Hindawi
Security and Communication Networks
Volume 2018, Article ID 2058429, 12 pages
https://doi.org/10.1155/2018/2058429

WILEY | Hindawi

Research Article

A Dynamic Hidden Forwarding Path Planning Method Based on Improved Q-Learning in SDN Environments

Yun Chen, Kun Lv, and Changzhen Hu

School of Software, Beijing Institute of Technology, Beijing, China

Correspondence should be addressed to Kun Lv; kunlv@bit.edu.cn

Received 10 January 2018; Accepted 12 March 2018; Published 23 April 2018

Academic Editor: Zheng Yan

Currently, many methods are available to improve the target network's security. The vast majority of them cannot obtain an optimal attack path and interdict it dynamically and conveniently. Almost all defense strategies aim to repair known vulnerabilities or limit services in target network to improve security of network. These methods cannot response to the attacks in real-time because sometimes they need to wait for manufacturers releasing corresponding countermeasures to repair vulnerabilities. In this paper, we propose an improved Q-learning algorithm to plan an optimal attack path directly and automatically. Based on this path, we use software-defined network (SDN) to adjust routing paths and create hidden forwarding paths dynamically to filter vicious attack requests. Compared to other machine learning algorithms, Q-learning only needs to input the target state to its agents, which can avoid early complex training process. We improve Q-learning algorithm in two aspects. First, a reward function based on the weights of hosts and attack success rates of vulnerabilities is proposed, which can adapt to different network topologies precisely. Second, we remove the actions and merge them into every state that reduces complexity from $O(N^3)$ to $O(N^2)$. In experiments, after deploying hidden forwarding paths, the security of target network is boosted significantly without having to repair network vulnerabilities immediately.

1. Introduction

A defense strategy represents a series of defense methods in the target information system network that can reduce the attack success rate of attackers. Currently, many methods are available to generate a defense strategy. The most important problem is the game between cost and performance. The defense strategy may own excellent performance, but defenders scan and recapture the information system in most instances, which is very uneconomic.

Generally speaking, whether it is SDN or traditional network, we can plan defense strategy through locate optimal attack path. Regarding this method, a majority of previous papers specify generating a complete attack graph [1–3]; however, in a very large computer cluster, the state explosion problem tends to affect the attack graph generation. Thus, the optimal attack path cannot be modeled quickly, and in extreme cases, it may not be possible to determine the optimal attack path. In [4], authors use the ant colony optimization (ACO) approach to search the optimal attack path based on the minimal attack path [5], but ACO can easily fall into a local optimum. Reference [6] proposes a HMM-based attack graph generation method, and then authors use ACO-based algorithm to compute the optimal attack path. Based on this path, evaluating the security of target network can be evaluated and corresponding countermeasures can be planned. But this method primarily handles the known vulnerabilities. Reference [7] proposes a malicious nodes-based security model enacting method, but its performance on handling zero-day vulnerability is not strong enough.

Reinforcement learning [8] is an area of machine learning inspired by behaviorist psychology, concerned with how software agents ought to take actions in an environment so as to maximize a cumulative reward. It differs from standard supervised learning in that correct input/output pairs are never presented, nor are suboptimal actions explicitly corrected. Further, the focus is on online performance, which involves finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge).

To determine a method that can model the optimal defense strategy in many conditions, the algorithm of the method must not depend on all states of the target network and it should be able to decide which atomic attack should be picked to be the next state dynamically. These abilities depend on the characteristics of the specific algorithm.

In this paper, the optimal attack path between source node and target node is computed by the improved Q-learning algorithm. Concretely, in the first stage, we collect information of known vulnerabilities and corresponding type of hosts from national vulnerability database (NVD) [9]. Then, a fuzzy neural network will be used to train these samples to gain the host weight. After getting host weight, the reward function of improved Q-learning can be built. Using this function, the optimal attack path can be located between source node and target node.

In Section 2, we will introduce Q-learning, which is followed by an overview of the main contributions of this paper. In Section 3, the definition of a network model will be discussed. We propose a reward function, and the optimal forwarding path will be built based on this function. In Section 4, we will discuss how to implement this method in a real information system network.

Section 5 provides the experimental results for the optimal protective path and discusses how to improve the Q-learning algorithm. The paper concludes with a summary and future work in Section 6.

1.1. Related Work. There are several works in defense strategy planning in recent years. Currently, many methods are available to generate a defense strategy and these methods can be classified into three categories. First, we can compute the optimal attack path using attack graph and enact policies to destroy the path. Second, we can locate the not-trust nodes in target network and plan countermeasures to prevent these nodes from being exploited by attackers. Third, special strategies can be designed to aim to specific network environments specific attack types. But all of these methods have inherent defects.

Regarding the first method, the intrinsic of this method is that system generates attack graph of target network and then finds and interdicts the optimal attack path in the attack graph. Wang et al. [6] propose a HMM-based attack graph generation method and ACO-based algorithm to evaluate the security of target network and plan corresponding countermeasures. This method can compute the transition probability between each two states. Based on the probability and ACO-based algorithm, the shortest attack path can be found, which can be used to evaluate security metrics. But this method owns some defects. Firstly, the complexity of ACO algorithm is $O(Nc * n^2 * m)$ (Nc is the iteration number, n is the number of vertices, and m is the number of ants in ant colony) that is too high to computation if we process computer cluster. Secondly, this technique's performance is not good enough when it deals with APT and zero-day vulnerabilities, because the interval of time series of HMM is slight less than the interval of APT and this method uses Common Vulnerability and Exposures (CVE) [10]. Ghosh et al. [4] proposes an ACO-based defense strategy planning method. This method is similar as [6]. It uses minimal attack graph to locate the optimal attack path, but this path may not be the global optimal and this attack graph will also show state explosion issue if it is used in very large computer clusters.

For the second technique, the core of this technique is to find the malicious nodes. Akbar et al. [7] propose a Support Vector Machine (SVM) and rough set-based security model building method. In that paper, authors use SVM and rough set to classify the nodes in target network as trust nodes, strange nodes, and malicious nodes. This technique can also acquire the transaction success rate. This method can handle zero-day vulnerabilities in some conditions, but it needs a large number of sample data to training SVM that is impossible to obtain enough data set in some network environments because the data need to spend a lot of time to collect.

Regarding the third method, the key of this method is to handle specific attack types or vulnerabilities. Hu et al. [11] characterize the interaction between defender and APT attacker and an information-trading game among insiders as a two-layer game model. Through their analysis, the existence of Nash Equilibrium for both games is certified and the security metric can be evaluated. But this method can only process APT; the generalization of it is limited. Same as [11], Wang et al. [12] propose a k-zero-day safety method. It starts with the worst case assumption that this is not measurable which unknown vulnerabilities are more likely to exist in the target network and ends to the number of zero-day vulnerabilities that can destroy the network asset. But the complexity of computing this metric is exponential in the size of the zero-day attack graph. Furthermore, the zero-day attack graph cannot reflect the condition of known vulnerabilities related work.

1.2. Contribution. In this paper, we use an improved Q-learning algorithm to generate the optimal attack path. In Q-learning [8], which action will be selected is based on a reward function. In other words, a large number of sample data are not required, as is the case in many other machine learning algorithms. Compared to temporal difference learning, Q-learning can directly iterate an optimal policy, which in this paper is the optimal attack path. Defining the reward function is the key issue in Q-learning. In this paper, we use the host weight and attack success rate of atomic attacks to build a reward function. Specifically, the host weight is decided by the position the host stays in and services the host offers. Besides, we improve the structure of state matrix in Q-learning. The dimension of the matrix is reduced, which can lower the space complexity. Furthermore, the network model that reflects the configuration of the target network will be used to analyze the result of Q-learning.

Our ultimate goal is to build a hidden forwarding path. In this path, we create virtual hosts that provide specific defense strategies in SDN to filter specific attacks. These hosts can be created or deleted dynamically, which can ensure the computation of hidden forwarding paths will occupy the SDN controller's minimal memory space when we want to change the routing path. Furthermore, through using the hidden forwarding paths, vulnerabilities are filtered,

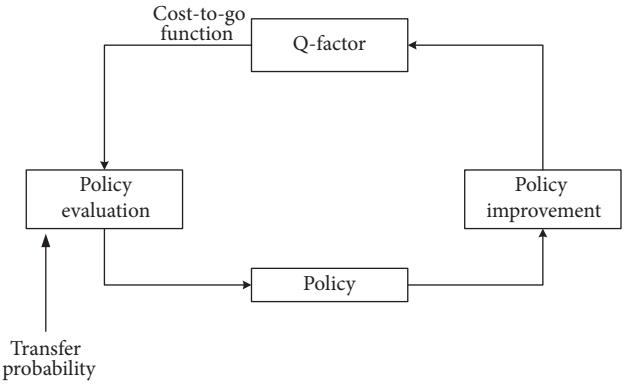

Figure 1: Illustration of policy iteration in reinforcement learning.

which can guarantee the system's security without repairing vulnerabilities or limiting services on hosts.

The other contribution of this paper is to render the defense strategy to be economical. Our method does not need scan or monitor hosts at all time. The hosts will be scanned only if our algorithm thinks it is not-trust node.

2. Preliminary

2.1. Description of Q-Learning Algorithm. Q-learning is a model-free reinforcement learning technique and it derives from policy iteration. The flow diagram of policy iteration is shown in Figure 1. Specifically, Q-learning can be used to find an optimal action-selection policy for any given (finite) Markov decision process (MDP). Therefore, the definition of Q-learning is given in

$$\begin{aligned} g_n &= (i_n, a_n, j_n) \\ s_n &= (i_n, a_n, j_n, g_n). \end{aligned} \tag{1}$$

This formula denotes that the state i_n transfers to $j_n = i_{n+1}$ using action a_n, and the cost of this process is g_n. Above, n represents the discrete time sequence.

In Section 2.1, Q-factor is considered as the sum of the immediate cost and all of the successor states' discount costs which are followed by the current state in policy μ.

However, different from policy iteration, Q-learning is an incremental dynamic programming process, and it is very suitable to solve MDP which does not have an apparent transfer probability. Furthermore, Q-learning works by learning an action-value function that ultimately gives the expected utility of taking a given action in a given state and following the optimal policy thereafter. A policy is a rule that the agent follows in selecting actions, given the state it is in. When such an action-value function is learned, the optimal policy can be constructed by simply selecting the action with the highest value in each state. Thus, the update formula of Q-factor is given in

$$\begin{aligned} J_n(j) &= \max_{b \in A} Q_n(j, b) \\ Q_{n+1}(i, a) &= (1 - \delta_n(i, a)) Q_n(i, a) \\ &\quad + \delta_n(i, a) [g(i, a, j) + \gamma J_n(j)]. \end{aligned} \tag{2}$$

j is the successor state and $\delta_n(i, a)$ is the learning rate of state-action (i, a) in time n. A is the action set. γ is the discount factor. Generally, in order to ensure the algorithm converges to the optimal value, the learning rate can be set based on

$$\delta_n = \frac{\alpha}{(\beta + n)}, \quad n = 1, 2, 3, \ldots. \tag{3}$$

In this formula, α, β are positive numbers. After we evaluated a series of values of α and β, we found that, if α/β is approximately 0.1, the convergence speed and the accuracy of the Q-learning agent are suitable.

But the traditional Q-learning algorithm owns some problems. Firstly, the Q-learning agent should choose an action when a state transfers to other state, but in the information system, we can fuse the action into state. Secondly, the traditional Q-learning may generate redundant terms, although the total reward of this path is the highest one. In Section 3, we will discuss and solve these problems.

2.2. Software-Defined Network. Software-defined networking (SDN) is an approach to computer networking that allows network administrators to programmatically initialize, control, change, and manage network behavior dynamically via open interfaces [13] and abstraction of lower-level functionality. SDN is meant to address the fact that the static architecture of traditional networks does not support the dynamic, scalable computing and storage needs of more modern computing environments such as data centers. This is done by decoupling or disassociating the system that makes decisions about where traffic is sent (the SDN controller or control plane) from the underlying systems that forward traffic to the selected destination (the data plane). The SDN structure is shown in Figure 2.

3. Building the Optimal Attack Path

The optimal attack path is the most cost-effective attack path from source node to target node in an information system network. Using the optimal attack path, the attacker can achieve his/her goal at minimum cost. In this paper, we use the optimal attack path to build the hidden forwarding path.

3.1. Network Model. A network model reflects host information in the target network, including software applications, host name, host's IP address and operating system, and communication link. Using a network model, we can produce a network topology graph and installed software applications configuration. In this paper, our network model refers to [14], but we have improved this model to suit our system. The structure of our network model is shown in Figure 3.

3.2. Evaluating Weights of the Hosts. In reinforcement learning, the reward function is the core facilitator. If and only if we have a reward function, the policy iteration or value iteration can be executed. Currently, two methods are available to obtain the reward function.

The first is to obtain the state set and the action set and determine the relationship between them. A reward

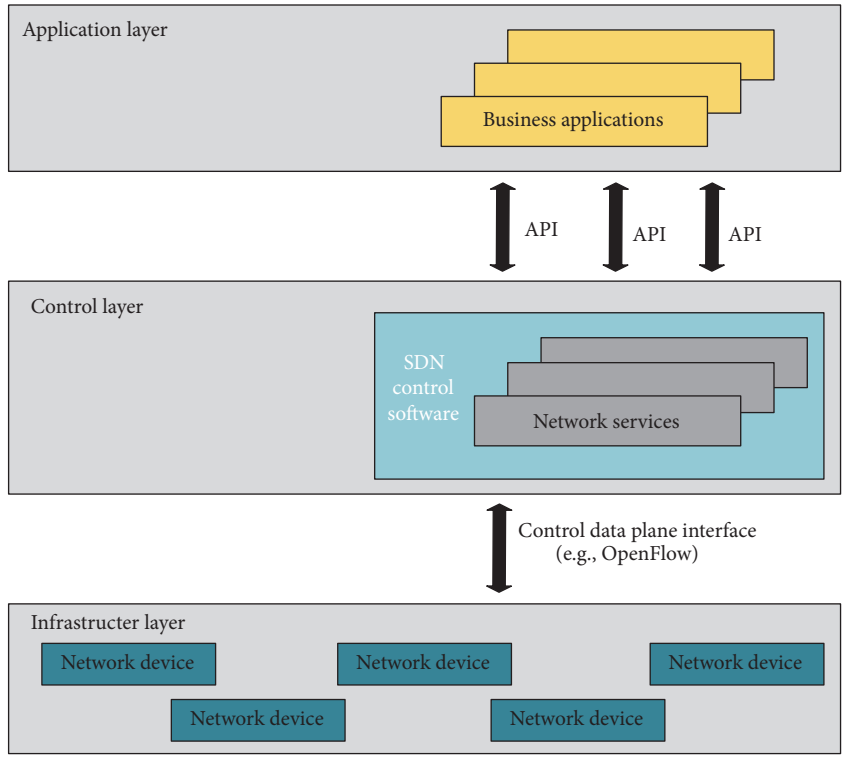

Figure 2: Model of software-defined network.

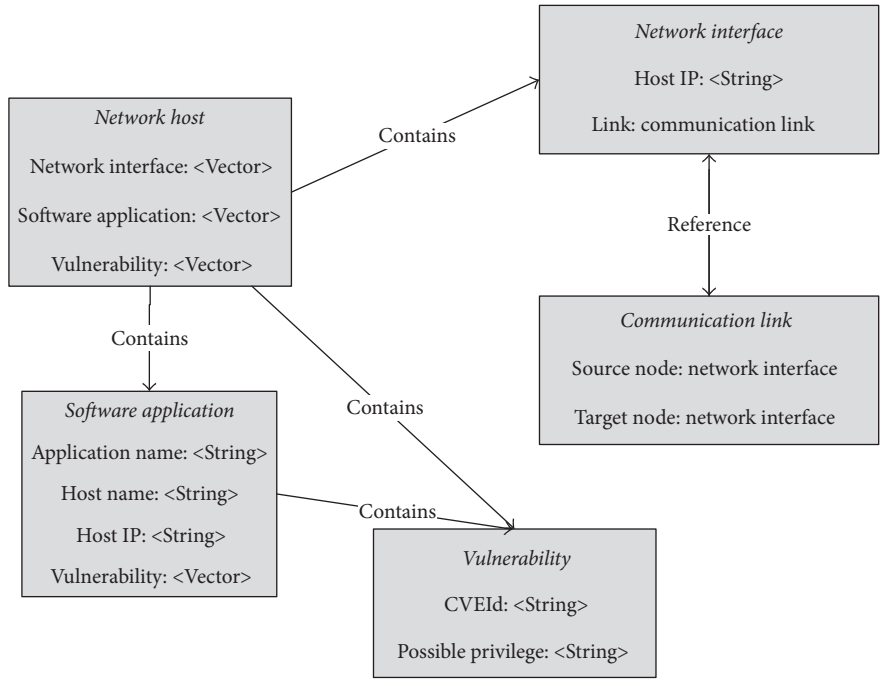

Figure 3: Network model.

function can then be fitted, which is named the state-action function or Q-factor. Furthermore, about several decades ago, another method that can also obtain the reward function was proposed, named inverse reinforcement learning [15, 16]. But to use the inverse reinforcement learning, the optimal policy should be specified beforehand. Thus, in this paper, we use the first method to obtain the reward function.

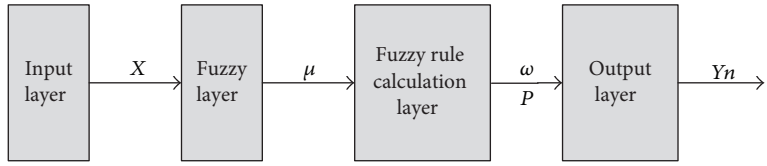

Figure 4: T-S fuzzy neural network structure.

Definition 1. Attack success rate of single weakness represents the rate at which a weakness in the information system is successfully exploited by attackers and the degree of difficulty in carrying out the attack. This element is denoted as Pr.

The attack success rate of a single weakness can be affected by a large number of factors, for instance, available information on the weakness, the attack method, the attack tool, and whether there is a communication link between the source host and target node. The level of detail known about a weakness will affect the attack success rate. The value of the success rate of a single weakness attack can be found from the Common Vulnerability Scoring System (CVSS) [17–19].

Definition 2. Host weight denotes a host's weight in the target network. It is the function defined by

$$\mathrm{Hw} = f(\mathrm{pos}, \mathrm{sev}). \tag{4}$$

In this equation, Hw is the host weight, pos represents the host's position (Internet, DMZ, Intranet), and sev denotes which services the host offers (MAIL, WEB, DNS, SQL, FTP, Management, etc.). Therefore, this equation indicates that host weight is decided by the host's position and the services it provides. Unfortunately, all services and position are not known, nor is the relationship between the position and a service. However, we can estimate the importance of these attributes based on basic knowledge of computer networks. For instance, the importance of different positions in an information system is Intranet > DMZ > Internet, and for the services, Management > SQL > FTP > DHCP > MAIL > DNS >= WEB. Furthermore, we know that pos and sev are positively correlated. Thus, we can build a fuzzy set based on this knowledge and use fuzzy computation to determine the value of the host weight. In this paper, we use a T-S fuzzy neural network (FNN) to fit the weight of host. The structure of the neural network is shown in Figure 4.

In this figure, X_i is an input vector. In this paper, in order to simplify the calculation and fit our network, it is denoted as a ten-element tuple <Pos_i, Pos_j, Pos_k, Sev_1, Sev_2, Sev_3, Sev_4, Sev_5, Sev_6, Sev_7>. pos and sev are the type of position and service, respectively. μ represents the membership function, ω is the result of the fuzzy computation, and P denotes the coefficient of the neural network. *Yn* is the neural network's output and is a single-element tuple <Weight>. *Weight* denotes the host weight. *Ye* is the expectation output. It represents the host weight. At first, we just categorize all hosts into five different degrees based on the impact of their confidentiality, integrity, and availability on the target network, *Danger, High, Medium, Low,* and *Very Low*, but these only reflect the weight of a host roughly. In the next step, we use our algorithm to fit more accurate results.

In this paper, the proposed host weight is computed based on a fuzzy self-adaption weight correction algorithm. In the first stage of the algorithm, we quantify the sample data. Because accurate results will be obtained by the fuzzy computation, we just need an initial value that reflects the importance of different attributes. We then normalize the sample and initialize the T-S fuzzy neural network, including the number of neurons in every layer, the initial learning rate and P, and the evolution times of the network. After training the network, for every input's attribute, $\widehat{\mu}$, a coefficient vector of the parameters in (4), which is defined in (5), can be obtained using the algorithm

$$\widehat{\mu} = \sum_j \mu_j, \quad j = 1, 2, \ldots, k. \tag{5}$$

In this equation, k is the number of inputs parameters. Thus, (4) can be redefined as

$$\mathrm{Hw} = \widehat{\mu} X_t. \tag{6}$$

X_t is the set of pos, sev in the target network. Therefore, the host weight can be solved from (6).

3.3. Modeling Optimal Attack Path Using Improved Q-Learning Algorithm. After we obtain the information on vulnerabilities and hosts in the target network, the process of Q-learning begins.

Definition 3. Optimal attack path is an attack path which has the highest probability to be chosen by intruders.

Definition 4. The not-trust nodes are the nodes in the optimal attack path.

After we acquire the state set and action set, if we can fit the reward function (cost function), the optimal attack path can be determined.

3.3.1. The Proposed Reward Function. The reward function (cost-to-go function) decides which action will be chosen in the current state. Using the reward function, we can obtain an optimal policy, which is the optimal attack path. Because a state node represents a host-vulnerability pair, the information of host and vulnerability will decide the reward function. In this paper we formulate the reward function based on host weight (Hw) and the attack success rate on

Require:
(1) Host weight Hw
(2) The attack success rate of vulnerability Pr
(3) Vulnerability V
(4) Host Name HN
Ensure: Optimal policy (attack path) π^*
(5) **function** IQL(Hw, Pr, V, HN)
(6) $n \leftarrow size(V)$ obtain the number of vulnerabilities
(7) $m \leftarrow size(\text{HN})$ obtain the number of hosts
(8) $S \leftarrow getstate(V, \text{HN})$ gain state set
(9) $\gamma \leftarrow getNumber()$ initialize discount factor
(10) $initialQ = 0$ initialize value matrix
(11) $Reward \leftarrow \mathbf{R}(S, \text{HN}, \text{Hw}, S \cdot V \cdot P_r)$ build reward matrix
(12) **for** $t = 1 : n$ n is iteration step **do** $Q_{t+1}(i_{t+1}) \leftarrow$ **funcQ**$(Q_t(i_t), \delta_t, \gamma, R(i_t, i_{t+1}), J_t(i_{t+1}))$ obtain the optimal policy
(13) **if** $Q_{t+1}(i) = Q_t(i)$ **then** break
(14) **end if**
(15) **end for** $\pi^* \leftarrow i_0, i_1, \ldots, i_n$
(16) **return** π^*
(17) **end function**

Algorithm 1: Planning optimal attack path based on improved Q-learning algorithm.

a single weakness (Pr). Therefore, the reward function is defined by

$$R = f(\text{Hw}, \text{Pr}). \tag{7}$$

From this equation, we know that the reward function is decided by Hw and Pr, but the relationship between the host weight and attack success rate of a single weakness is not known. Generally, there is a positive relationship between Hw and Pr, and they are interdependent. Thus, we use a multiplication sign to connect Hw and Pr. We should also consider the impact of a host on the information system network (see the intermediate item in (8)). For the relationship between attack success rate and host weight, we believe the host weight is less important than the attack success rate, because the calculation aims to vulnerabilities; thus, the square root of Pr is used in (8), which ensures the variation range of the value of attack success rate is bigger than host weights. The reward function is rewritten as

$$R = \sqrt{P_r} \times \frac{\text{Hw}_i}{\sum_j \text{Hw}_j} \times \overline{\sum_m T_m^i}. \tag{8}$$

In this equation, Hw_i is the weight of host H_i which owns a specific vulnerability. T_m^i denotes the mth privilege which can be captured on H_i based on a specific vulnerability. Hw_j is the weight of a host in the target network. c is the total number of hosts in the target network.

3.3.2. Improved Q-Learning Algorithm Structure. After obtaining the reward function, we determine g_n in (2) and calculate the Q-factor. In this paper, we use deterministic state transition model. Our contribution is that we merge actions into state set, because we actions and states are bounder together. Thus, the dimension of state transition matrix in Q-learning is reduced, that means the space complexity of improved Q-learning is lower than traditional Q-learning algorithm. In addition, to avoid the attack circle, we set P_r to a small number if the target host is the source host; thus the reward matrix is recalculated based on the new P_r. Using this Q-factor, the optimal policy/attack path from source node to target node can be acquired. The improved Q-learning algorithm to identify the optimal attack path is shown in Algorithm 1.

In this algorithm, i_t represents a state in time n. In this paper, a state is a two-element tuple <HostName, Vulnerability>. Because the states in the Q-learning system are host-vulnerability pairs, attaining the optimal policy means obtaining a set of host-vulnerability pairs that define the optimal attack path.

4. Building Dynamic Forwarding Paths in a SDN Environment

Nowadays, for an information system network, the most common defense strategy is to divide the target network into Intranet and DMZ. In the Intranet, some hosts are prohibited to visit other hosts or services, but the restrictions are very inconvenient for users. If the administrators find new vulnerabilities in a target network, some services or applications must be stopped to repair these vulnerabilities, which is uneconomic. Besides, if firewalls and/or IDS do not have a specific property, they cannot defend from specific attacks. To solve these problems, we propose the hidden forwarding path in this paper.

4.1. Building Hidden Forwarding Paths Using SDN. Using SDN, we can build forwarding nodes arbitrarily. In Figure 5(a), all hosts in the target network can connect with each other. In this case, the security of the target network cannot be very high because the forwarding

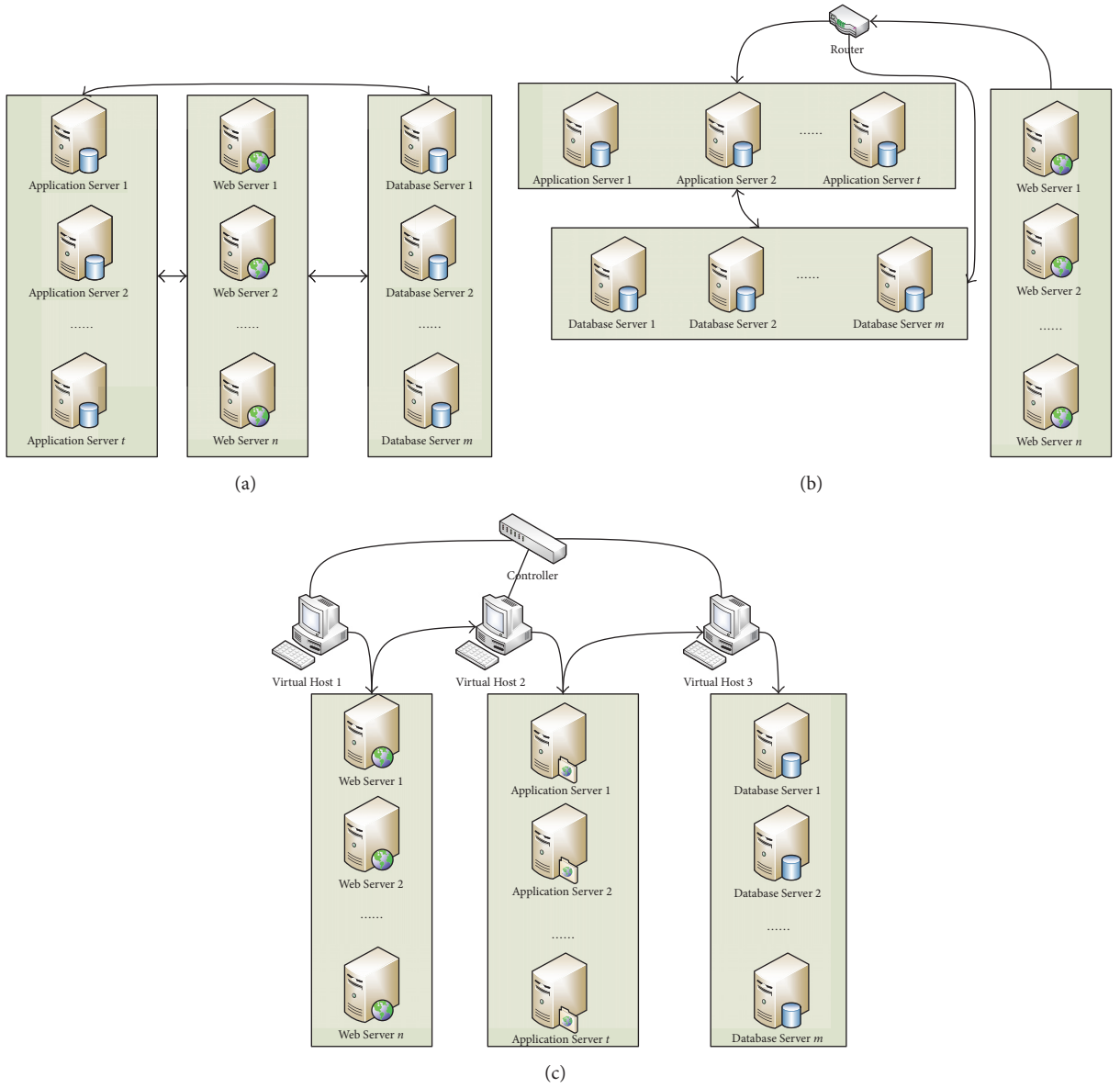

FIGURE 5: (a) An information system network's forwarding structure without any router. (b) The forwarding structure using traditional three-layer switching. (c) The forwarding path using SDN.

structure is too complex to monitor. To solve this issue, traditional networks use routers and switches to control the forwarding path (see Figure 5(b)). Generally, the core switch divides the information system network into two zones, DMZ and Intranet. If we change the forwarding path using the optimal attack path, the network structure should be altered. For example, if the web server visits application server, the application server responds to the requests of the web server by accessing data in the database server. If the application server is located in optimal attack path, the initial routing path should be changed or prohibited to protect system's security; for example, theoretically speaking, the web server can send packets to other servers, and these severs will visit database server. But this is unpractical scheme, because in traditional network, part of network topology is fixed. Thus, we cannot modify the forwarding path directly, meaning we cannot avoid the not-trust nodes without repairing vulnerabilities or limiting some services. In this paper, we use SDN to create the specific virtual hosts being used as forwarding nodes. In the first step, according to the not-trust nodes, we create virtual hosts that offer specific services, defense strategies, or interfaces to keep the original network structure in SDN environments. A sample of a hidden forwarding path is shown in Figure 5(c). In this network, the optimal attack path is

$$\text{WebServer} \longrightarrow \text{ApplicationServer} \longrightarrow \text{DatabaseServer}. \quad (9)$$

According to this attack path, we can add virtual hosts to be forwarding nodes between each two nodes and specific

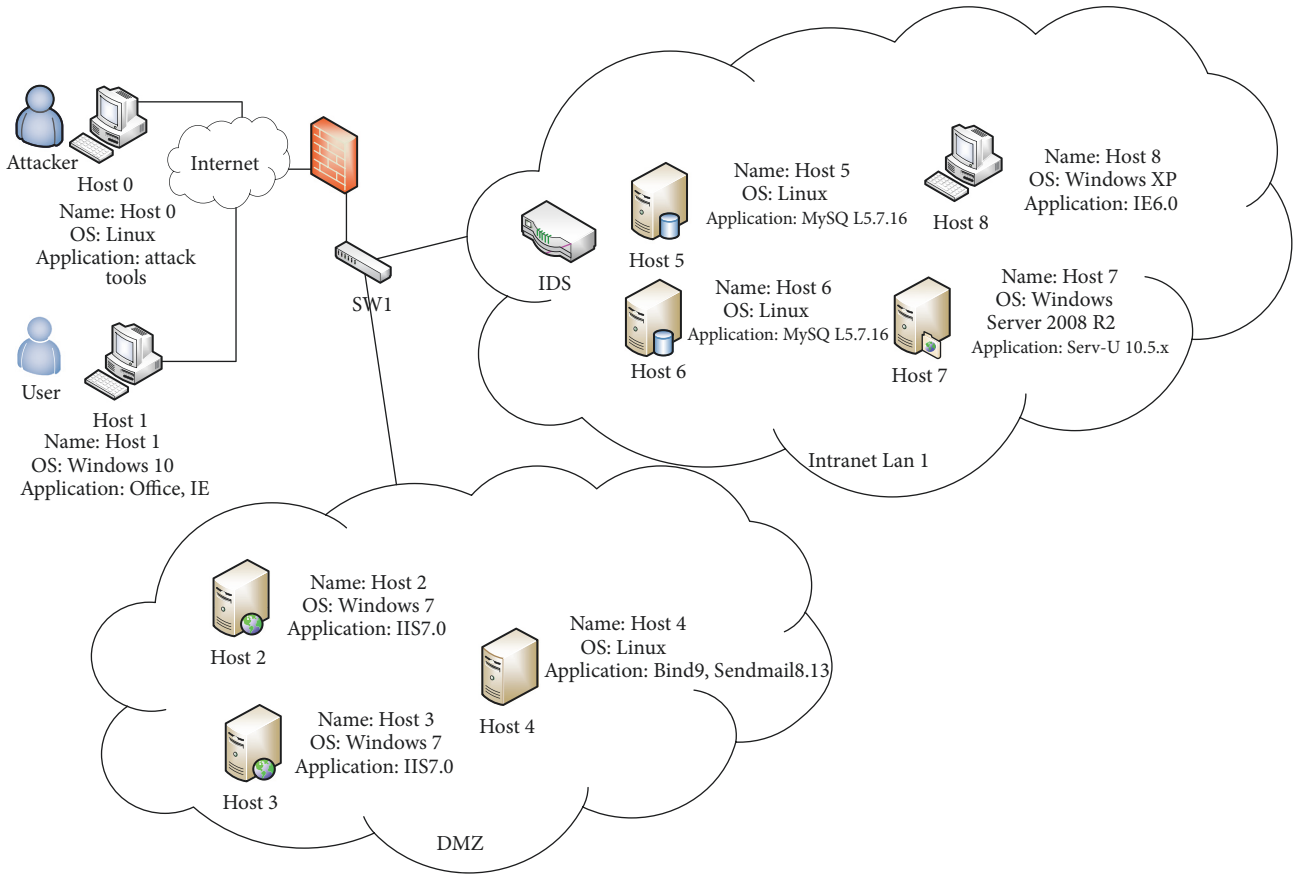

Figure 6: Network topology graph.

security strategies can be enacted on these virtual hosts, and the hidden forwarding path is shown as follows:

$$\begin{aligned}&(\text{VirtualHost1}\cdots\rightarrow)\ \text{WebServer} \longrightarrow (\text{VirtualHost2}\cdots\rightarrow)\\ &\text{ApplicationServer} \longrightarrow (\text{VirtualHost3}\cdots\rightarrow)\ \text{DatabaseServer}\end{aligned} \tag{10}$$

In virtual Host 1, we just analyze the packets that represent an HTTP request, and the function of Host 2 and Host 3 is similar to Host 1. Thus, every virtual host can defend against specific attacks, which is more effective and cost-effective than placing firewalls between every not-trust node. Concretely, in Figure 5(b), if the web server wants to communicate with application server, it will send the request packets to router and the router will send packets to application server through its routing table. But in Figure 5(c), if the web server sends packets, they will be sent to virtual Host 2 first. Then, virtual Host 2 will send packets application server. For the defense strategy in this optimal attack path, we should add filters that can filter attacks aimed at web, application, and database servers. We have two methods to add filters. First, we add web, ftp, and database filters into the first node (virtual Host 1). Second, because different servers are divided into different blocks in SDN environments, we can add specific filters on corresponding virtual hosts. In the first method, we must change the defense strategy if the optimal attack path changes. In the second method, we just need to add or delete corresponding virtual hosts in SDN environments, which is much more convenient and efficient than the first method.

After implementing these specific functions on the virtual hosts, they form the hidden forwarding path. We can easily monitor the security logs of these virtual hosts. Furthermore, because the route is computed by the controller, we can change the hidden forwarding path dynamically and update the mapping table in the DNS server, which will not influence the physical components of the information system network. By using this method, administrators need not repair known and 0-day vulnerabilities in the target network immediately.

5. Experiment

In this experiment, we offer an example of building a hidden forwarding path. The network topology graph which is shown in Figure 6 is based on the network model mentioned in Section 2.1. Some security policies are implemented in this network: the firewall divides the information system network into three zones which represent Internet, DMZ, and Intranet, respectively. The web servers are configured in DMZ to offer web service to users. Hosts in the Intranet are not allowed to access the Internet. In addition, there is an intrusion detection system (IDS) in the Intranet to supervise

Table 1: Information of software and vulnerabilities on the terminals in the target network.

Host	Network segment	Service	CVE number	Success rate
H_0	Internet	Attack Tools	None	0
H_1	Internet	Office	None	0
H_2	DMZ	IIS7.0(HTTP)	CVE-2015-1635	0.9
H_3	DMZ	IIS7.0(HTTP)	CVE-2015-1635	0.9
H_4	DMZ	BIND9(DNS)	CVE-2015-5477	0.6
		Sendmail(Mail)	CVE-2009-4565	0.5
		OpenSSH 5.4(SSH)	CVE-2016-0778	0.3
H_5	Intranet	OpenSSH 5.4(SSH)	CVE-2016-0778	0.3
		MySQL 5.7.12(SQL)	CVE-2016-3521	0.6
			CVE-2016-3614	0.3
H_6	Intranet	MySQL 5.7.12(SQL)	CVE-2016-3521	0.6
			CVE-2016-3614	0.3
H_7	Intranet	Serv-U 10.5.0.19(FTP)	CVE-2011-4800	0.7
H_8	Intranet	Outlook(Mail)	CVE-2015-6172	0.4
		System	CVE-2003-0352	0.6

Table 2: Grade of pos and sev in host weight.

	Internet	DMZ	Intranet	Web	DNS	MAIL	DHCP	SQL	FTP	Manager
Initial value	1	2	3	1	2	3	4	5	6	7
Coefficient	0.279	0.362	0.363	0.345	0.378	0.337	0.355	0.375	0.399	0.384
Result	0.28	0.72	1.09	0.25	0.76	1.01	1.42	1.88	2.39	2.69

the entire target network. Also, Internet users can only access the IIS web services on Host 2 and Host 3 and the domain name service on Host 4. At the same time, Host 2 and Host 3 can access Host 4's Sendmail service and the SQL service on Host 5 and Host 6. Host 7 is a FTP server which can also be accessed by Host 2 and Host 3. But Host 2, Host 3, and Host 4 are prohibited from accessing Host 8 (Intranet management terminal) directly. Host 8 can access and download data from Hosts 2 to 7. The software applications and vulnerabilities on every terminal are shown in Table 1.

In this paper, we extract 3000 fuzzy information items from the hosts to build the sample data. Because we only need the initial value of the input's attributes (X in Section 3.2) to represent the importance of different attributes, if a host does not offer a service or is not installed in the position, the attribute's value is set to 0. In contrast, if the host offers a service or is installed in the position, the value of the attribute is set to a number based on the importance of the attribute. Initially, an attribute with least importance is set to 1, and other attributes' values will in turn increment at an interval of 1 based on the least important attribute's value. Thus, the initial values of sev and pos are shown in the first line in Table 2.

Based on this table, we obtain the value of input data. The neural network's output is the host weight. Because the expectation output (Ye) only reflects the host weight roughly, it is only necessary for the network's output to display a similar trend to the expectation results. Thus, the result of testing instance prediction is shown in Figure 7.

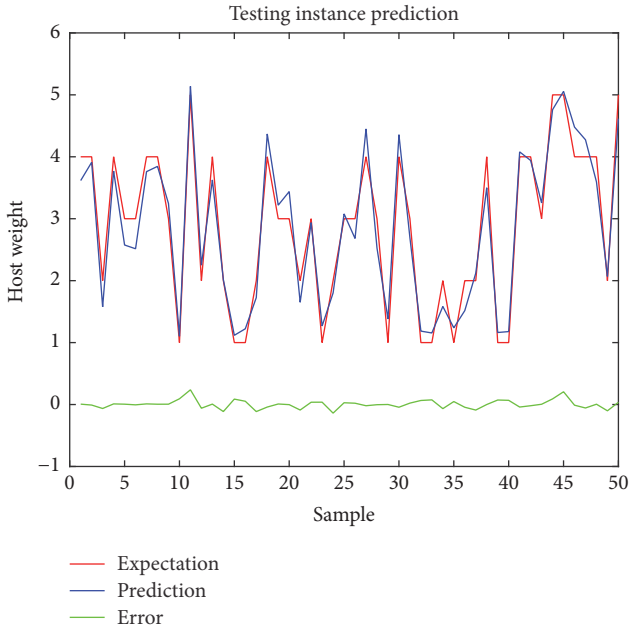

Figure 7: Results of training instance of host weight using FNN.

From Figure 7, the prediction of testing instance follows the same trend as the expectation result of testing instance. If we round off the prediction result (Figure 8), the prediction is equal to the expectation. Therefore, the parameters trained by the fuzzy neural network are assumed to be correct.

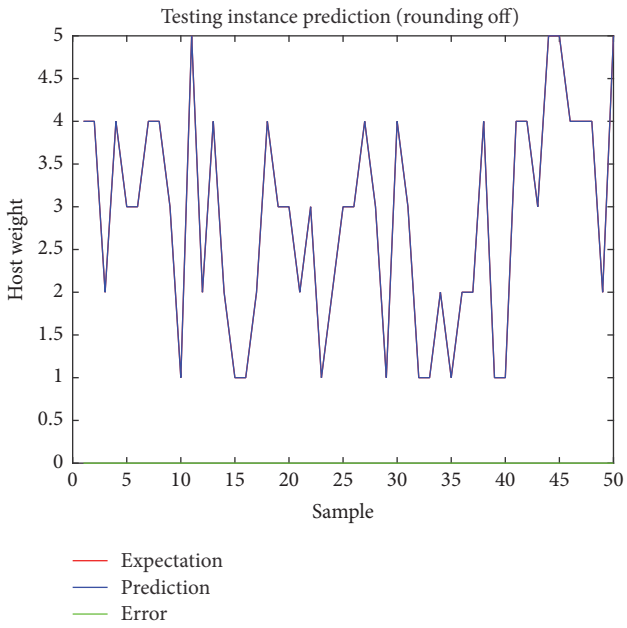

Figure 8: Results of training instance of host weight using FNN (round off).

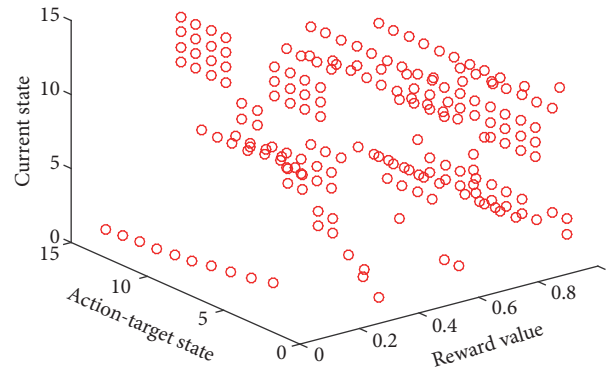

Figure 9: Reward matrix of improved Q-learning algorithm.

The coefficients and the grading standards of pos and sev defined in (4) are shown in the second and third lines in Table 2. In this table, the first line shows the membership of every pos and sev, and the second shows every element's factor.

According to Table 2, the initial reward matrix is obtained and is shown in Figure 9.

In this paper, we set Pr to 0.1 if the target host is the source host and the reward matrix is recalculated based on the new P_r. Besides, we assume that the attackers' goal is to destroy the database of the target network; thus the optimal attack path is

$$H_{1,\text{Root}} \longrightarrow \text{CVE} - 2015 - 1635 \longrightarrow H_{3,\text{Root}} \longrightarrow \text{CVE} - 2011 - 4800 \longrightarrow H_{7,\text{Root}}. \quad (11)$$

Thus, we can find the not-trust nodes. Based on these nodes' information and SDN, the hidden forwarding path can be built. Thus, the new network topology is shown in Figure 10.

5.1. Testing Results. In this section, we test the performance of the hidden forwarding path (Figure 10) and the traditional forwarding path (Figure 5) in a real information system network. We assume that the firewall cannot prevent the attacker. We use CVE − 1025 − 1635 and CVE − 2011 − 4800 to be the test vulnerabilities, because they have the highest attack success rate in the DMZ and Intranet, respectively. The test results are shown in Table 3.

In Table 3, the reason for "Maybe" in the third line is that the firewall and IDS may not provide an effective defense against a specific type of attack, that is, a potentially dangerous function. But in the hidden forwarding path, the defense strategy is designed to ensure that the suspicious packet is filtered by the virtual host. Furthermore, because we implement fewer defense strategies on a virtual host, the cost of the operation will not be excessive. On the other hand, adding some virtual forwarding nodes in the target network will add memory utilization and CPU utilization in the target network, although the total cost of this defense strategy is much lower than stopping some servers until the system vulnerabilities are repaired.

Besides, we also offer the optimal attack path using traditional Q-learning algorithm:

$$H_{1,\text{Root}} \longrightarrow \text{CVE} - 2015 - 1635 \longrightarrow H_{3,\text{Root}} \longrightarrow \text{CVE} - 2015 - 1635 \longrightarrow H_{3,\text{User}} \longrightarrow \text{CVE} - 2011 - 4800 \longrightarrow H_{7,\text{Root}}. \quad (12)$$

Comparing the optimal attack path obtained from improved Q-learning algorithm with the path gained from traditional Q-learning algorithm, we can see that the result of improved Q-learning algorithm is terser than traditional Q-learning.

We also compare our method with other defense strategies. The results are shown in Table 4. The computing method of value of "*Cost*" of an algorithm refers to security metrics and defense cost mentioned in [6]. The "*Cost*" consists of two parts: one is defense cost, and the other is attack cost. The "*Complexity*" and "*Cost*" determine "*Performance*."

In Table 4, IQL is the method proposed in this paper, TQL is traditional Q-learning algorithm, and DG-HMM is the algorithm proposed in [6]. From this table, we can find that

Table 3: Test result of attack target node.

Network structure	Vulnerability name	Attack vector	Filter or not	Attack successfully or not
Original routing path	CVE-1025-1635	Network	No	Yes
Hidden routing path	CVE-2015-1635	Network	Yes	No
Original routing path	CVE-2011-4800	Network	Maybe	Maybe
Hidden routing path	CVE-2011-4800	Network	Yes	Not

Table 4: Result of comparing test.

Method	Time complexity	Space complexity	Cost	Performance
IQL	$O(e * N)$	$O(N^2)$	(10.4, 29)	High
TQL	$O(e * N)$	$O(N^3)$	(10.4, 29)	Middle
AG-HMM	$O(T * N^2)$	$O(N^2)$	(14.3, 20.9)	Middle

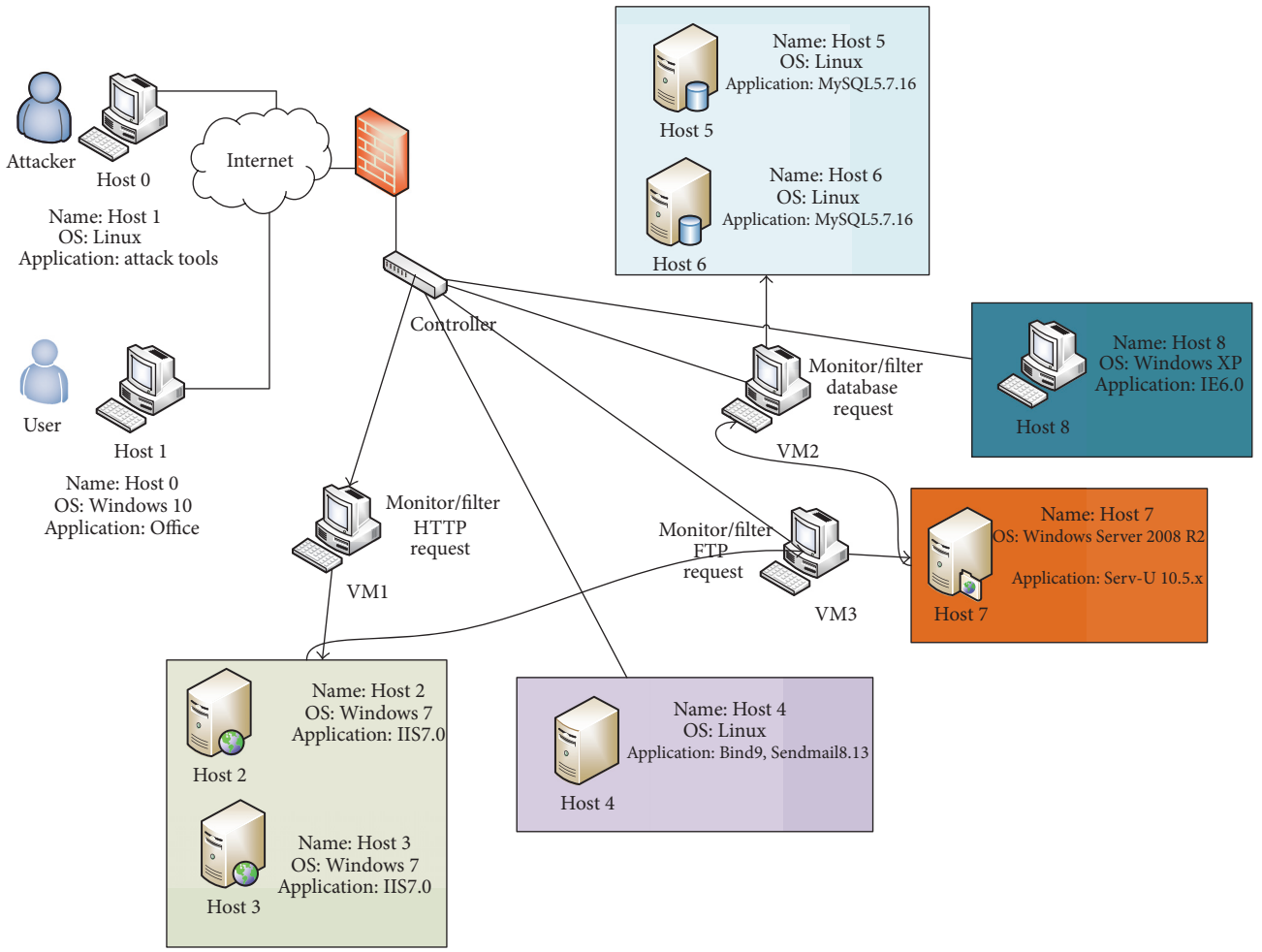

Figure 10: Implementing hidden routing path.

our approach is superior to the other two methods, whether in time complexity or at cost.

6. Conclusion

In this paper, we propose a dynamic hidden forwarding path planning method using improved Q-learning (IQL). The IQL improves running speed of Q-learning agent. Using IQL, the optimal attack path can be obtained quickly and does not rely on the complete or minimal attack graph. The defense strategy obtained from optimal attack path, hidden forwarding path, can be implemented in many different kinds of network environments. SDN can allocate the bandwidth based on the network traffic condition, that means although a large number of packets may be sent to one virtual host, these data will not trigger a DoS attack indirectly.

In the next stage, we aim to optimize the cost function to improve the speed of convergence. In addition, we are going to implement this method in a traditional office network without triggering a DoS attack.

Conflicts of Interest

There are not any conflicts of interest related to this paper.

Acknowledgments

This work is supported by funding from the Basic Scientific Research Program of Chinese Ministry of Industry and Information Technology (Grant no. JCKY2016602B001).

References

[1] R. Ritchey, B. O'Berry, and S. Noel, "Representing TCP/IP connectivity for topological analysis of network security," in *Proceedings of the 18th Annual Computer Security Application*, 2002.

[2] R. W. Ritchey and P. Ammann, "Using model checking to analyze network vulnerabilities," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 156–165, IEEE, May 2000.

[3] O. Sheynar, *Scenario Graphs and Attack Graphs [Ph.D. thesis]*, Carnegie Mellon University, Pittsburgh, Penn, USA, 2004.

[4] N. Ghosh, S. Nanda, and S. K. Ghosh, "An ACO Based Approach for Detection of an Optimal Attack Path in a Dynamic Environment," in *Proceedings of the 11th International Conference Distributed Computing and Networking*.

[5] S. Noel, S. Jajodia, B. O'Berry, and M. Jacobs, "Efficient minimum-cost network hardening via exploit dependency graphs," in *Proceedings of the 19th Annual Computer Security Applications Conference, ACSAC 2003*, pp. 86–95, December 2003.

[6] S. Wang, Z. Zhang, and Y. Kadobayashi, "Exploring attack graph for cost-benefit security hardening: a probabilistic approach," *Computers & Security*, vol. 32, pp. 158–169, 2013.

[7] S. Akbar, J. A. Chandulal, K. Nageswara Rao, and G. Sudheer Kumar, "Improving network security using machine learning techniques," in *Proceedings of the 2012 3rd IEEE International Conference on Computational Intelligence and Computing Research, ICCIC 2012*, December 2012.

[8] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, The MIT Press, Cambridge, Mass, USA, 2016.

[9] Natl Institute of Standards and Technology, *National Vulnerability Database Version 2.2*, 2008, http://www.nvd.org.

[10] Common vulnerabilities and exposures. http://cve.mitre.org/.

[11] P. Hu, H. Li, H. Fu, D. Cansever, and P. Mohapatra, "Dynamic defense strategy against advanced persistent threat with insiders," in *Proceedings of the 34th IEEE Annual Conference on Computer Communications and Networks, IEEE INFOCOM 2015*, pp. 747–755, May 2015.

[12] L. Wang, S. Jajodia, A. Singhal, P. Cheng, and S. Noel, "K-zero day safety: A network security metric for measuring the risk of unknown vulnerabilities," *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 1, pp. 30–44, 2014.

[13] Software Defined Networking (SDN): Layers and Architecture Terminology. https://www.rfc-editor.org/rfc/rfc7426.txt.

[14] Y. Chen, K. Lv, and C. Hu, "Optimal Attack Path Generation Based on Supervised Kohonen Neural Network," *Network and System Security*, pp. 399–412, 2017.

[15] K. Shiarlis, J. Messias, and S. Whiteson, "Inverse reinforcement learning from failure," in *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2016*, pp. 1060–1068, May 2016.

[16] A. Y. Ng and S. J. Russell, "Algorithms for Inverse Reinforcement," in *Proceedings of the 17th International Conference on Machine Learning*, pp. 663–670, 2000.

[17] Common vulnerability scoring system. https://www.first.org/cvss.

[18] Y. Zhang, X. Yun, and M. Hu, "Research on privilege-escalating based vulnerability taxonomy with multidimensional quatitative attribute," *Journal of China Institute of Communications (in Chinese with English abstract)*, pp. 107–114, 2004.

[19] G. Stoneburner, A. Goguen, and A. Feringa, "Risk management guide for information technology systems :," National Institute of Standards and Technology NIST SP 800-30, 2002.