$

$$R^{TCS}_2(s) = -(\sum_{s'\in S} p^{TCS}_{ss'} |s - s'| - n(1-s)(w_x + w_y) - s w_z) \tag{11b}$$

$$R^{TCS}_3(s) = -\sum_{s'\in S} p^{TCS}_{ss'} [\eta(\mathbb{t}_a + \mathbb{t}_d) + \mathbb{t}_u] \tag{11c}$$

where the rewarding functions of the attacker and user in Eqs. (11b) and (11c) are negative due to the objectives of minimization. Given the transition probability and the rewarding functions, therefore, the cumulation of respective rewards with the policy TCS after $t+1$ shuffles can be calculated as follows:

$$V^{TCS}_{t+1,k}(s) = R^{TCS}_k(s) + \gamma_k \sum_{s'\in S} p^{TCS}_{ss'} V^{TCS}_{t,k}(s') \tag{12}$$

where $V^{TCS}_{t+1,k}(s) = R^{TCS}_k(s)$ when $t = 0$, and γ_k represents a discount value where $\gamma_1 = \gamma_2 = \gamma$ (described in Eqs. (6a) and (7a)) and $\gamma_3 = 1$. Therefore, according to the proposed TCS algorithm

and above rewarding and value functions, we can then solve the optimal decision problem efficiently.

6. Evaluations and results

In this section, we evaluate and analyze the effectiveness and cost of the proposed TCS algorithm against DDoS attacks in simulation and experiment. First, we describe the simulation settings and compare our TCS algorithm with other existing shuffling algorithms. Then we introduce the experimental settings and implementation of the shuffling scenario in full. Finally, we measure the cost and effectiveness of our proposed TCS algorithm in our shuffling scenario with comparisons to some specific strategies.

6.1. Simulation

Software-defined network (SDN) (Yan et al., 2015) can provide flexible infrastructure for developing and deploying MTD mechanisms efficiently by decoupling the control plane and the data plane in the network. In SDN, the network controller monitors and controls the entire network from a central vantage point, and different functions of switches distributed in the network can be defined accurately and synchronously. To investigate the effectiveness of our proposed TCS algorithm and compare it with other methods, such as RRT (Renewal Reward Theory) (Wang et al., 2016), CSA (Cost-effective Shuffling Algorithm) (Lin et al., 2017), and CES (Cost-Effective Shuffling Method) (Zhou et al., 2019), we simulate a network environment by Mininet, which can create multiple instances of OpenFlow switches and hosts as a virtual SDN testbed. We also implement all these algorithms on an OpenFlow controller called OpenDayLight (OpenDayLight, 2020) that acts as the central authority to manage the network flexibly. The simulations are all performed on a 3.6 GHz Intel Core CPU with 32GB of RAM running 64-bit Ubuntu 16.04 operating system.

First, to find out the whole system state transition probability, we independently execute Algorithm 1 10,000 times with pre-defined parameters (m, n, q, r, u), where $m = 20, n = 50, q = 1000, r = 20, u = 100$. Afterward, we compare the expected value functions of TCS with that of RRT, CSA and CES in terms of different players' payoffs. More specifically, the sum cost of three kinds of defense mechanisms is set to 1, where the weights for IP hopping, port hopping and migration shuffling are separately set to 0.2, 0.1 and 0.7. Similarly, the weights for the defender payoff, attacker payoff, and user cost are set to 0.6, 0.3 and 0.1 here. In simulation, the reward values of successfully defending against an attack or attacking to cause the VM to crash are both calculated by 1, and the discount value γ is set to 0.9. In addition, to calculate the user cost, the average overhead $\mathbb{t}_d$, $\mathbb{t}_a$, and $\mathbb{t}_u$ induced by each player's action are set to 1.5, 2, and 1 for simplicity. Note that RRT is indifferent to the online users of the VMs, and CSA randomly selects half of the users to migrate in a single shuffle. Although CES pays attention to the number of online users, it does not calculate user payoffs separately but merges into the defensive payoffs together.

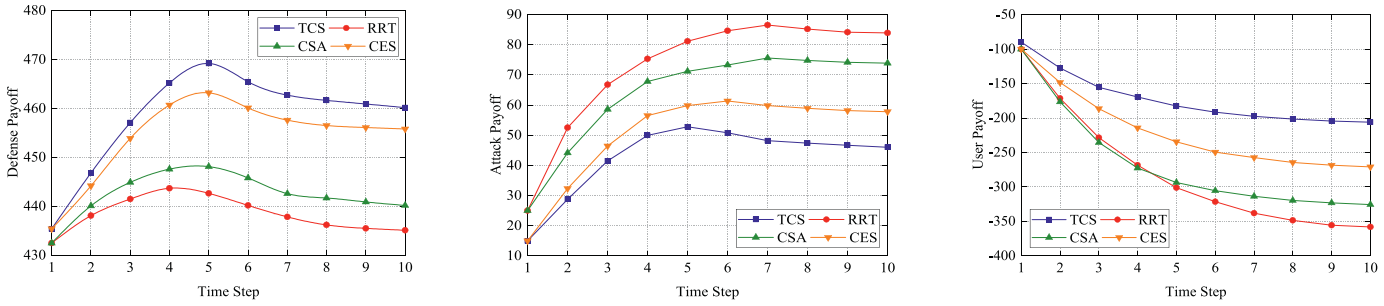
For a more comprehensive comparison among these algorithms, the pseudo codes from RRT, CSA and CES are utilized to make MTD shuffles in our simulation, and all payoff values are calculated by functions in this paper. Meanwhile, in order to make a fair comparison, we only calculate the VM migration cost for RRT and CSA algorithms which lack the other two MTD mechanisms. Specially, there is not a third-party participant for the shuffling scenarios of RRT, CSA, and CES, which means common users would not take actions initiatively and their overhead would be only affected by the defender and attacker. Therefore, user strategies in this paper cannot apply to them, and $\mathbb{t}_u$ should be set to 0 for these three algorithms.

Figs. 1 and 2 first compare the these algorithms with different time step and different number of average online users in one VM, respectively. In Fig. 1, 10 online users are initially involved within each VM in the overall shuffling scheme, and the system is allowed to allocate at most 50 VMs for shuffling. In Fig. 2, there are 0 to 20 online users within one VM at time step 10, when the cumulative values of all the players' payoffs have levelled off. Fig. 1 demonstrates that the TCS approach performs better when the time step increases whereas Fig. 2 manifests that the difficulty of limiting the attacker's payoff has increased by MTD shuffling approach when there are more online users accessing the protected system. Figs. 1(a) and 2(a) present the theoretical defense payoffs of shuffling, whereas Figs. 1(b) and 2(b) show the payoffs that the attacker can obtain in the game. In addition, Figs. 1(c) and 2(c) demonstrate the payoffs of common users that passively participate in the trilateral game, where the negative values indicate that users are always affected by service performance when the MTD shuffle happens.

For the cumulative defense payoff, Fig. 1(a) indicates that more payoff is gained when the time step increases in the beginning and it declines after 4 or 5 time steps. In detail, it takes costs for the defender to make every MTD shuffle, however, it could not lead to more effectiveness when the system has reached a stable state. As seen in Fig. 2(a), it is more difficult to relocate more users, thus there is no linear increase in the defense payoff with the growth of online users. Meanwhile, since each algorithm gives priority to the payoff of defender, there are similar results on the defense payoffs when different algorithms are deployed. Nevertheless, the proposed TCS algorithm still outperforms other methods in terms of defense payoff for the given time step. For the payoff of the attacker, Fig. 1(b) indicates that attack payoff increases in the beginning of the game but soon decays when the shuffling mechanisms start to work. On the contrary, it can be seen from Fig. 2(b) that the effect of the attack will be magnified as more users demand the service, thus the attack payoff almost linearly increases when more users get access to the VMs. Even so, our method still strives to protect the system, thereby effectively limiting the attack payoffs when compared to other algorithms. As shown in Fig. 1(c), the user payoff declines as the game progresses, and will tend to a stable value at the time horizon. When more users accessing to the system, Fig. 2(c) indicates that more users will suffer from the other two players' actions and their payoffs decline almost linearly. However, taking the impact of the shuffling on users into account and treating users as a member of the trilateral game, our proposed algorithm TCS outperforms RRT, CSA and CES in terms of the user payoff.

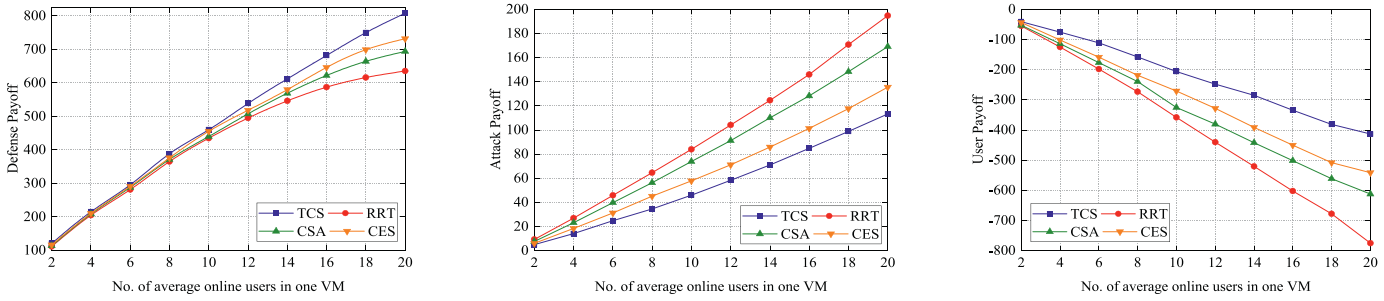
The performance of shuffling mechanisms in TCS outperforms that in RRT, CSA and CES due to three reasons. First, the state transition probability of TCS fully takes the correlation between states into consideration, while there is no detailed explanations of transition probability in RRT and CSA, and CES only focuses on the state of security resources. Second, TCS can utilize three kinds of defense mechanism, whereas RRT and CSA can only utilize one, and CES pays no attention to user payoff over the MTD shuffling. In detail, the underlying reason is because only utilizing migrations might introduce more cost if the system state is not so bad, while TCS is capable of determining the shuffling mechanism based on the game history. Finally, other methods do not fully consider the impact of the MTD defense mechanisms on users, whereas TCS builds a trilateral game which involves users as special participants, and takes into account the users' overhead of shuffling to make the effectiveness of defense and user overhead optimal.

Since RRT and CSA were proposed by other researchers, there might be some limitations in this comparison, for example, parameter setting of the algorithm in this scenario and many other experimental factors are all unknown for the existing techniques.



(a) Defense Payoff (b) Attack Payoff (c) User Payoff

Fig. 1. Comparison of TCS, RRT, CSA and CES in different players' payoffs at different time steps.



(a) Defense Payoff (b) Attack Payoff (c) User Payoff

Fig. 2. Comparison of TCS, RRT, CSA and CES in different players' payoffs with different number of average online users in one VM at time step 10.

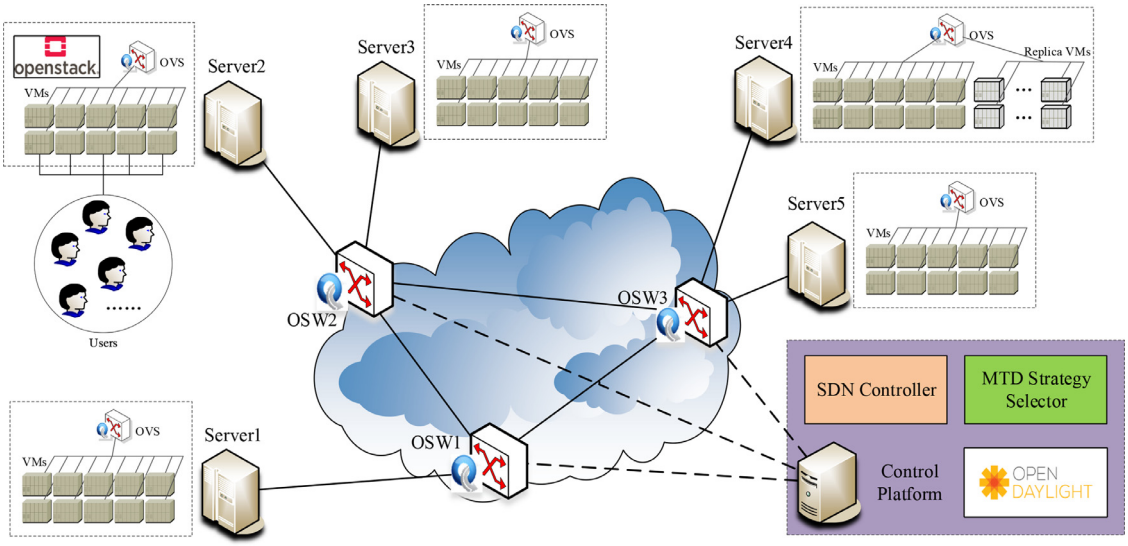


Fig. 3. Implementation of the shuffling scenario in an experimental SDN network.

However, we have tried to make as fair a comparison as possible, and according to the comparison results indicated in Figs. 1 and 2, our proposed TCS method still provides a powerful competitive advantage in the cost-effective MTD shuffling.

6.2. Experimental settings

We implement the shuffling scenario in an experimental SDN testbed, which is shown in Fig. 3. The testbed that we use for experimental analysis is composed of 5 Dell PowerEdge R720 servers and a Dell PowerEdge R430 server. Each Dell PowerEdge R720 has 32 GB of RAM, 4 TB hard disk storage and 12 core CPU. Dell PowerEdge R430 has 16 GB RAM, 1 TB disk storage and 4 core CPU.

One single server is employed to construct the control platform, using OpenDayLight (ODL) based SDN controller and PHP Laravel web framework as front-end. ODL is an open source SDN controller for shuffling rule installations and we deploy our algorithm as an MTD strategy selector module on it. Meanwhile, for the virtual network deployment, we utilize OpenStack (OpenStack, 2020) for computing and network resource provisioning on the other five servers. OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources, it provides a virtual layer on physical servers which decouples hardware from the workload. In addition, the VMs are managed and controlled by SDN controller via Open vSwitch (OVS) (OpenvSwitch, 2020). OVS is heavily used in cloud computing frameworks, and is designed to

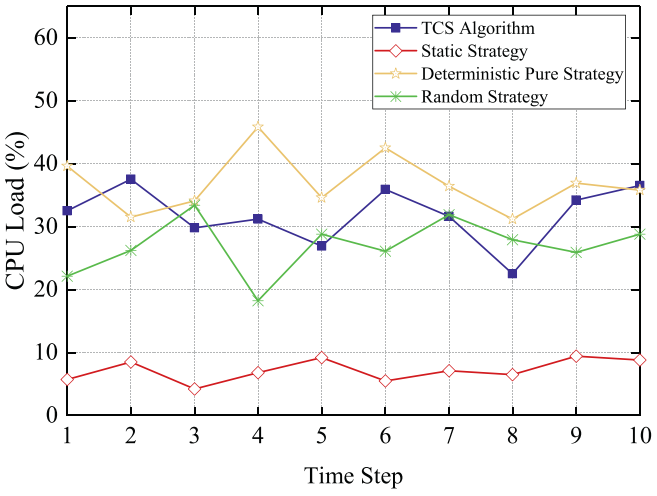


Fig. 4. CPU load of SDN controller under different shuffling strategies.

enable massive network automation through programmatic extension, while still supporting standard management interfaces and protocols.

In the implementation, we first create 50 VMs which are equally allocated to five servers, and each VM is assigned for at most 20 users with equal CPU and memory. In addition, the 50 VMs are organized with different IP and ports, the attacker can overload the VMs through DDoS attack tools and the defender can only create a replica for each VM under certain circumstances.

6.3. Results

First, we implement our TCS algorithm and execute the program as an application on the control platform. Then, we compare our trilateral game cost-effective shuffling method with other strategy selection methods, including static strategy, deterministic pure strategy, and random strategy in terms of overhead and performance against DDoS attacks. Finally, detailed experimental results are followed in Section 6.3.1–6.3.3.

6.3.1. Overhead of SDN controller's CPU load

In order to evaluate the processing overhead on the SDN controller consumed by the shuffling strategies, we run these strategies as different modules and evaluate the influence on the SDN controller, which is shown in Fig. 4. A static strategy, which does not modify any network resource over time, makes the CPU load never exceed 10%. Deterministic pure strategy selects a pure strategy that shuffles all optional VMs and their configurations, which inevitably results in a higher CPU load of SDN controller ranging from 31.2% to 45.8%. While a random strategy does not consider all the network resources, but randomly selects a strategy to shuffle part of them at each time step. Compared to the deterministic pure strategy, the CPU load is about 18.2%–28.8% when under a random strategy, and about 22.5%–37.5% under the proposed TCS algorithm.

Obviously, the SDN controller under static strategy has the least CPU load but with the highest risk. In order to keep the protected system from attacks, the deterministic pure strategy chooses to shuffle all resources at each time step and thus introduces extra CPU load inevitably, whereas the random strategy shuffles some of the protected VMs in the testbed randomly. Although it takes less load, the effectiveness of the shuffling could not be guaranteed all the time. However, our approach can find a good balance between the performance and overhead of the MTD mechanisms according to the solution to the objective function. The processing overhead on the SDN controller is in an acceptable level when the TCS algorithm has been deployed, which is between that of the random strategy and deterministic pure strategy.

6.3.2. Evaluation of QoS under DDoS attacks

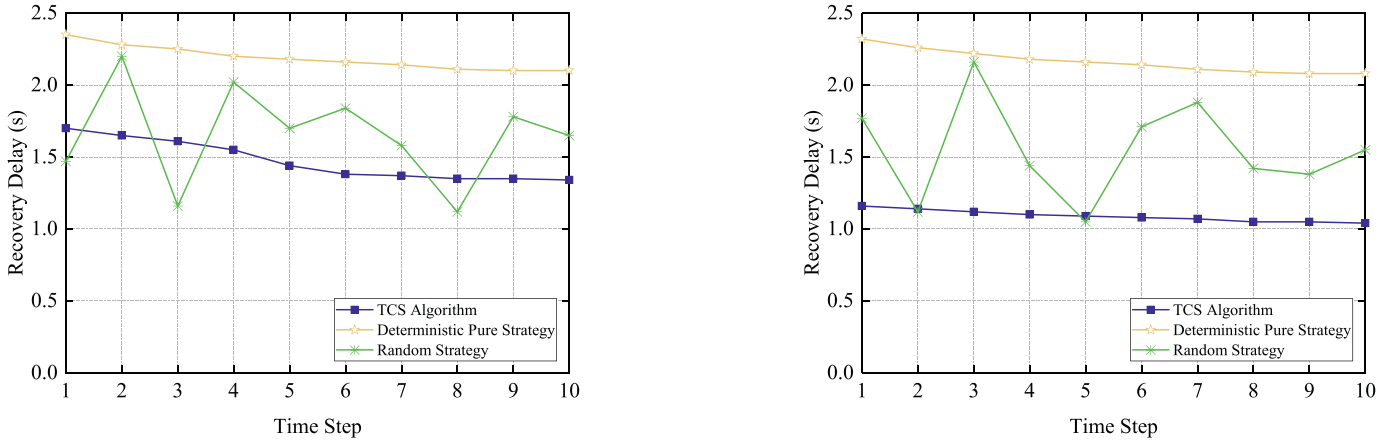
Quality of Service (QoS) is commonly used to describe non-functional attributes of a service with typical metrics, including response time, throughput, availability, reliability, etc. (Zhu et al., 2017). Ideally, QoS values can be directly specified in Service-Level Agreements (SLAs) by service providers. However, due to temporal and spatial variation of network service, user-perspective QoS data may reflect more individual characteristics (Luo et al., 2019). In terms of shuffling-based MTD, obtaining QoS information is still a challenging issue due to the dynamic and stochastic nature of MTD mechanisms. This dynamic nature is due to the frequent changes that may occur in MTD scenarios, whereas stochastic nature is due to the unpredictable occurrence of these changes. Taking the relationship between users and services into consideration, in this paper, we mainly take the average time of every online user waiting for the service recovery as the QoS measure in an MTD shuffling scenario. Service recovery delay is defined as the time interval between the moment when the service is unavailable and when a user can receive the response for service request again. The average service recovery delay $\bar{d}$ for time step t is formulated by the following quantitative measure:

$$\bar{d}_t = \frac{\sum_{i=1}^{n} \sum_{j=1}^{\eta_t(v_i)} d_t(i,j)}{\sum_{i=1}^{n} \eta_t(v_i)} \tag{13}$$

where $d_t(i, j)$ is the delay of the jth user waiting for the ith VM recovery, and $\eta_t(v_i)$ represents the number of users served by v_i at time step t.

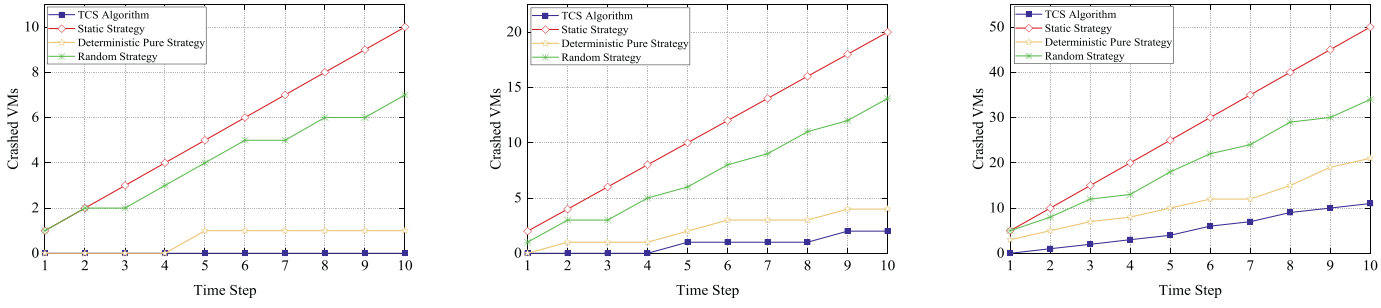
We claim that using average values is reasonable, because, first, several MTD mechanisms are placed among VMs in this paper, which means online users served by different VMs may have significantly different values of recovery delay. Second, if comparisons were to be made based on every user's value, the results would be biased towards one of the methods in some cases, which is not reasonable. Therefore, in order to evaluate the QoS under DDoS attacks, we construct a typical DDoS attack tool using hping3 (RGhanti and GM Naik, 2015), and carry out two kinds of DDoS attacks including SYN (synchronize) flood and UDP (User Datagram Protocol) flood, where the rate of flood traffic equals 200 Mbps on one VM. The experimental results of resisting these typical DDoS attacks with different strategies are shown in Fig. 5. Meanwhile, it is worth noticing that the result of static strategy (represented as red line in Figs. 4, 6, etc.) is not included in the comparison. Different from other strategies in this part of experiment, the failure to adopt the MTD mechanism will cause a large part of VMs to crash, resulting in a sharp increase in recovery delay and even complete communication interruption when the service is unreachable for legitimate users. Even if the service responds to the user's request after a long delay, it cannot be concluded that the service recovers from the attacks, and the average waiting time of static defense cannot be accurately calculated in the experiment. Therefore, to make a fair comparison of other strategies in terms of service recovery delay, we have to exclude the static defense out of Fig. 5.

In general, the results in Fig. 5(a) indicate that our approach in total requires only 1.34–1.70 s for every online user during SYN flood, which is an acceptable time for users to wait during the restart of services. Meanwhile, as the time step increases, there is a slight decrease on the time consumption of shuffling procedure. Then, we compare the QoS value of TCS with the deterministic pure strategy and random strategy. As seen in Fig. 5(a), the



(a) under SYN flood (b) under UDP flood

Fig. 5. Users' average time overhead of waiting for service recovery with different strategies.



(a) Target on 1 VM at each time step (b) Target on 2 VMs at each time step (c) Target on 5 VMs at each time step

Fig. 6. Numbers of crashed VMs under SYN flood with different shuffling strategies.

deterministic pure strategy consumes the most waiting time that exceeds 2 s for every user, because it mobilizes all MTD mechanisms and performs a full shuffling at each time step. This may cause the system service to be unavailable to some degree, thereby increasing the delay and damaging the users' payoffs. In addition, the time consumed by the random strategy fluctuates from 1.12 s to 2.2 s due to its random scheme during the shuffling process.

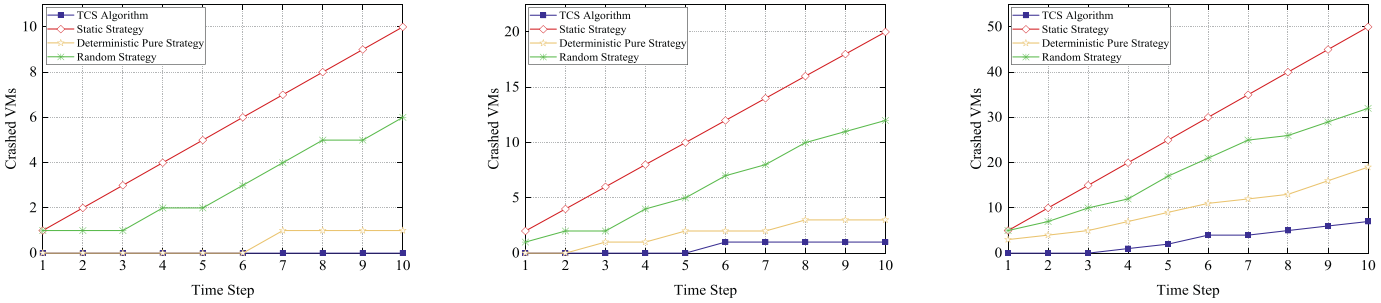
Similarly, as seen in Fig. 5(b), the recovery delay is still more than 2 s for the deterministic pure strategy during the UDP flood attacks, and there is also a random distribution of a legitimate user's waiting time with the random strategy. Different from SYN flood attacks attempting to consume enough system resources to make it unresponsive, UDP flood usually sends a large number of UDP packets to random ports of the target system, forcing it to send many ICMP (Internet Control Message Protocol) packets, and eventually leads it to be unreachable. Despite all this, the results of deterministic pure strategy and random strategy during UDP flood are almost the same as those under SYN flood attacks, because all of the strategies generated by these methods have no significant difference between SYN flood and UDP flood. However, the proposed TCS algorithm will focus more on the use of port and IP hopping to resist attacks when developing strategies, rather than unnecessary VM migration when facing such attacks. Owing to this, the average recovery delay with TCS during the UDP flood ranges from 1.04 s to 1.16 s, which is obviously lower than the other two strategies and guarantees the QoS of the protected system.

In summary, our approach reduces the service recovery delay of nearly one second compared to the deterministic pure strategy, while in most cases it is less than the recovery time spent by the random strategy, which is attributed to our method that takes full account of user behaviors during the shuffling process and thereby achieving the highest level of QoS values.

6.3.3. Performance of resisting DDoS attacks

Finally, to evaluate the capability of our proposed method to resist DDoS attacks, we reuse the experimental configurations in Section 6.3.2 to send SYN flood and UDP flood on the protected VMs with random source IP addresses. In addition, to implement a more detailed comparison, we set different attack intensity to simulate different attackers in this part of experiments, where the maximum attack rate can reach 1Gbps when the attacker floods 5 VMs at one time, and experimental results under several defense strategies can be seen from Fig. 6 to Fig. 9.

It is obvious that the proposed TCS algorithm has a better performance than other strategies in the ability against DDoS attacks. As shown in Fig. 6, no matter how many VMs attackers select to conduct attacks, the number of crashed VMs increases linearly as static strategy has no ability to resist SYN flood attacks. Meanwhile, the random strategy causes a surge in the number of VMs that crash due to lack of analysis on the system state when the trilateral game happens. Once the configuration after single shuffle still can be reached by the attacker, the MTD shuffle will be invalid with costs. Although the deterministic pure strategy can resist the attack to a certain extent, it is still weaker than the performance of TCS especially when the attacker's ability has increased. In addition, the deterministic pure strategy may be predicted by the attacker, which makes the defender passive in the beginning. In contrast, results indicate that TCS can effectively keep the protected system safe when the attacker only floods a VM at each time step,



(a) Target on 1 VM at each time step (b) Target on 2 VMs at each time step (c) Target on 5 VMs at each time step

Fig. 7. Numbers of crashed VMs under UDP flood with different shuffling strategies.

and restrict the number of crashed VMs to around 10 even when the attacker strikes more VMs.

Fig. 7 indicates the similar results of the crashed VMs during the UDP flood attacks. It can be seen that the static strategy also cannot afford to resist UDP flood attacks, and the random strategy still causes a large number of VMs to crash when the game ends although it is slightly better than static defense. Similarly, the deterministic pure strategy is able to defend against UDP flood attacks with low attack intensity, and there is a sharp increase in the number of crashed VMs when the attacker floods more VMs at a time. Although it is not easy to deal with high-intensity attacks, the proposed method can effectively guard most VMs of the protected system regardless of the attack intensity. Furthermore, when comparing Figs. 6 and 7, it can be indicated that although the overall trend for the number of the crashed VMs is the same, the impact of the SYN flood is obviously greater than the UDP flood. On one hand, since the size of SYN packets is much smaller than that of UDP packets, the SYN flood attacker would send more packets even if the rate of attack traffic is the same as UDP flood. On the other hand, SYN flood attacks work by consuming enough target system resources whereas UDP flood attacks target network link bandwidth, thereby leading more VMs to crash during SYN flood attacks. Nevertheless, it can be concluded that TCS is more effective in preventing protected systems from DDoS attacks, and better at restricting the number of crashed VMs within the limited time steps.

To gain a deeper insight into the understanding of the behavior and performance of different strategies in resisting DDoS attacks, we use the following key metrics to evaluate the defense, including packet loss rate and RTT (Round-Trip Time). In detail, packet loss rate is defined as the rate of lost packets or bytes due to the interaction of the legitimate traffic with the attack, or due to collateral damage from defense. RTT is defined as the interval between when a request is issued and when a complete response is received from the destination, which gives a measure of latency. Once the DDoS attacks last for a while, the queue will be full and the subsequent packets will be dropped, which causes a substantial increase in the packet loss rate and RTT. In particular, we report results of these key performance measures when sending DDoS traffic at different rates. Fig. 8 shows the performance degradation in terms of packet loss measured at different rates of attack traffic ranging from 0 to 200 Mbps. It demonstrates the patterns of packet loss rate in the cases of SYN flood and UDP flood attacks. As it can be seen from Fig. 8(a), when the rate of attack traffic is less than 170 Mbps, packet loss of TCS algorithm is always zero. When the rate of attack traffic becomes higher than 170 Mbps, or even a little, the loss rate is changed to about 1% and reaches about 12% when the SYN flood rate equals 200 Mbps. However, the packet loss rates of the static strategy and random strategy have a huge increase even at a low SYN flood rate, and are separately changed to about 93% and 83% at 200 Mbps of attack rate. Even though the performance of the deterministic pure strategy is very close to that of TCS when the rate of SYN flood is less than 120 Mbps, its highest packet loss is still about 50%.

Similarly, Fig. 8(b) indicates that the packet loss of static strategy under UDP flood also begins to increase when the rate of attack traffic becomes higher than 10 Mbps, and reaches the maximum value of 88% at 200 Mbps of attack rate. For the random strategy and deterministic pure strategy, the highest packet loss rates are 74% and 29%, respectively. Notably, the proposed TCS algorithm outperforms all other methods during UDP flood in terms of packet loss rate, which is effectively limited to 10% even under the maximum attack intensity.

In addition, Fig. 9 exhibits the degree of RTT increase of different defense strategies when sending DDoS attacks traffic to the target server at different rates. In terms of the overall trend, whatever the kind of DDoS attacks is, the average RTT of all methods shows a gradual upward trend as the rate of attack traffic increases. As the Fig. 9(a) shows, in the case of static defense, the average RTT increases when the rate of attack traffic ranges from 10 Mbps to 110 Mbps. When the rate of attack traffic is higher than 110 Mbps, the RTT of static defense becomes constant because 200 ms is the timeout value that we set throughout the experiments. An interesting observation is that random strategy usually has a higher value of average RTT since it randomly selects one of the MTD mechanisms to shuffle the target VM at every time step which may not overlap with the attacker's reconnaissance phase. However, this is not the case for the deterministic pure strategy. It protects the system service by all shuffling mechanisms regardless of highly suspicious activities, which defends against the SYN flood attacks but makes RTT affected by unnecessary shuffle. Another important observation in Fig. 9(a) is that the average RTT of TCS algorithm is always at a low level due to its cost-effective characteristic, and only increases to almost 20 ms when the rate of attack traffic equals 200 Mbps.

Finally, we also consider another kind of DDoS attack called UDP flood and represent the results in Fig. 9(b). In this graph, we can see similar results which are basically due to the defender's strategies. Although the overall trend is similar to that of Fig. 9(a), the average RTT at the same attack rate is generally lower when defending against UDP flood. At the same time, the average RTT of the static strategy dramatically increases from 10 Mbps of the attack rate, however, its highest value of RTT equals 193 ms, which does not exceed the preset timeout value. Unlike the static strategy, the random strategy has lower RTT, and floats around 100 ms even the attack rate becomes higher than 150 Mbps. Even though the gap between the deterministic pure strategy and TCS decreases under UDP flood attacks, the average RTT is always less than 20 ms for the TCS case. Hence, we claim that it is reasonable to use the TCS algorithm considering the overall advantages.

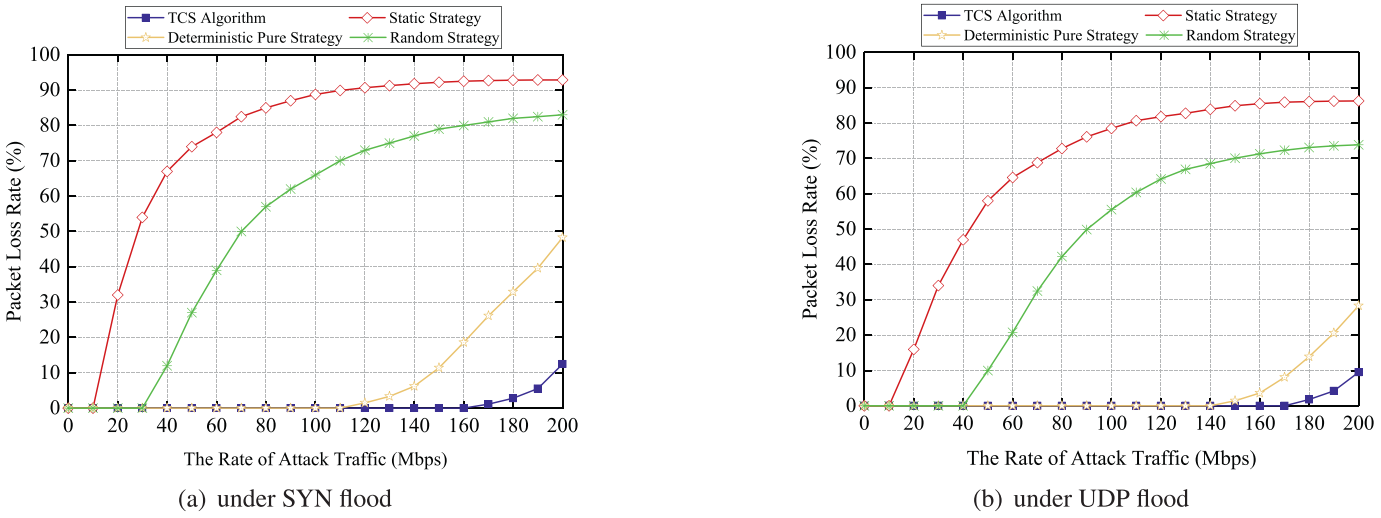


Fig. 8. Lost packets percentage during DDoS attacks with different strategies.

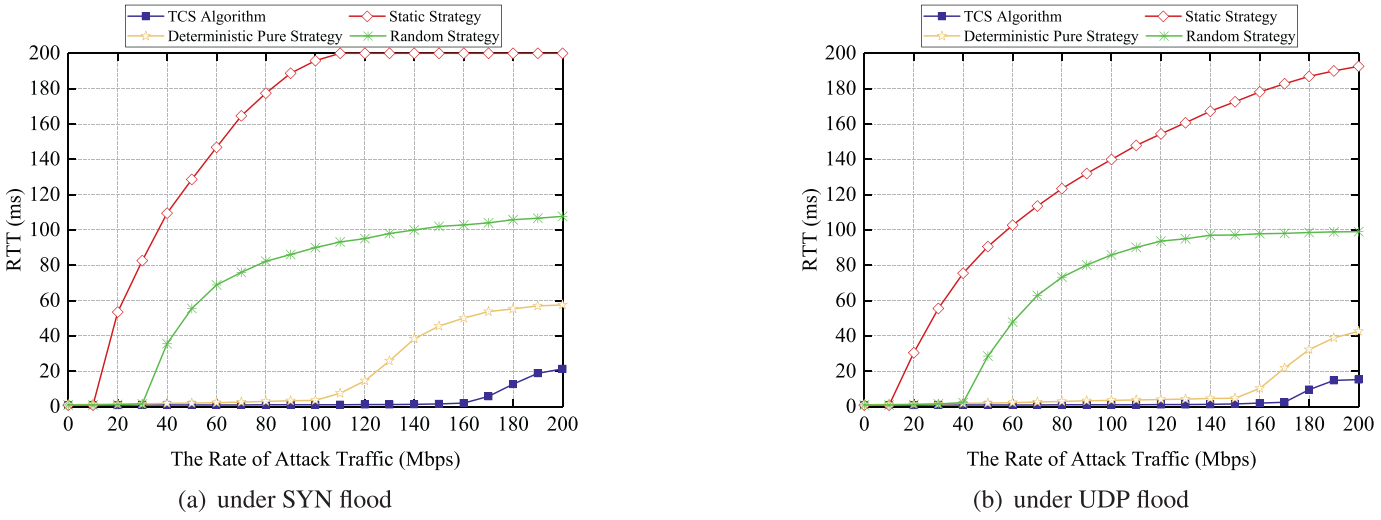


Fig. 9. Average RTT during DDoS attacks with different strategies.

7. Conclusions

MTD has recently emerged as one of the potentially game-changing themes in cyber security. Although the attacker under traditional static defense has an innate advantage, MTD holds promise to change the game in favor of the defender. Thus, MTD has received significant attention to solve cyber security problems such as mitigating DDoS attacks. Unfortunately, most such techniques has not adequately demonstrated the rationality of strategies, and in some cases has not considered defense costs when evaluating the utility of MTD strategies. The problem of balancing the effectiveness and cost associated with the deployment of MTD techniques has not received sufficient attention, and cost-effective MTD mechanisms are still lacking.

Our preliminary work (Zhou et al., 2019) provided a first important step toward addressing some of these limitations. This paper significantly extended that work by introducing a novel trilateral game theory that can include users' overhead at the same time. We also presented heuristic game strategies to characterize the behavior of attackers, defenders and users, and model the interaction among them with a sequential game. Then, Multi-Objective Markov Decision Processes are utilized to capture the effects on the overhead and performance of MTD. Finally, the TCS algorithm was proposed to seek the best trade-off between cost and effectiveness in the shuffling scenario.

The cost-effectiveness of our approach was evaluated in simulation and outperformed other existing algorithms, such as RRT, CSA and CES. In addition, TCS was deployed on an SDN based shuffling testbed and evaluated in terms of overhead and performance. The comparison with other strategies showed several key advantages of the proposed algorithm. First, the lower required CPU and recovery delay ensured the feasibility of the method and guaranteed the quality of service. Second, it is evident that the deployment of TCS algorithm was beneficial for improving overall system security and for protecting the system against DDoS attacks effectively.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Yuyang Zhou: Conceptualization, Methodology, Writing - original draft. **Guang Cheng:** Supervision, Resources, Project adminis-

tration. **Shanqing Jiang:** Software, Investigation. **Yuyu Zhao:** Formal analysis, Visualization. **Zihan Chen:** Writing - review & editing.

Acknowledgment

This work is supported by National Key Research and Development Program of China under Grant No. 2018YFB1800602 and No. 2017YFB0801703, CERNET Innovation Project (NGIICS20190101, NGII20170406), and Ministry of Education-China Mobile Research Fund Project (MCM20180506).

References

Alavizadeh, H., Jang-Jaccard, J., Kim, D.S., 2018. Evaluation for combination of shuffle and diversity on moving target defense strategy for cloud computing. In: 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). IEEE, pp. 573–578. doi:10.1109/TrustCom/BigDataSE.2018.00087.

Alavizadeh, H., Kim, D.S., Jang-Jaccard, J., 2019. Model-based evaluation of combinations of shuffle and diversity MTD techniques on the cloud. Future Gener. Comput. Syst. doi:10.1016/j.future.2019.10.009.

Albanese, M., Jajodia, S., Venkatesan, S., 2018. Defending from stealthy botnets using moving target defenses. IEEE Secur. Privacy 16 (1), 92–97. doi:10.1109/MSP.2018.1331034.

Ashok, P., Chatterjee, K., Daca, P., Kretínský, J., Meggendorfer, T., 2017. Value iteration for long-run average reward in Markov decision processes. In: Computer Aided Verification. Springer International Publishing, pp. 201–221. doi:10.1007/978-3-319-63387-9_10.

Aydeger, A., Saputro, N., Akkaya, K., 2019. A moving target defense and network forensics framework for ISP networks using SDN and NFV. Future Gener. Comput. Syst. 94, 496–509. doi:10.1016/j.future.2018.11.045.

Bardas, A.G., Sundaramurthy, S.C., Ou, X., DeLoach, S.A., 2017. MTD CBITS: moving target defense for cloud-based it systems. In: European Symposium on Research in Computer Security. Springer, pp. 167–186. doi:10.1007/978-3-319-66402-6_11.

Blakely, B., Horsthemke, W., Poczatec, A., Nowak, L., Evans, N., 2019. Moving Target, Deception, and Other Adaptive Defenses. Springer International Publishing, Cham, pp. 95–118.

Bopche, G.S., Mehtre, B.M., 2017. Graph similarity metrics for assessing temporal changes in attack surface of dynamic networks. Comput. Secur. 64, 16–43. doi:10.1016/j.cose.2016.09.010.

Cai, G.-l., Wang, B.-s., Hu, W., Wang, T.-z., 2016. Moving target defense: state of the art and characteristics. Front. Inf. Technol. Electron.Eng. 17 (11), 1122–1153. doi:10.1631/FITEE.1601321.

Carvalho, M., Ford, R., 2014. Moving-target defenses for computer networks. IEEE Secur. Privacy 12 (2), 73–76. doi:10.1109/MSP.2014.30.

Chang, S.-Y., Park, Y., Babu, B.B.A., 2018. Fast IP hopping randomization to secure hop-by-hop access in SDN. IEEE Trans. Netw. Serv. Manage. 16 (1), 308–320. doi:10.1109/TNSM.2018.2889842.

Connell, W., Menasce, D.A., Albanese, M., 2018. Performance modeling of moving target defenses with reconfiguration limits. IEEE Trans. Dependable Secure Comput. doi:10.1109/TDSC.2018.2882825.

Crouse, M., Prosser, B., Fulp, E.W., 2015. Probabilistic performance analysis of moving target and deception reconnaissance defenses. In: Proceedings of the Second ACM Workshop on Moving Target Defense. ACM, pp. 21–29. doi:10.1145/2808475.2808480.

Cybenko, G., Wellman, M., Liu, P., Zhu, M., 2019. Overview of Control and Game Theory in Adaptive Cyber Defenses. Springer International Publishing, Cham, pp. 1–11.

Feng, X., Zheng, Z., Cansever, D., Swami, A., Mohapatra, P., 2017. A signaling game model for moving target defense. In: IEEE INFOCOM 2017-IEEE Conference on Computer Communications. IEEE, pp. 1–9. doi:10.1109/INFOCOM.2017.8057200.

Gillani, F., Al-Shaer, E., Lo, S., Duan, Q., Ammar, M., Zegura, E., 2015. Agile virtualized infrastructure to proactively defend against cyber attacks. In: 2015 IEEE Conference on Computer Communications (INFOCOM). IEEE, pp. 729–737. doi:10.1109/INFOCOM.2015.7218442.

Hahn, E.M., Hashemi, V., Hermanns, H., Lahijanian, M., Turrini, A., 2019. Interval Markov decision processes with multiple objectives: from robust strategies to Pareto curves. ACM Trans. Model. Comput. Simul. 29 (4). doi:10.1145/3309683.

Hong, J.B., Enoch, S.Y., Kim, D.S., Nhlabatsi, A., Fetais, N., Khan, K.M., 2018. Dynamic security metrics for measuring the effectiveness of moving target defense techniques. Comput. Secur. 79, 33–52. doi:10.1016/j.cose.2018.08.003.

Hong, J.B., Kim, D.S., 2015. Assessing the effectiveness of moving target defenses using security models. IEEE Trans. Dependable Secure Comput. 13 (2), 163–177. doi:10.1109/TDSC.2015.2443790.

Hu, Z., Zhu, M., Liu, P., 2017. Online algorithms for adaptive cyber defense on bayesian attack graphs. In: Proceedings of the 2017 Workshop on Moving Target Defense. ACM, New York, NY, USA, p. 99109. doi:10.1145/3140549.3140556.

Huang, Y., Ghosh, A.K., 2011. Introducing diversity and uncertainty to create moving attack surfaces for web services. In: Moving Target Defense. Springer, pp. 131–151. doi:10.1007/978-1-4614-0977-9_8.

Jia, Q., Wang, H., Fleck, D., Li, F., Stavrou, A., Powell, W., 2014. Catch me if you can: a cloud-enabled DDoS defense. In: 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. IEEE, pp. 264–275. doi:10.1109/DSN.2014.35.

Kampanakis, P., Perros, H., Beyene, T., 2014. SDN-based solutions for moving target defense network protection. In: Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014. IEEE, pp. 1–6. doi:10.1109/WoWMoM.2014.6918979.

Leeuwen, B.P.V., Stout, W.M.S., Urias, V., 2016. MTD assessment framework with cyber attack modeling. In: 2016 IEEE International Carnahan Conference on Security Technology (ICCST), pp. 1–8. doi:10.1109/CCST.2016.7815722.

Lei, C., Ma, D., qi Zhang, H., 2017. Optimal strategy selection for moving target defense based on Markov game. IEEE Access 5, 156–169. doi:10.1109/ACCESS.2016.2633983.

Lei, C., Zhang, H.-Q., Tan, J.-L., Zhang, Y.-C., Liu, X.-H., 2018. Moving target defense techniques: a survey. Secur. Commun. Netw. 2018. doi:10.1155/2018/3759626.

Lei, C., Zhang, H.-Q., Wan, L.-M., Liu, L., Ma, D.-h., 2018. Incomplete information Markov game theoretic approach to strategy generation for moving target defense. Comput. Commun. 116, 184–199. doi:10.1016/j.comcom.2017.12.001.

Lin, Y.-H., Kuo, J.-J., Yang, D.-N., Chen, W.-T., 2017. A cost-effective shuffling-based defense against http ddos attacks with sdn/nfv. In: 2017 IEEE International Conference on Communications (ICC), pp. 1–7. doi:10.1109/ICC.2017.7997190.

Liu, C.-C., Huang, B.-S., Tseng, C.-W., Yang, Y.-T., Chou, L.-D., 2018. Sdn/nfv-based moving target ddos defense mechanism. In: International Conference of Reliable Information and Communication Technology. Springer, pp. 548–556. doi:10.1007/978-3-319-99007-1_51.

Luo, X., Wu, H., Yuan, H., Zhou, M., 2019. Temporal pattern-aware QoS prediction via biased non-negative latent factorization of tensors. IEEE Trans. Cybern. doi:10.1109/TCYB.2019.2903736.

Manadhata, P., Wing, J.M., 2004. Measuring a System's Attack Surface. Technical Report. Carnegie-Mellon Univ Pittsburgh pa School of Computer Science.

Miehling, E., Rasouli, M., Teneketzis, D., 2015. Optimal defense policies for partially observable spreading processes on bayesian attack graphs. In: Proceedings of the Second ACM Workshop on Moving Target Defense. ACM, pp. 67–76. doi:10.1145/2808475.2808482.

OpenDayLight, 2020. Home - opendaylight.. https://www.opendaylight.org/, (accessed 20 april 2020)

OpenStack, 2020. Build the future of open infrastructure. https://www.openstack.org/.

OpenvSwitch, 2020. Open vswitch http://www.openvswitch.org/.

Prakash, A., Wellman, M.P., 2015. Empirical game-theoretic analysis for moving target defense. In: Proceedings of the Second ACM Workshop on Moving Target Defense. ACM, pp. 57–65. doi:10.1145/2808475.2808483.

RGhanti, S., GM Naik, G., 2015. Design of system on chip for generating SYN flood attack to test the performance of the security system. Int. J. Comput. Appl. 122 (7), 14–17.

Roijers, D.M., Whiteson, S., 2017. Multi-objective decision making. Synth. Lect. Artif. Intell.Mach. Learn. 11 (1), 1–129. doi:10.2200/S00765ED1V01Y201704AIM034.

Steinberger, J., Kuhnert, B., Dietz, C., Ball, L., Sperotto, A., Baier, H., Pras, A., Rodosek, G.D., 2018. DDoS defense using MTD and SDN. In: NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium, pp. 1–9.. doi:10.1109/NOMS.2018.8406221.

Sugrim, S., Venkatesan, S., Youzwak, J.A., Chiang, C.-Y.J., Chadha, R., Albanese, M., Cam, H., 2018. Measuring the effectiveness of network deception. In: 2018 IEEE International Conference on Intelligence and Security Informatics (ISI). IEEE, pp. 142–147. doi:10.1109/ISI.2018.8587326.

Tan, J., Lei, C., Zhang, H., Cheng, Y., 2019. Optimal strategy selection approach to moving target defense based on Markov robust game. Comput. Secur. 85, 63–76. doi:10.1016/j.cose.2019.04.013.

Venkatesan, S., Albanese, M., Amin, K., Jajodia, S., Wright, M., 2016. A moving target defense approach to mitigate DDoS attacks against proxy-based architectures. In: 2016 IEEE Conference on Communications and Network Security (CNS). IEEE, pp. 198–206. doi:10.1109/CNS.2016.7860486.

Wahab, O.A., Bentahar, J., Otrok, H., Mourad, A., 2019. Resource-aware detection and defense system against multi-type attacks in the cloud: repeated bayesian stackelberg game. IEEE Trans. Dependable Secure. Comput. doi:10.1109/TDSC.2019.2907946.

Wang, H., Jia, Q., Fleck, D., Powell, W., Li, F., Stavrou, A., 2014. A moving target DDoS defense mechanism. Comput. Commun. 46, 10–21. doi:10.1016/j.comcom.2014.03.009.

Wang, H., Li, F., Chen, S., 2016. Towards cost-effective moving target defense against DDoS and covert channel attacks. In: Proceedings of the 2016 ACM Workshop on Moving Target Defense. ACM, pp. 15–25. doi:10.1145/2995272.2995281.

Wang, K., Du, M., Maharjan, S., Sun, Y., 2017. Strategic honeypot game model for distributed denial of service attacks in the smart grid. IEEE Trans. Smart Grid 8 (5), 2474–2482. doi:10.1109/TSG.2017.2670144.

Wang, K., Yuan, L., Miyazaki, T., Chen, Y., Zhang, Y., 2018. Jamming and eavesdropping defense in green cyber–physical transportation systems using a stackelberg game. IEEE Trans. Ind. Inf. 14 (9), 4232–4242. doi:10.1109/TII.2018.2841033.

Wright, M., Venkatesan, S., Albanese, M., Wellman, M.P., 2016. Moving target defense against DDoS attacks: an empirical game-theoretic analysis. In: Proceedings of the 2016 ACM Workshop on Moving Target Defense. ACM, pp. 93–104. doi:10.1145/2995272.2995279.

Xiong, X.-L., Yang, L., Zhao, G.-S., 2019. Effectiveness evaluation model of moving target defense based on system attack surface. IEEE Access 7, 9998–10014. doi:10.1109/ACCESS.2019.2891613.

Yan, Q., Yu, F.R., Gong, Q., Li, J., 2015. Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: a survey, some research issues, and challenges. IEEE Commun. Surv. Tutor. 18 (1), 602–622. doi:10.1109/COMST.2015.2487361.

Zangeneh, V., Shajari, M., 2018. A cost-sensitive move selection strategy for moving target defense. Comput. Secur. 75, 72–91. doi:10.1016/j.cose.2017.12.013.

Zhang, H., Zheng, K., Wang, X., Luo, S., Wu, B., 2019. Efficient strategy selection for moving target defense under multiple attacks. IEEE Access doi:10.1109/ACCESS.2019.2918319.

Zhang, H.-q., Lei, C., Chang, D.-x., Yang, Y.-j., 2017. Network moving target defense technique based on collaborative mutation. Comput. Secur. 70, 51–71. doi:10.1016/j.cose.2017.05.007.

Zhou, Y., Cheng, G., Jiang, S., Hu, Y., Zhao, Y., Chen, Z., 2019. A cost-effective shuffling method against DDoS attacks using moving target defense. In: Proceedings of the 6th ACM Workshop on Moving Target Defense, pp. 57–66. doi:10.1145/3338468.3356824.

Zhu, J., He, P., Zheng, Z., Lyu, M.R., 2017. Online QoS prediction for runtime service adaptation via adaptive matrix factorization. IEEE Trans. Parallel Distrib. Syst. 28 (10), 2911–2924. doi:10.1109/TPDS.2017.2700796.

Zhuang, R., DeLoach, S.A., Ou, X., 2014. Towards a theory of moving target defense. In: Proceedings of the First ACM Workshop on Moving Target Defense. ACM, pp. 31–40. doi:10.1145/2663474.2663479.

Yuyang Zhou is currently pursuing the Ph.D. degree in the School of Cyber Science and Engineering, Southeast University. His research interests include cyber security, moving target defense, and traffic classification.

Guang Cheng received the B.S. degree in Traffic Engineering from Southeast University in 1994, the M.S. degree in Computer Application from Heifei University of Technology in 2000, and the Ph.D degree in Computer Network from Southeast University in 2003. He is a full professor in the School of Cyber Science and Engineering, Southeast University, Nanjing, China. He is a senior member of the IEEE. His research interests include network security, network measurement and traffic behavior analysis.

Shanqing Jiang is currently pursuing the Ph.D. degree in the School of Cyber Science and Engineering, Southeast University. His research interests include cyber security, active defense, and cyber resilience.

Yuyu Zhao received the M.S. degree in cyber security from Southeast University, Nanjing, China, in 2019. He is pursuing the Ph.D. degree in the School of Cyber Science and Engineering at Southeast University, Nanjing, China. His main research interests include cyber security, future network architecture and TCP/IP.

Zihan Chen is currently pursuing the Ph.D. degree in the School of Cyber Science and Engineering, Southeast University. His research interests include cyber security, encrypted traffic analysis, and network scheduling.