Computers, Materials & Continua CMC, vol.62, no.2, pp.763-786, 2020

Strategy Selection for Moving Target Defense in Incomplete Information Game

Huan Zhang[1], Kangfeng Zheng[1, *], Xiujuan Wang[2], Shoushan Luo[1] and Bin Wu[1]

Abstract: As a core component of the network, web applications have become one of the preferred targets for attackers because the static configuration of web applications simplifies the exploitation of vulnerabilities by attackers. Although the moving target defense (MTD) has been proposed to increase the attack difficulty for the attackers, there is no solo approach can cope with different attacks; in addition, it is impossible to implement all these approaches simultaneously due to the resource limitation. Thus, the selection of an optimal defense strategy based on MTD has become the focus of research. In general, the confrontation of two players in the security domain is viewed as a stochastic game, and the reward matrices are known to both players. However, in a real security confrontation, this scenario represents an incomplete information game. Each player can only observe the actions performed by the opponent, and the observed actions are not completely accurate. To accurately describe the attacker's reward function to reach the Nash equilibrium, this work simulated and updated the strategy selection distribution of the attacker by observing and investigating the strategy selection history of the attacker. Next, the possible rewards of the attacker in each confrontation via the observation matrix were corrected. On this basis, the Nash-Q learning algorithm with reward quantification was proposed to select the optimal strategy. Moreover, the performances of the Minimax-Q learning algorithm and Naive-Q learning algorithm were compared and analyzed in the MTD environment. Finally, the experimental results showed that the strategy selection algorithm can enable defenders to select a more reasonable defensive strategy and achieve the maximum possible reward.

Keywords: Moving target defense, Nash-Q learning algorithm, optimal strategy selection, incomplete information game, web service.

1 Introduction

With the continuous development of computer networks and systems in various fields of daily life, the use of web applications has become the most popular way for businesses to provide services over the Internet. Consequently, web applications are vulnerable to various forms of exploits, such as SQL injects, XSS, Trojan. Since web applications are

[1] School of CyberSpace Security, Beijing University of Posts and Telecommunications, Beijing, 100088, China.

[2] College of Computer Sciences, Beijing University of Technology, Beijing, 100124, China.

[*] Corresponding Author: Kangfeng Zheng. Email: zkf_bupt@163.com.

the "front door" for many businesses, the security is of paramount importance.

To defend systems from exploits, the existing network defense techniques, for instance, firewalls, intrusion detection systems and anti-viruses are often employed in web applications. However, the attacks on web applications are becoming more complex, which has made it difficult for the existing defense techniques to counter them. To reduce the cost and information asymmetry between malicious intruders and defenders, the moving target defense (MTD) has been proposed by the networking and information technology research and development program (NITRD). MTD can continuously change the attack surface of the web application and make the targeted application random, dynamic and heterogeneous, which increases the difficulty level for the attackers. It is noted that MTD is used to complement the existing security techniques, not to replace them.

Research on MTD has experienced tremendous growth [Sushil (2013)]. Taguinod et al. [Taguinod, Doupe, Zhao et al. (2015)] divided web applications into different layers and discussed the components that can be shifted in the web application layers. The authors developed a compiler that could perform the translation from Python to PHP and a tool that could automate the conversion or migration between MySQL and PostgreSQL. Heydari et al. [Heydari, Kim and Yoo (2017)] presented an anti-censorship framework using the moving target defense designed using Mobile IPv6. It assigned end-users to random groups and provided them with an IP that the end-users can use to access the website. The framework could enable government or private groups to prevent the public from accessing certain types of information easily. Vikram et al. [Vikram, Yang and Gu (2013)] presented a non-intrusive moving-target defense system that could prevent web bots from automating web resource accesses by randomizing HTML elements without affecting the normal users. In addition, mutating the IP address, route and proxy could also protect web applications [Jafarian, Al-Shaer and Duan (2015); Duan, Al-Shaer and Jafarian (2013); Jia, Sun and Stavrou (2013); Venkatesan, Albanese, Amin et al. (2016)]. Connell et al. [Connell, Menasce and Albanese (2018)] proposed a quantitative analytic model for assessing the availability and performance of resources that are reconfigured by an MTD. Connell et al. [Connell, Pham and Philip (2018)] proposed a method to defend against multi-step attacks by transforming the service platform and IP address, and they analyzed the effectiveness of the method. Some existing studies have proposed various MTD technologies for protecting web applications. However, these MTD technologies are aimed at defending against one specific type of attack, and they cannot defend against other attacks.

Because different network security threatens need to explore different exploration surfaces [Zhuang, DeLoach and Ou (2014)], there is still no sole MTD technology that can defend the system against all possible threats. In addition, running all these mutation technologies simultaneously is not feasible due to the limitation of host or network resources. Therefore, facilitating effective selection of MTD technology has become extremely important in real-time defense. Some existing studies have formulated this problem as a game between the defender and attacker.

Vadlamudi et al. [Vadlamudi, Sengupta, Taguinod et al. (2016)] used the Bayesian Steinberg game to formalize the attack-defense process specifically based on Web platforms to determine an optimal mixed policy. The empirical results showed that the

technique could effectively identify the most critical vulnerabilities and most sensitive attack types. In Steinberg game, the defender knows the attacker's reward function, but this is practically difficult to achieve. Sun et al. [Sun, Li, Xiong et al. (2018)] regarded the network security as a zero-sum multi-objective game. They combined Pareto optimization and Q-learning methods to determine the most harmful attacks and find the best defense strategy against an attack. Manadhata [Manadhata (2013)] presented a two-player stochastic game model to determine an optimal MTD strategy. The defender could choose the optimal strategies to mutate using the concept of the Nash equilibrium. Unfortunately, Manadhata just put forward the theoretical method, and there is no experimental verification. In addition, they did not consider the accuracy of the observation of the attacker's actions. Lei et al. [Lei, Ma and Zhang (2017)] modeled the MTD confrontation as a Markov game (MG-MTD). In this game, the attacker and the defender considered the cost and benefit during the strategy selection, and the cost and benefit could be converted to change the attack surface [Manadhata and Wing (2011)] and the exploration surface. Maleki et al. [Maleki, Valizadeh, Koch et al. (2016)] proposed a Markov model-based framework for MTD analysis. It was proved that the defense was effectively improved when multiple MTD schemes at different layers of the software stack were combined. He et al. [He, Dai, Ning et al. (2015)] proposed an algorithm that combined conventional reinforcement learning with a Bayesian-type identification procedure for IDS configuration. A type identification was conducted by analyzing the actions of the intruder and then updating these using Bayes' formula. Next, an optimal configuration was selected based on the type of intruder. The results showed that this method can identify the intruder type with high accuracy and provide an effective IDS configuration with a given set of reward matrices.

The discussion above indicates that certain achievements have been made in the domain of defense strategy selection. Unfortunately, most of these studies assumed that both the defender and the attacker have the complete information concerning the opponent. In fact, the defenders have limited information regarding the attackers in terms of accurate intent and reward.

Yang et al. [Yang, Niu and Peng (2017)] regarded the incomplete information game as a blind confrontation. They established a channel model that could transform "the attacker or the defender wins one time" to "one bit is transmitted successfully in the channel". Chung et al. [Chung, Kamhoua, Kwiat et al. (2016)] proposed the Q-Learning to learn the opponent's action and make a proper decision in cases that the defender had limited information concerning the attacker's reward and strategies. In terms of performance, Naive-Q Learning outperforms the Minimax-Q Learning. He et al. [He, Dai and Ning (2015)] proposed two algorithms in which the defender can learn and adapt faster in unknown dynamic environments by exploiting the private local information of the environment, which is unknown to the attacker. To accelerate the speed of convergence and avoid the local optimum, Lin et al. [Lin, Wang, Han et al. (2009)] proposed an improved Q-Learning algorithm that combined the Q-learning algorithm, truncated TD estimation and simulated annealing algorithm into a military chess game. The results show that the improved Q-learning converged faster than the simple Q-learning and that it can avoid suboptimum and repeated learning. Sandholm [Sandholm (2015)] abstracted the original game as a similar game. Then, the abstracted game was solved using Nash

equilibrium and mapped back to the original game. Nguyen et al. [Nguyen, Alpcan and Basar (2008); Nguyen, Alpcan and Basar (2009)] viewed the security game as a fictitious play game. The authors analyzed the convergence of the players' strategies; however, these strategies have not been applied yet. Hu et al. [Hu, Liu and Zhang (2018)] transformed the uncertainty of the game characteristics between both sides of the attacker and defender to the uncertainty of each other's type. However, the attacker's reward is considered to be a fixed value in the article, and the authors did not consider the inaccuracy of the observation.

It is common for researchers to apply the Reinforcement Learning algorithm, or an improved method to cope with the incomplete information game. These algorithms rely on observing the actions of the opponent. However, the accuracy of observations is always ignored, leading to a reduction in the practicality of the proposed methods. In addition, these methods are only applied to the general attack defense, and there is no research on the application of the incomplete information game to MTD. In this study, the interaction between a defender and an attacker is viewed as a two-player incomplete game, which aims to maximize the strategic reward by selecting optimal strategies for the defender without exceeding the limitations of resources. To solve the problem of incompleteness, the defender simulates the distribution of the attacker's attack strategies through the historical knowledge, and the reward of attack is corrected through the observation matrix, which made the optimal strategy selection more practical. In terms of the optimal strategy selection, the Nash-Q learning algorithm is applied in MTD for the first time. Further, the experimental results demonstrated the possibility of applying Nash-Q learning to effectively learn an opponent's behavior and make a proper decision.

2 Attack model

This section provides a detailed description of the two sides of the security game in Web applications and establishes an interaction model between the attacker and the defender. On one hand, the defensive party, usually a Web system administrator, is responsible for managing and securing applications. On the other hand, the attacker is a malicious opponent who attempts to compromise the Web application and obtain sensitive data available in the servers.

In the attack model, Web applications have different attack surfaces. Therefore, the attacker can use different attacks to attack the targets.

As shown in Fig. 1, attackers can reach different targets through different types of attack. On the defensive side, the defender can choose different system configurations to change according to the attack types. It should be noted that the chosen defense method is not only capable of defending against one type of attack. It may be able to defend against different attacks, and it may have different defense effect. For example, changing the database can protect against reconnaissance attacks, as well as SQL injection attacks, but it is more effective for SQL injection attacks. Therefore, the defender needs to select the most effective mutation element to transform according to the type of attack.

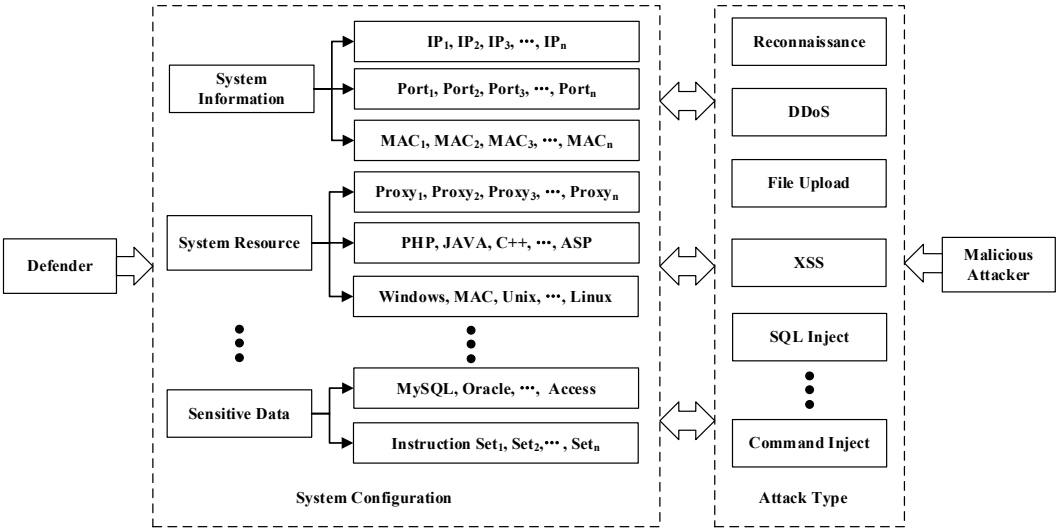


Figure 1: Mapping relationship between the attacker and defender in Web applications

In the context of attackers, a Web attack is any type of offensive maneuver that targets databases, infrastructures or networks. Web attacks mainly include SQL injections, XSS, Trojan Horses, reconnaissance, DDoS and so on. These attacks attempt to expose, alter, disable, destroy, steal or gain unauthorized access to a WEB application. For example, a Distributed Denial of Service (DDoS) attack poses a constant threat to Web services as they may critically deteriorate the service performance or cause the complete shutting down of a website, even if for a brief time. The purpose of DDoS is to consume the target's bandwidth, CPU or service resources. The SQL injection is a code injection technique that inserts nefarious SQL statements into an entry field for execution (e.g., to dump the database contents to the attacker). An XSS enables attackers to inject client-side scripts into web pages viewed by other users; subsequently the attacker can gain session cookies maintained by the browser on behalf of the user. Currently, some sophisticated technologies exist [Boyd, Kc, Locasto et al. (2010); Faghani and Nguyen (2013); Simpson, Shirazi, Marnerides et al. (2018)] to detect and defend against these attacks. However, the use of these technologies is limited and they cannot defend against unknown attacks.

In contrast with the monogamous and passive attributes of traditional defense methods, MTD keeps transforming the resource vulnerabilities of the protected systems by randomly shifting the configuration and the status of network components to deceive and confuse the attackers.

There are many configurations that can be shifted by the defender, such as the IP address, port, databases, MAC, and operating system. Different configurations can prevent different attacks. For example, the use of random IP addresses and ports can disrupt the reconnaissance activities; they can also defend against DDoS attacks and provide certain defense against worm attacks. Replacing one or more proxies and remapping clients to proxies can mitigate DDoS and reconnaissance attacks. When the defender mutates the database and the implementation language, the efforts of the attackers may increase considerably to launch successful SQL inject attacks. It can be seen that a configuration

can defeat a variety of attacks, but the defense effects are different. The simplest way to ensure service security is to transform all configurations at the same time; however, this strategy is unrealistic owing to the limited system resources. As a result, the defender must select an optimal defense configuration according to the attack type.

3 MTD confrontation model based on game theory

Game theory is the study of mathematical models of conflict between intelligent rational decision-makers. Game theory describes a game by specifying the players involved in the game, the order in which the players take actions, the possible actions of the players, each player's knowledge of the previous actions taken and each player's knowledge about the reward function [Liang and Xiao (2013)]. In game theory, a player's strategy is any of the actions the player can choose in a setting in which the reward depends on not only the player's own actions but also the action of the opponent. A player's strategy will determine the action that the player may adopt at any stage of the game.

During an attack process, a malicious intruder chooses an action among a set of possible actions at each internal step of the process. Likewise, the defender needs to arrange some plans to thwart the attack. In practical cases, the security resources of the system are limited. These limited resources must be allocated and scheduled efficiently to avoid predictability. Fortunately, computational game theory can build decision-aids for realizing the allocation of efficient security resources.

In this study, the interactions between an attacker and a defender are modeled as a two-player stochastic game. It is noted that the game theory assumes that each player is rational. This means that each player aims to choose the best strategy that can bring the greatest benefit. As shown in Fig. 2, the game is played in a sequence of stages. At the beginning of each stage, the game is in an initial state S_n . The players select actions from the defend set and attack set, and each player receives a reward that depends on the current state and the chosen actions. Next, the game moves to a new state S_{n+1}. The procedure is repeated at the new state and continues for a finite or infinite number of stages. The total reward for a player is often taken to be the discounted sum of the stage rewards or the limit inferior of the averages of the stage rewards.

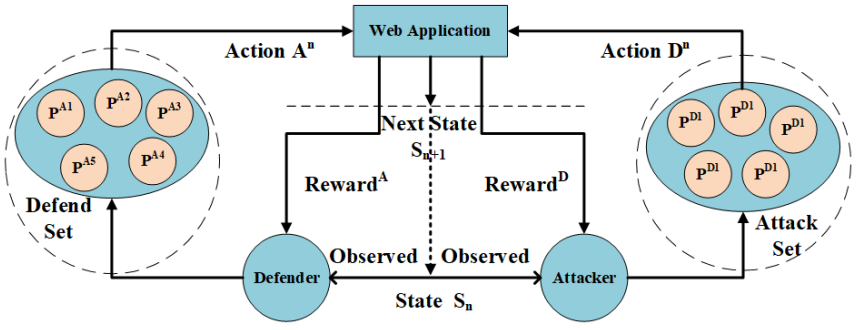


Figure 2: MTD confrontation model

In the MTD confrontation model, the game can be constructed as follows:

Definition: The MTD confrontation model based on a game is a seven-tuple, $< N, S, A, D, RA, RD, P >$; these terms are the basic elements used to describe a game.

(1) $N=\{N_A, N_D\}$ is the player set in the MTD confrontation model, which is assumed to contain only two players; the malicious intruder N_A and the defender N_D.

(2) $A=\{A_1, A_2, A_3..., A_k\}$ is the action set of the attackers. It includes k types of attacks. In each state, attackers can select one action from A to launch. These attacks are independent of each other.

(3) $D=\{D_1, D_2, D_3..., D_l\}$ is the action set of defenders. There are l types of defending methods. In this MTD confrontation model, different defensive actions represent different MTD techniques. Each MTD technology can defend against one or more attacks with different defense effects. For example, the MTD technology based on proxy mutation can defend against scanning attacks and DDoS attacks, and the proxy mutation provides better defense against the scanning attacks than against the DDoS attacks. In addition, due to different resource allocations, the same defensive action has different defensive effects in different states.

(4) $S=\{S_1, S_2, ..., S_n\}$ is the set of system states. Each state contains all the defensive actions, but the defense focus of these states is different. The system resources allocated for defense actions in different states are different. This implies that different states are targeted for different attacks. For example, the defense focuses of state S_n is a DDoS attack and that of state S_{n+1} is an SQL injection attack. In state S_n, the system allocates a large number of port resources for defense, and in state S_{n+1}, the system allocates a large amount of database resources for defense. In contrast, state S_n has a better defense against DDoS attacks than state S_{n+1}, while state S_{n+1} is more effective against SQL injection attacks.

Although each state can defend against all attacks, the effectiveness of the defense is not the same. Due to the limited resources, defenders can only choose one defensive action in each state. Therefore, the defender should choose the appropriate state and defense action. Further, the state transforms constantly according to the confrontation of the defender and intruder with a certain probability.

(5) RA represents the reward function for the attackers. After the attacker has taken action in the game, he/she will receive a negative or a positive return, and this return is his/her reward. However, in an actual scenario, both sides of the MTD confrontation model are not aware of the reward matrix of the other side. This study focuses on methods to determine the reward of the attacker.

(6) RD represents the revenue of the defensive side. When a defender performs an action, the action may benefit the defender. From the above introduction, the damages caused by the same attack may be different in different system states. Hence, the benefit of the same defense method is different in different system states. It is worth noting that the benefit of the defender will be different even in the same state.

(7) $P=\{P(S_j \mid S_i, A_k, D_l)\}\left(S_i, S_j \in S, A_k \in A, D_l \in D\right)$ is the system state transition probability, where $P(S_j \mid S_i, A_k, D_l)$ denotes the possibility of system state transition from S_i to S_j when the defender takes action A_k and attackers launch D_l. The transition of the network state occurs in the process of occurrence of different confrontation phases.

Based on the MTD confrontation model, the selection algorithm is designed as described in the following section.

4 Optimal strategy selection algorithm

In an offensive and defensive game, both sides of the game are rational, and players will choose strategies in response to other players' strategies to maximize their reward. In order to solve this security game, the Nash-Q learning algorithm was used to learn and decide the optimal policy of the defenders.

4.1 Reward quantification

4.1.1 Reward of attacker

In a security game, if the reward matrix of both players is known to each other, players can compute the set of Nash equilibria of the game and play one of these strategies to maximize their expected gain. However, this assumption is generally not practical. Although it is impossible to determine the opponent's reward matrix in advance, it is possible to observe the actions performed by the opponent. Therefore, the rewards of the opponent can be evaluated based on the observed actions.

Let $v_i(j)$ indicate the Boolean value corresponding to the attacker performing action i in state S_j. If the attacker takes action i in state j, then $v_i(j)$ is equal to 1; otherwise $v_i(j)$ is equal to 0. Hence the frequency of attack action i observed after t confrontation times can be expressed as:

$$q(i)=(\sum_{j=0}^{t} v_i(j))/t \tag{1}$$

where $\sum_{j=0}^{t} v_i(j))$ represents the number of times action i is executed in the previous t states. The observed frequency depends on the observation of the attacker's actions. However, these observations are not perfect, since a false positive probability always exists. We define O_D as the observation matrix of the defender, this matrix represents the probability that a defender recognizes attack action m as attack action n, and it satisfies $\sum_{m}^{k} o_n^m = 1$.

$$O_D = \begin{pmatrix} o_1^1 & o_2^1 & \cdots & o_k^1 \\ o_1^2 & o_2^2 & \cdots & o_k^2 \\ \vdots & \vdots & \vdots & \vdots \\ o_1^k & o_2^k & \cdots & o_k^k \end{pmatrix}$$

Let $F_D = \left[q(1), q(2), q(3), \ldots, q(k)\right]$ be the observed frequency of the defender and $\tilde{F}_D = \left[\tilde{q}(1), \tilde{q}(2), \tilde{q}(3), \ldots, \tilde{q}(k)\right]$ be the empirical frequency. The correspondence between F_D and $\tilde{F}_D$ can be expressed as

$$F_D = O_D * \tilde{F}_D \tag{2}$$

Hence, the empirical frequency can be computed by

$$\tilde{F}_D = O_D^{-1} * F_D \tag{3}$$

By counting the empirical frequency of the attacker's actions, it can be considered that a higher action frequency corresponds to a greater reward for the action. This is because the attacker always chooses the action that can maximize the reward to attack. The reward of the attacker depends on the damage of attack and the action taken by the defender. The damage of A_n is defined as $DM(A_n)$. Different types of attacks lead to different damage to the target. For example, the scan attack obtains only the information of the target, while the SQL injection attack can obtain the administrator's username and password, or even the administrator privileges. The damage of SQL injection attack is greater than that of the scan attack, that is, $DM(Scan) < DM(SQLI)$. As described in the previous section, different configurations have different defense effects against different attacks in different states. Let $\eta_j(D_m, A_n)$ represent the defense effect of defense action D_m against attack A_n in state j. When the attack is completely blocked, $\eta_j(D_m, A_n) = 1$. When the defense is completely invalid, $\eta_j(D_m, A_n) = 0$, and in other cases, $0 < \eta_j(D_m, A_n) < 1$. Note that it is unrealistic to completely block an attack; the goal is to improve the defense as much as possible. Thus, the attacker's reward in a different state can be expressed as:

$$RA(S_j, A_n, D_m) = DM(A_n) * \tilde{q}(n) * (1 - \eta_j(A_n, D_m)) \tag{4}$$

where $\tilde{q}(n)$ represents the empirical frequency of the attack A_n, and $1 - \eta_j(A_n, D_m)$ represents the proportion of benefit that the attacker can obtain.

4.1.2 Reward of defender

The reward of the defender depends on the type of attack and the state. However, the defender's observation of the attack is inaccurate. Thus, the reward of the defender in state j can be expressed as

$$RD\left(S_j, A_n, D_m\right) = DM(A_n) * o_n^n * \eta_j(A_n, D_m) \tag{5}$$

where $DM(A_n)$ represents the hazard of the observed attack. o_n^n is the element on the diagonal in the observation matrix O_D, which indicates the probability that the defender accurately identifies the attack A_n. $\eta_j(A_n, D_m)$ represents the defense effect of D_m on A_n in state j.

4.2 Nash-Q learning algorithm

In the Nash-Q learning algorithm, the Q-function for any individual player becomes $Q(S, D, A)$ rather than the single-agent Q-function $Q(S, A)$ or $Q(S, D)$. Therefore, the optimal quality function for the defender is defined as

$$QD(S_j, A_n, D_m) = RD(S_j, A_n, D_m) + \beta * \sum_{S_{j+1} \in S} P(S_{j+1} \mid S_j, A_n, D_m) * V^D(S_{j+1}, \pi_{j+1}^A, \pi_{j+1}^D) \tag{6}$$

where $QD(S_j, A_n, D_m)$ represents the total expected discounted reward for the defender when the defender takes action D_m and the attacker takes action A_n in state S_j and all players follow the Nash equilibrium from then on. β represents the discount factor. π_{j+1}^A and π_{j+1}^D represent the Nash equilibrium strategy of the attacker and defender in state S_{j+1}, respectively. $VD(S_{j+1}, \pi_{j+1}^A, \pi_{j+1}^D)$ represents the optimal value function for the defender in the next state S_{j+1} and it can be represented as

$$VD(S_{j+1}, \pi_{j+1}^A, \pi_{j+1}^D) = NASH^D(QD(S_{j+1}, A_*, D_*), QA(S_{j+1}, A_*, D_*)) \tag{7}$$

where $NASH^D(QD(\bullet), QA(\bullet))$ in Eq. (7) represents the reward of the defender under the Nash equilibrium. D_* and A_* represent the actions taken by both sides of the game under the condition of Nash equilibrium.

Similar to the defender, the optimal quality function $QA(S_j, A_n, D_m)$ and value function $VA(S_{j+1}, \pi_{j+1}^A, \pi_{j+1}^D)$ for the attacker are defined as

$$QA(S_j, A_n, D_m) = RA(S_j, A_n, D_m) + \beta * \sum_{S_{j+1} \in S} P(S_{j+1} \mid S_j, A_n, D_m) * VA(S_{j+1}, \pi_{j+1}^A, \pi_{j+1}^D) \tag{8}$$

$$VA(S_{j+1}, \pi_*^A, \pi_*^D) = NASH^A(QD(S_{j+1}, A_*, D_*), QA(S_{j+1}, A_*, D_*)) \tag{9}$$

Next, the Nash equilibrium strategies π_j^A and π_j^D can be written as

$$\pi_j^D = \arg NASH^D(QD(S_j, A_*, D_*), QA(S_j, A_*, D_*)) \tag{10}$$

and

$$\pi_j^A = \arg NASH^A(QD(S_j, A_*, D_*), QA(S_j, A_*, D_*)) \tag{11}$$

It is apparent that if a defender wants to choose a defense strategy that is consistent with the

Nash equilibrium, the defender must know not only his/her quality function, but also the quality function of attacker. Therefore, after each confrontation, the state transfers from S_j to S_{j+1}, and both sides of the game need to update their quality functions based on the observed actions and rewards respectively. Thus, the updated quality functions become

$$QD'(S_j, A_n, D_m) = QD(S_j, A_n, D_m) + \alpha * [RD(S_j, A_n, D_m) + \beta * VD(S_{j+1}, \pi^A_{j+1}, \pi^D_{j+1}) - QD(S_j, A_n, D_m)] \tag{12}$$

$$QA'(S_j, A_n, D_m) = QA(S_j, A_n, D_m) + \alpha * [RA(S_j, A_n, D_m) + \beta * VA(S_{j+1}, \pi^A_{j+1}, \pi^D_{j+1}) - QA(S_j, A_n, D_m)] \tag{13}$$

where α represents the learning rate. The updated strategies then become

$$\pi'^D_j = \arg NASH^D(QD'(S_j, A_*, D_*), QA'(S_j, A_*, D_*)) \tag{14}$$

$$\pi'^A_j = \arg NASH^A(QD'(S_j, A_* D_*), QA'(S_j, A_* D_*)) \tag{15}$$

A completely specific version of the algorithm is given as follows.

Algorithm 1: Algorithm of Optimal Defense Strategy Selection

Input: Attack action A, Empirical frequency $\tilde{F}_D$, Defense action D

Output: optimal defense strategy π

1. Initialization:
2. Let $QA = 0$, $QD = 0$, $VD = 0$, $VA = 0$, $j \leftarrow 0$
3. Initialize(state S_0)
4. Choose Action:
5. if random() < *explor* then
6. return Random
7. return π^D_j
8. end if
9. Learn:
10. Update($\tilde{F}_D$),Update(RA)
11. System State $S_{j+1} \leftarrow S_j$
12. Update (QD'), Update(QA'), Update(π'^D_j),Update(π'^A_j)
13. $j+1 \leftarrow j$

In this algorithm, parameter *explor* controls how often the agent will deviate from its current policy to ensure that the state space is adequately explored.

5 Experiments and analyses

5.1 Experimental environment

In this study, the experiments are simulated and computed using Python and MATLAB. The experimental environment includes Python2.7 and Virtual Machine with Ubuntu 16.04 and 8 GB memory, MATLAB R2017b and a server with Windows 10 and Intel Xeon E5-2620 CPU and 32 GB memory.

5.2 Initializing parameters

In this experiment, the assumption is that the attacker can launch three different attacks A_1, A_2, A_3 and the defender can choose one of the actions from D_1, D_2, D_3 to defend. The action information of each is given in Tab. 1.

Table 1: Set of Actions

Attacker	Defender
A_1 : Reconnaissance	D_1 : IP Address
A_2 : SQL Inject	D_2 : Database
A_3 : Trojan	D_3 : Operating System

The damages of different attacks are different, and the damage $DM(A_n)$ can be divided into three levels: high damage, medium damage and low damage. The values of the high, medium, and low damage are 20, 10 and 5, respectively. Based on experience, the degrees of damage of the SQL injection attack and Trojan attack are high, and the degree of damage of the reconnaissance attack is low. In addition, the effectiveness of the defensive actions against the attack actions can be divided into five levels $\{\eta 1, \eta 2, \eta 3, \eta 4, \eta 5\}$. When $\eta 1 = 0$, the defensive action has no effect on the attack, and $\eta 5 = 1$ indicates that the defensive action can completely defend against the attack, however, this scenario is unrealistic. Then, we assign $\eta 4$, $\eta 3$ and $\eta 2$ values of 0.75, 0.5 and 0.25, respectively.

The system can be in one of three states, $S = \{S_1, S_2, S_3\}$. Each state contains all defensive actions, but the mutating ranges and periods are different. $S_1 = \{(B\ class, 10), (2DB, 60), (2SYS, 60)\}$ indicates that a defender can select a new IP address from the B class IP address, and the mutating period is 10 seconds; $(2DB, 60)$ indicates that the mutant element is Database type, the value range is 2 and its period is 60 seconds. When the defender chooses to mutate the operating system in state S_1, $(2SYS, 60)$ indicates that two operating systems must be changed every 60 seconds. Analogously, $S_2 = \{(B\ class, 60), (4DB, 30), (2SYS, 60)\}$ and $S_3 = \{(B\ class, 60), (2DB, 60), (4SYS, 30)\}$. It can be seen that the three states have different defensive priorities. S_1 mainly defends against reconnaissance attack, S_2 mainly defends against SQL Injection attack, and S_3 mainly defends against Trojan attack.

Therefore, let $\eta_1(A_1, D_1) = \eta_2(A_2, D_2) = \eta_3(A_3, D_3) = \eta 4$; the other defense efficiencies are determined depending on the ratio of the available resources. For example, the IP resource in S_1 is six times that of S_2. Therefore, the defense effect against Reconnaissance attacks in S_1 is also six times that of S_2, that is, $\eta_1(A_1, D_1) = 6 * \eta_2(A_1, D_1)$. Thus, the defense effects in different states are as presented in the following tables.

Table 2: Defense effects in S_1

Defensive actions	Attack actions		
	A1	A2	A3
D1	0.75	0	0
D2	0.125	0.1875	0
D3	0.1875	0	0.1875

Table 3: Defense effects in S_2

Defensive actions	Attack actions		
	A1	A2	A3
D1	0.125	0	0
D2	0.5	0.75	0
D3	0.1875	0	0.1875

Table 4: Defense effects in S_3

Defensive actions	Attack actions		
	A1	A2	A3
D1	0.125	0	0
D2	0.125	0.1875	0
D3	0.75	0	0.75

The frequency of attack actions F_D can be continuously updated through observation. As a result, the reward of the attacker in each confrontation process can be obtained using Eq. (4). Besides, the transition probability between the states also depends on the defense effect of the different states. The system always tends to shift to a state with a satisfactory defense effect. For example, when the attacker adopts A_n and the defender adopts D_m, the defense efficiencies in the three states are $\eta_1(A_n, D_m)$, $\eta_2(A_n, D_m)$, $\eta_3(A_n, D_m)$. Therefore, the transition probability can be expressed as

$$P(S_j \mid S_i, A_n, D_m) = \frac{\eta_j(A_n, D_m)}{\sum_{c=1}^{3} \eta_c(A_n, D_m)} \tag{16}$$

According to Eq. (5), the reward of a defender also depends on the observation matrix, and the observation matrix depends on the accuracy of the detection device, such as IDS. However, the detection of attacks is not the focus of this paper. The observation matrix of the defender is set as $O_D = \begin{pmatrix} 0.6 & 0.1 & 0.05 \\ 0.2 & 0.8 & 0.05 \\ 0.2 & 0.1 & 0.9 \end{pmatrix}$.

5.3 Algorithm comparisons and analyses

The numerical results presented in this section are used to validate the effectiveness of the algorithm of the optimal strategy selection. In the experiments, three different policies for defending were adopted: the Nash-Q learning algorithm, Minimax-Q learning algorithm and Naive-Q learning algorithm. In each learning algorithm, the defender was trained against a Nash-Q learning, Minimax-Q learning, and Naive-Q learning opponent. The performance of different algorithms is compared and discussed in the following section.

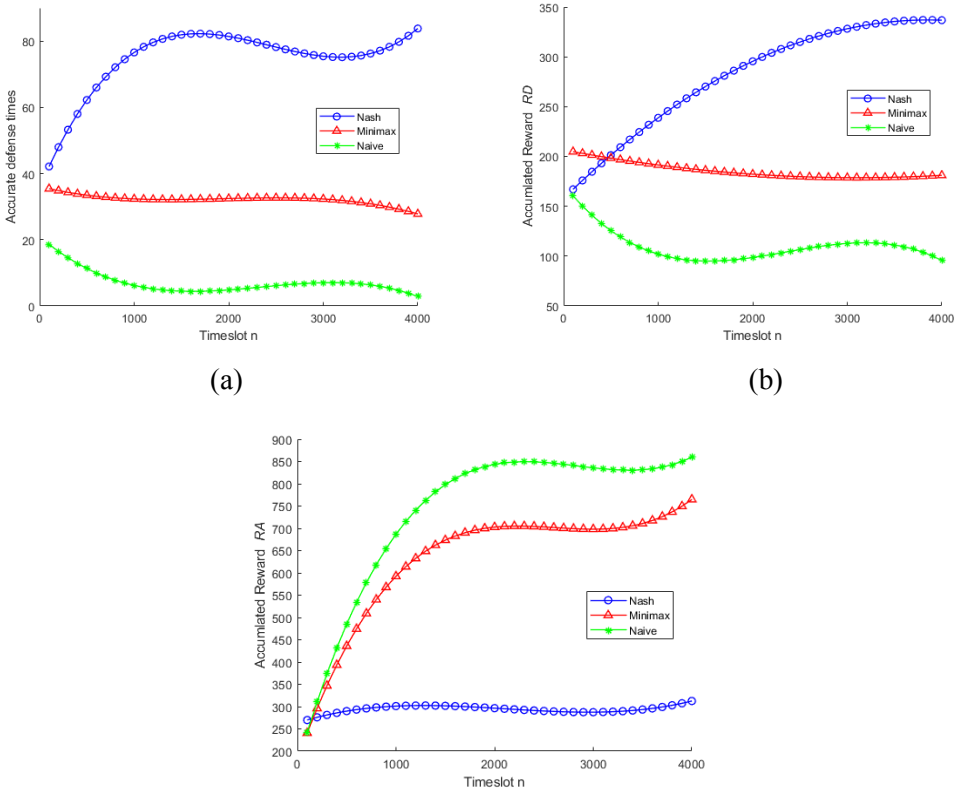
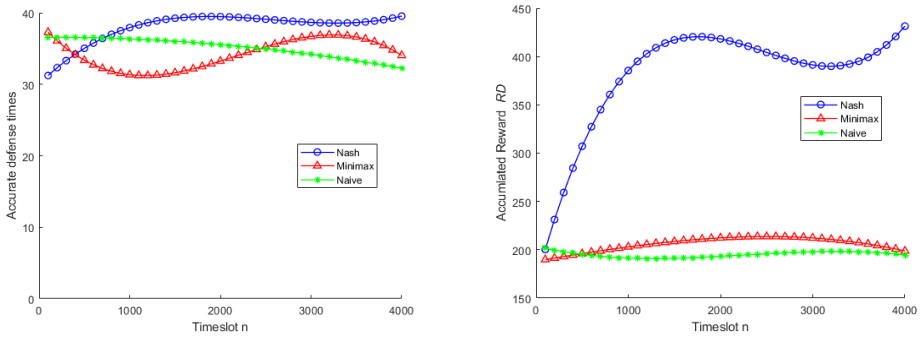
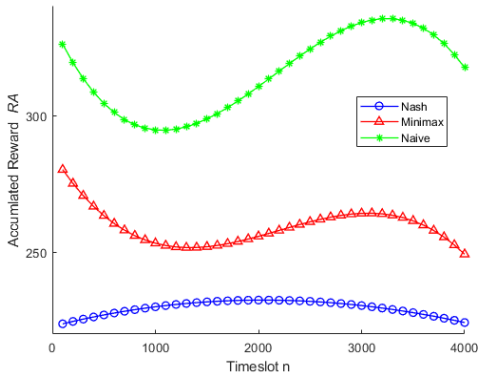


Figure 1: Results for different policies against the Nash-Q learning attacker with *explor*=0.9 and α=0.02, β=0.9, for attacker and α=0.02, β=0.9 for the Nash-Q defender; α=1, β=0.9, $decay = 0.9984$ for the Minimax-Q defender and

α=0.02，β=0.9 for the Naive-Q defender, (a) Accurate defense time for each defender, (b) Accumulated reward for each defender, (c) Accumulated reward for attacker against different defender

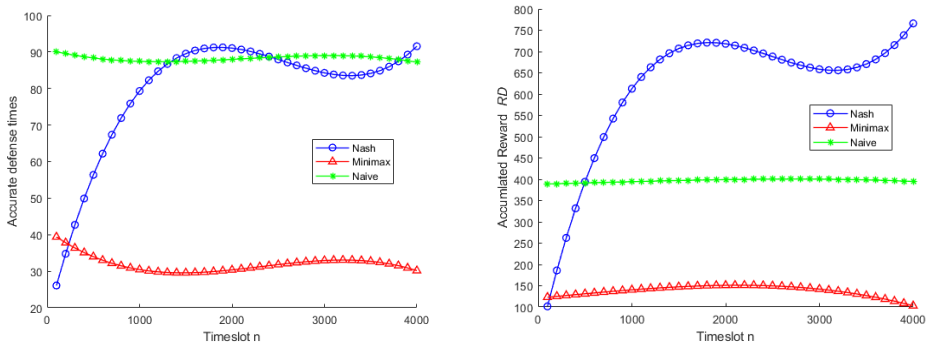


(a) (b)

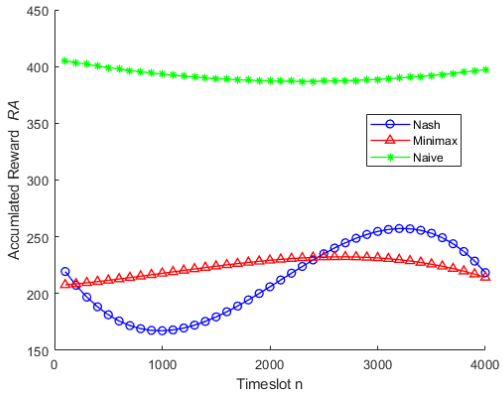


(c)

Figure 4: Results for different policies against the Minimax-Q learning attacker with *explor*=0.9 and α=1，β=0.9, $decay = 0.9984$ for attacker and α=0.02，β=0.9 for the Nash-Q defender; α=1，β=0.9, $decay = 0.9984$ for the Minimax-Q defender, α=0.02，β=0.9 for the Naive-Q defender, (a) Accurate defense time for each defender, (b) Accumulated reward for each defender, (c) Accumulated reward for attacker against different defender



(a) (b)



(c)

Figure 5: Results for different policies against the Naive-Q learning attacker with *explor*=0.9 and α=0.02, β=0.9 for attacker and α=0.02, β=0.9 for the Nash-Q defender; α=1, β=0.9, $decay = 0.9984$ for the Minimax-Q defender, α=0.02, β=0.9 for the Naive-Q defender, (a) Accurate defense time for each defender, (b) Accumulated reward for each defender, (c) Accumulated reward for attacker against different defender

Fig. 3(a) showed the accurate defense time when the attacker was trained via the Nash-Q learning and the defender was trained using different policies. An accurate defense indicated a situation in which the defender selected the best defense strategy. For example, when the attack was a SQL injection attack, the defender chose the database to make a mutation. When the attack was a Trojan attack, the defender should choose the operating system to perform a mutation. It could be clearly seen from Fig. 3(a) that when the defender adopted the Nash-Q learning strategy, the time required to attain the accurate defense was the maximum. When the defenders adopted Minimax-Q learning and Naive-Q learning, the results corresponding to the defense were inferior to those of Nash-Q learning, and the results for the Naive-Q learning were the worst.

This situation can be explained as follows: When a defender adopted the Nash-Q learning algorithm, the defender maintained a model of the attacker's Q-function and used this

information to update his/her own Q-function. Both the attacker and defender acted based on the equilibrium in each state. Moreover, the attacker's Q-function depended on the actions of the attack, as observed by the defender. When the defender was trained by the Minimax-Q learning algorithm, the defender endeavored to maximize his/her reward while the attacker strived to minimize it. Consequently, the defender's minimum expected reward should be as large as possible. Nevertheless, when the attacker was trained by the Nash-Q learning algorithm, the actual action chosen by the attacker might not have been the action that could minimize the reward of the defender. Since Minimax mainly solved the strategic problems of a dynamic game, the defender only obtained the information concerning his/her action and the minimum reward that he/she could acquire. The defender's strategy was a probability distribution over the actions. In contrast to the one trained by Minimax-Q learning, the defender trained by Nash-Q learning can obtain more accurate information of the attacker's actions and take more accurate defense measures; thus, their result was more accurate. However, if the defender was trained by the Naive-Q learning algorithm, the accurate defense time of convergence was less than 10 times. This was because the Nash-Q learning algorithm can provide more information about the attacker than the Naive-Q learning algorithm can, leading to the formulation of a more accurate confrontation model. Naive-Q Learning cannot provide the information about the attacker that is required to formulate a complete confrontation model, and the defender adopts defensive strategies depending on his/her own past actions. Consequently, when the Naive defender defended a rational attacker such as a Nash-Q defender, Naive-Q learning may misguide the defender to blindly learn the worst-case policy and thus lead to performance loss.

Fig. 3(b) and Fig. 3(c) demonstrated the accumulated rewards of the defenders and attackers in every 100 confrontations, respectively. The defense reward and attack reward correspond to the time of accurate defense. As the accurate defense time increased, the defender's reward increased and the attacker's reward decreased, and, vice versa.

Fig. 4 and Fig. 5 showed the results corresponding to the attackers trained by the Minimax-Q learning algorithm and the Naive-Q learning algorithm. Fig. 4 that the defensive result was the best when the defender adopted the Nash-Q learning algorithm. In Fig. 4(b), when the timeslot was greater than 2,000, the defender's reward gradually decreases. This was because of the presence of uncertainty in the learning process, and because defenders could not ensure that every defensive effect was better than the previous one. In particular, the accumulated defense of the Naive-Q was similar to the Minimax-Q, but the accumulated attack reward obtained during the entire confrontation was less than that for Minimax-Q. This was because the attacker's goal was to maximize his/her reward as a feature of Minimax.

When the attacker was a Naive-Q learner, the defenders can perform effective defense when the defenders adopted Nash-Q learning. With the increased in the number of confrontations, the Nash-Q learning defender achieved a higher reward and this would reduce the attacker's reward to the lowest. When the defender also adopted the Naive-Q algorithm, the average accurate defense time was the highest, but the defender's defense reward was lower than that for a Nash-Q learning defender. By observing the specific experimental process, it was found that although Naive-Q learning had a higher defense

success rate, it was not in the most appropriate state. For example, when an attacker launched a Trojan attack, the defender choosed to change the operating system, but the current state was instead of. Therefore, despite the success of the defense, the defender gets less reward than the Nash-Q learning defender. Besides, when the defender adopted Minimax-Q learning, the number of accurate defenses was the least and the defense reward was the least; however, the attacker received a lower reward than the Naive-Q learner did. The experimental process showed that the attacker always performed a reconnaissance attack in this confrontation, and the reconnaissance attack had the lowest damage.

To illustrate the effectiveness of return simulation, we calculated the reward of attack without simulation. Similar to Hu et al. [Hu, Liu and Zhang (2018)], the rewards became some fixed value. Then Eq. (4) becomed

$$RA\left(S_j, A_n, D_m\right) = DM(A_n) * (1 - \eta_j(A_n, D_m)) \tag{17}$$

The results of the comparison between the without simulation and the method presented in this paper are as follows:

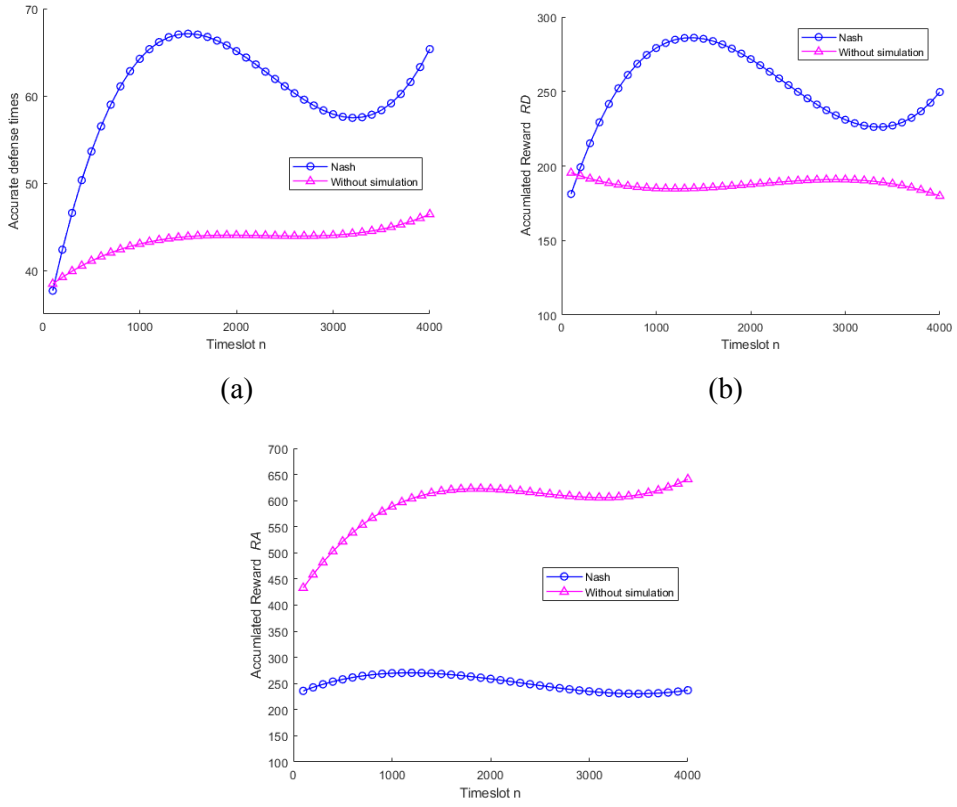


Figure 6: Results for different policies against the Nash-Q learning attacker with *explor*=0.9 and α=0.02, β=0.9 for attacker and α=0.02, β=0.9 for the Nash-Q

defender with reward simulation, α=0.02, β=0.9 for the Nash-Q defender without reward simulation; (a) Accurate defense time for each defender, (b) Accumulated reward for each defender, (c) Accumulated reward for attacker against different defender

Fig. 6(a) showed the accurate defense time when the attacker was trained via the Nash-Q learning and the defender was trained via different Nash-Q learning. It can be seen from the experimental results that although defenders all used Nash -q algorithm to select defense strategies, the method proposed in this paper can select more accurate defense strategies in the defense process. This was because it is inaccurate to determine the attacker's reward only by the observed actions. As described in Section 4.1.1, due to the presence of false positives, the defender often has errors in the observation of the attacker's actions. In addition, the attacker's reward will also be affected by the previous attack experience. The more times an attack type was selected, the higher the reward of this attack may be. On the contrary, if the number of times selected was very small, the reward of the attack will be very low. When the number of confrontations exceeded 1000, the accurate defenses time was reduced. This was caused by the uncertainty of the NASH-Q algorithm in the learning process. As can be seen from the results, although the accurate defenses time decreases, the defense effect was still better than that when the attack rewards were not simulated. Fig. 6(b) and Fig. 6(c) showed the return of defenders and attackers, respectively. The comparison results showed that the proposed method in this paper can make the defender get higher reward while effectively limiting the attacker's reward.

Through the above experiments, it could be concluded that Nash-Q Learning performs well against irrational attackers and proved that it was necessary to select defense methods by simulating the reward of attackers in the incomplete information game.

5.4 Complexity analyses

For the Nash-Q Learning algorithm, the main operation was traversing the Nash matrix and updating the empirical frequency of the attacker's action. The size of the matrix depended on the number of defender actions and attacker actions. Therefore, the complexity of the Nash-Q learning in each confrontation was $O(K*L)$, where L indicated the number of actions of the defender and K indicated the number of actions of the attacker. The complexity of the entire process was $O(N*K*L)$, where N denoted the number of confrontations. For the Naive-Q learning algorithm, the main operation was traversed and updated the quality function of each player, its complexity was also $O(N*K*L)$. When the defender adopts the Minimax-Q learning algorithm, the main operation was to solve the linear programming problem, and the complexity of this algorithm in each confrontation is $O(2^L)$. The complexity of the entire process was $O(N*2^L)$. It could be concluded that the complexity of the Minimax-Q learning algorithm was higher than that of the other two algorithms.

5.5 Burden analyses

In the experiments mentioned above, only one mutation element was selected for each

confrontation due to the limited resources of the system. However, the case in which defenders chose multiple mutation elements simultaneously must be evaluated. If multiple elements were mutated at the same time, the system may be more secure, but the burden on the system increased accordingly. The experiment discussed in the following paragraph was performed to investigate the relationship between the system burden and the number of mutant elements. In this experiment, both the defender and the attacker adopted the Nash-Q learning to choose their own strategies.

An Apache server was set up in a laboratory environment. This server may suffer from reconnaissance attacks, SQL injection attacks and Trojan attacks. The attacks could be prevented by changing the IP address, database type, and operating system type. Further, the page loading time (PLT) was measured to examine the time overhead of web applications when the server changes these elements. PLT was the time interval needed to load the complete page on the browser.

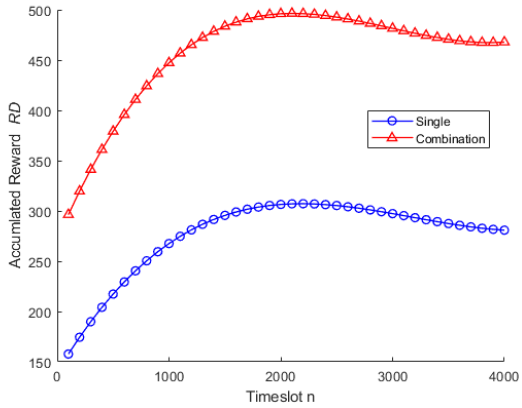


Figure 7: Reward of the defender when the defender selects single element and combinatorial-elements

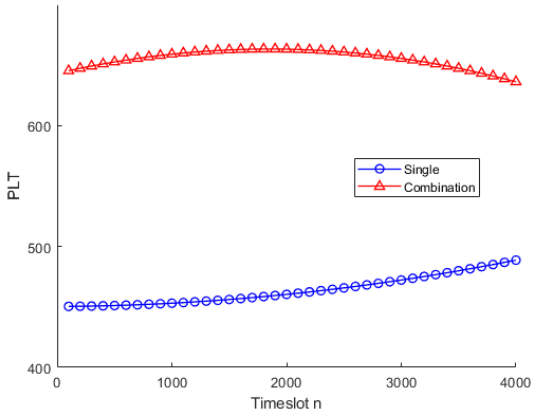


Figure 8: PLT when the defender selects single element and combinatorial-elements Tables

Fig. 7 showed the rewards that a defender could achieve while using single-element mutation and combinatorial mutation. The reward of the defender was set the same as that defined above. It is apparent that the reward obtained using the combinatorial-elements achieved higher defensive benefits than the single element did. The average reward when using a single element was 276, and that using the combinatorial-elements was 456, which represented an improvement of 65%.

In this experiment, it was assumed that the transformation periods of the different elements were the same. Fig. 8 depicted the PLT of a user accessing a web service in different defense scenarios throughout the confrontation process. Each point in the graph represented the average PLT against 100 runs. It could be clearly seen from the results that when an attacker adopted combinatorial-elements, the value of PLT was greater than the PLT of the defender using a single transformation element. The statistics showed that when the defender changed a single transformation element, the average PLT during the entire confrontation process was 463 ms, and the average PLT of combinatorial-elements was 655 ms.

These results indicated that although the combinatorial-elements mutation increased the security of the system, it induced an extra average time burden of 41%. Moreover, the page loading time will be doubled or extended even longer, which may lead to an unpleasant user experience. In an actual defense, the transition period of different elements was different, thus the PLT in the experiment was higher than the PLT in an actual defense scenario; however, the overall trend was the same. The user still needed to spend more time to load the complete page when the defender changed the combinatorial-elements. Therefore, it was more reasonable to use a single element for mutation to ensure the quality of user experience in the case of limited resources.

6 Conclusions

As a revolutionary defense method, MTD can reduce the information asymmetry between an attacker and defender. Choosing the right mutation element is the key to defend against various attacks. To solve the problem of incomplete information in the game model, the attacker's reward matrix was simulated by observing the attacker's action frequency. The Nash-Q learning algorithm was used to select an optimal defense strategy. To illustrate the effectiveness of the selected method, Nash-Q learning was compared with Minimax-Q learning and Naive-Q learning. The results showed that when the defender got trained by Nash-Q learning, the defensive performance was the best. Regardless of the type of attacker, Nash-Q learning can maximize the reward of the defender and minimize the reward of the attacker. This indicates that the theory of Nash can reflect actual scenarios better than other theories. Furthermore, the results of the experiments showed that when multiple mutation elements were changed simultaneously, the performance of defense will be enhanced, but an unpleasant user experience ensues.

This study assumed that web applications will only suffer from one type of attack each time. Thus, to the future research will address the selection of the best defense strategy to defend against multiple types of attacks simultaneously.

Acknowledgement: Thanks for the valuable review comments of every expert and editor. This paper is supported by the National Key R&D Program of China (2017YFB0802703), the National Nature Science Foundation of China (61602052).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Boyd, S. W.; Kc, G. S.; Locasto, M. E.; Keromytis, A. D.; Prevelakis, V. (2010): On the general applicability of instruction-set randomization. *IEEE Transactions on Dependable and Secure Computing*, vol. 7, no. 3, pp. 255-270.

Chung, K.; Kamhoua, C. A.; Kwiat, K. A.; Kalbarczyk, Z. T.; Iyer, R. K. (2016): Game theory with learning for cyber security monitoring. *IEEE 17th International Symposium on High Assurance Systems Engineering*, pp. 1-8.

Duan, Q.; Al-Shaer, E.; Jafarian, H. (2013): Efficient random route mutation considering flow and network constraints. *IEEE Conference on Communications and Network Security*, pp. 260-268.

Faghani, M. R.; Nguyen, U. T. (2013): A study of XSS worm propagation and detection mechanisms in online social networks. I*EEE Transactions on Information Forensics and Security*, vol. 8, no. 11, pp. 1815-1826.

He, X.; Dai, H.; Ning, P.; Dutta, R. (2015): Dynamic IDS configuration in the presence of intruder type uncertainty. *IEEE Global Communications Conference*, pp. 1-6.

He, X.; Dai, H.; Ning, P. (2015): Improving learning and adaptation in security games by exploiting information asymmetry. *IEEE Conference on Computer Communications*, pp. 1787-1795.

Heydari, V.; Kim, S. I.; Yoo, S. M. (2017): Scalable anti-censorship framework using moving target defense for Web servers. *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 5, pp. 1113-1124.

Jafarian, J. H.; Al-Shaer, E.; Duan, Q. (2015): An effective address mutation approach for disrupting reconnaissance attacks. *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2562-2577.

Jia, Q.; Sun, K.; Stavrou, A. (2013): Motag: moving target defense against internet denial of service attacks. *22nd International Conference on Computer Communication and Networks*, pp. 1-9.

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

Connell, W.; Menasce, D. A.; Albanese, M. (2018): Performance modeling of moving target defenses with reconfiguration limits. *IEEE Transactions on Dependable and Secure Computing*, no. 99, pp. 1.

Connell, W.; Pham, L. H.; Philip, S. (2018): Analysis of concurrent moving target defenses. *Proceedings of the 5th ACM Workshop on Moving Target Defense*, pp. 21-30.

Lei, C.; Ma, D. H.; Zhang, H. Q. (2017): Optimal strategy selection for moving target defense based on Markov game. *IEEE Access*, vol. 5, pp. 156-169.

Liang, X.; Xiao, Y. (2013): Game theory for network security. *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 472-486.

Lin, J.; Wang, X.; Han, L.; Zhang, J.; Xu, X. (2009): The improvement of Q-learning applied to imperfect information game. *IEEE International Conference on Systems, Man and Cybernetics*, pp. 1562-1567.

Maleki, H.; Valizadeh, S.; Koch, W.; Bestavros, A.; van Dijk, M. (2016): Markov modeling of moving target defense games. *Proceedings of the 2016 ACM Workshop on Moving Target Defense*, pp. 81-92.

Manadhata, P. K. (2013): *Game Theoretic Approaches to Attack Surface Shifting*. *Moving Target Defense II*. Springer, London.

Manadhata, P. K.; Wing, J. M. (2011): An attack surface metric. *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 371-386.

Nguyen, K. C.; Alpcan, T.; Basar, T. (2009): Security games with incomplete information. *IEEE International Conference on Communications*, pp. 1-6.

Nguyen, K. C.; Alpcan, T.; Basar, T. (2008): Fictitious play with imperfect observations for network intrusion detection. *Preprints of the 13th International Symposium on Dynamic Games and Applications*.

Hu, H.; Liu, Y. L.; Zhang, H. Q.; Pan, R. X. (2018): Optimal network defense strategy selection based on incomplete information evolutionary game. *IEEE Access*, vol. 6, pp. 29806-29821.

Sandholm, T. (2015): Solving imperfect-information games. *Science*, vol. 347, no. 6218, pp. 122-123.

Simpson, S.; Shirazi, S. N.; Marnerides, A.; Jouet, S.; Pezaros, D. et al. (2018): An inter-domain collaboration scheme to remedy DDoS attacks in computer networks. *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 879-893.

Sun, Y.; Li, Y.; Xiong, W.; Yao, Z. H.; Moniz, K. et al. (2018): Pareto optimal solutions for network defense strategy selection simulator in multi-objective reinforcement learning. *Applied Sciences*, vol. 8, no. 18, pp. 136.

Sushil, J. (2013): *Application of Game Theory and Adversarial Modeling. Moving Target Defense II*. Springer, London.

Taguinod, M.; Doupe, A.; Zhao, Z.; Ahn, G. J. (2015): Toward a moving target defense for web applications. *IEEE International Conference on Information Reuse and Integration*, pp. 510-517.

Vadlamudi, S. G.; Sengupta, S.; Taguinod, M.; Zhao, Z. M.; Doupé, A. et al. (2016): Moving target defense for web applications using bayesian stackelberg games. *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pp. 1377-1378.

Vikram, S.; Yang, C.; Gu, G. F. (2013): Nomad: towards non-intrusive moving-target defense against web bots. *IEEE Conference on Communications and Network Security*, pp. 55-63.

Yang, Y.; Niu, X.; Peng, H. (2017): Games based study of nonblind confrontation. *Mathematical Problems in Engineering*, vol. 2017, no. 2, pp. 1-11.

Zhuang, R.; DeLoach, S. A.; Ou, X. (2014): Towards a theory of moving target defense. *Proceedings of the First ACM Workshop on Moving Target Defense*, pp. 31-40.