Openflow Table Decision Method under Mimic Defense

Rui Lei*, Chuanhuang Li and Zhongyun Tang
School of Information and Electrical Engineering, Zhejiang Gongshang University
Hangzhou, China

*Email: 1025926085@qq.com

Abstract. SDN is an implementation of network virtualization. It implements flexible control of network traffic by separating control and forwarding of network equipment. However, there are security issues in SDN networks. If the controller is hijacked, an attacker can manipulate the entire network and tamper with and intercept network traffic at will, causing serious problems such as network paralysis. Based on the overall architecture of the mimic controller, this paper proposes a controller flow table decision method for the key decision module. Finally, the decision algorithm is developed and implemented, and the attack defense verification is performed. The flow table decision method in this paper greatly improves the decision efficiency, and successfully implements the random active defense of the controller's flow table.

1. Introduction
In recent years, technologies such as cloud computing and the Internet of Things have developed rapidly. The traditional network architecture can no longer meet the increasing network demand. SDN, as a new type of network architecture, dynamically and centrally schedules the path of traffic by separating the control and forwarding of network equipment, and realize intelligent management of traffic.

However, SDN also brings new security issues. The controller is the core component of the SDN architecture. It controls the forwarding plane to perform various network functions just like the brain. If the controller is hijacked, an attacker can manipulate the entire network, and causing serious problems such as network paralysis. Mimic defense technology introduces the redundancy and heterogeneity of the actuators to change the unity and similarity of the system, and uses non-similar margin space blocking to achieve the requirement of controllable system security risks.

Based on the overall structure of the mimic controller, this paper proposes a decision method for the controller flow table for the key decider module. The decision method is modeled based on the characteristics of the controller flow table. Finally, the decision algorithm is developed and implemented, and the attack defense verification is performed. The flow table decision method in this paper greatly improves the decision efficiency, and successfully implements the random active defense of the flow table issued by the controller.

2. Research Status

2.1. Status of Controller Security Research
Now, there are a lot of discussions on the security of SDN controller in industry and academia. There
are two directions in solving the problem of controller safety. One is the conservative expansion of the existing controller, and the other is the reformed design of the new controller.

A conservative way to patch security issues such as existing controller vulnerabilities, and add new security mechanisms to eliminate discovered security issues. For example, researchers have upgraded the existing controllers and designed controllers such as FortNOX [1], SE-Floodlight [2], and OFX [3].

Based on the open source NOX controller [4], Porras et al. Designed a FortNOX controller with a safety detection mechanism. Security policies such as role authentication, flow table rule conflict detection, and policy synchronization are implemented to improve the security of the controller. Conservatism increases the security of the controller to a certain extent, but does not redevelop the controller architecture.

The revolutionary idea is to design a new controller, and add safety functions to the core design of the controller at the beginning of the controller design. The controllers currently represented include PANE [5] and Rosemary [6].

PANE is a new type of controller designed by researchers such as Ferguson to resolve conflicting requests between untrusted users. Policy tree [7], and three types of data structure modules of the network information base are introduced in PANE to Achieve goals such as conflict resolution, and message fault tolerance. In addition to the Openflow protocol, MPLS [8], 4D [8] can also be used to control the SDN data layer.

There are more and more measures to improve the security performance of the controller, but it still does not fundamentally solve the security problems faced by SDN controllers.

2.2. Research Status of Mimic Security Defense
Mimic defense is an initiative defense idea pioneered by Academician Wu. Mimic defense is to dynamically and randomly select heterogeneous executors in the system, so that the entire system maintains diversity, dynamics and randomness, which interferes with the attacker's acquisition of system information, making it impossible to accurately grasp the information Vulnerabilities, increase the work required to break the system to achieve the purpose of security defense.

In the field of mimic security defense, the paper[9] proposed a mimic defense web server based on a dynamic heterogeneous redundant structure. The dynamic and heterogeneous collaborative work disturbs the feedback information of the attack, and the probability of a successful single-step attack has been greatly reduced, the randomness and uncertainty of the system have been improved to a certain extent. The experimental results show that the mimic defense model can effectively improve the security of the system.

In order to defend against malicious attacks against cloud workflows, the paper[10] proposed a mimic workflow execution system based on a "dynamic heterogeneous redundancy" structure. This system employs a dynamic workflow execution environment switching strategy, which improves the heterogeneity and difficulty of being attacked, as well as the security of the system effectively.

The paper[11] addresses the potential safety hazards in the current software-defined network control layer. Based on the idea of mimic defense, a mimic security controller model is proposed. The scheduler's scheduling mechanism is used to implement controller scheduling to improve system dynamics. The experimental results show that the mimic controller architecture can increase the difficulty of the attacker to control the system and change the system data processing. The dynamic scheduling of the scheduler greatly improves the randomness of the system to the attacker, and the security and reliability of the system is greatly increase in amplitude.

3. The overall architecture of the mimic controller

3.1. Dynamic heterogeneous redundancy model
The dynamic heterogeneous redundancy model[12] is the basic model of mimic defense. As shown in figure 1 below, the scheduler copies the same input into M copies and sends these M messages to the
heterogeneous controller execution set. The M executors in the construction executor set process the message, send the result to the decider, get the only correct result through the decision of the decider, and the scheduler generates a new heterogeneous executor set according to the feedback message to replace the current set.

**Figure 1. Dynamic heterogeneous redundancy model**

### 3.2. Mimic controller architecture

According to the mimic defense idea proposed by Academician Wu, this paper designed a mimic defense architecture based on a dynamic heterogeneous redundancy model. The overall architecture is shown in figure 2. The main functional modules of the architecture are divided into the following parts:

**Scheduler:** Responsible for receiving and delivering information, and dynamically selecting multiple controller executor from the controller execution pool according to the scheduling algorithm to form a heterogeneous controller execution set for work. And multiple copies of the issued information are sent to the heterogeneous controller execution set. Finally, the final judgment result is used to realize the abnormality detection, and the controller executor with the abnormality is fed back to the scheduler, and the scheduler cleans the controller execution body with the abnormality.

**Controller executors pool:** The controller executors pool is the functional core of the mimic defense architecture. It consists of M controller executors with different operating systems and controllers and the underlying hardware architecture.

**Decider:** The decider judges the flow table sent by the controller executor according to the deciding algorithm, and obtains the only correct result. Then, randomly select a controller among the normal controllers to access the underlying switch, and issue the correct flow table. Finally, the abnormal controller is fed back to the scheduler.

**Figure 2. Overall architecture diagram**

In this mimic defense, an attacker must successfully attack most of the controllers at the same time in order to issue the tampered flow table, so as to achieve complete control of the entire system. To achieve this goal, the attacker needs to cost multiple controllers at the same time.

### 4. Decision method of controller flow table

#### 4.1. Decision model

In this paper, a decider is used to judge the output results of each execution body in the execution body set. We use \( u = (u_1, u_2, \ldots, u_L) \) to indicate that a normal executor will really execute the user request correctly. \( v = (v_1, v_2, \ldots, v_L) \) represents the original execution result of an executor's request to the user.
\( w = (w_1, w_2, L, w_i) \) indicates that the arbiter actually received the execution result of an executor on a user request. In general, under normal circumstances, for \( \forall i (1 \leq i \leq t) \), we have \( v_i = w_i \). However, if the executor is attacked by an attacker, becomes an abnormal execution body, or the execution result is tampered with by the attacker during the process of sending to the arbiter, it may cause \( j (1 \leq j \leq t) \) and \( v_j \neq w_j \) to exist. Suppose that the probability of this result caused by the attacked executor is:

\[
P_s = P(u \neq w)
\]

Exist \( i (1 \leq i \leq t) \):

\[
P_i = P(u_i \neq w_i) = P(u \neq w)
\]

Therefore, the task of the decider is to determine a relatively correct execution result from those execution results that may be tampered with by the attacker. Through analysis, it can be seen that the task of the decider can be transformed into solving the following formula. \( v^* \) represents the final output of the decider.

\[
v^* = \text{arg max}_v P(v | w)
\]

Because the executor in the executor set are mutually heterogeneous, it can be considered that the execution results of the user requests between each executor and each executor are also independent of each other, which can be obtained by Bayes’ theorem:

\[
v^* = \text{arg max}_v P(w | v)
\]

Use the -1/1 sequence of \( b_s = (b_1, b_2, L, b_n) \) to indicate the status of each executor in the executor set, if \( b_s = 1 \), indicates that the s-th executor is normal, if \( b_s = -1 \), indicates that the s-th executor is abnormal.

It can be inferred

\[
b_s = \begin{cases} 
1 & \text{normal executor} \\
-1 & \text{abnormal executor}
\end{cases}
\]

Suppose \( P(b_s) \) represents the probability that the state of each executor in the current executor set is the above sequence, then:

\[
v^* = \text{arg max}_v \sum_{b_s} P(w | b_s)P(b_s)
\]

Because the solution of the above formula is complicated. Therefore, this article intends to split the above formula and solve the optimal solution of each field of the execution result of the execution body separately, and decompose \( v^* \) into multiple segments, and then find the optimal solution for each field.

\[
v^* = (v_1^*, v_2^*, L, v^*_t)
\]

The following formula is obtained:

\[
v^*_t = \text{arg max}_v \sum_{b_s} P(w | b_s)P(b_s)
\]

After calculating the optimal solution of each field of the execution result of the executor, the optimal solution can be obtained by combining. In addition, this paper also considers the system error \( \alpha \) of each executor in the execution set. Let B be used to describe the probability of a system error in the executor itself. Therefore, the probability that the execution result of a normal executor that is actually sent to the decider is different from the true correct result is:

\[
P(w' \neq v | b_i = 1) = P(u_i' \neq v_i \vee u_i = w_i' | b_i = 1)\alpha = P((u_i' \neq v_i \vee u_i = w_i') | b_i = 1)
\]

The probability that the execution result of the abnormal executor actually sent to the decider is different from the true correct result is:
\[P(w' \neq v \mid b_i = -1) = P((u_i' \neq v \land u_i' = w_i') \mid b_i = -1) +
\]
\[= \alpha(1 - P_i) + (1 - \alpha)P_i \tag{7}\]

From the above formula, it can be known that the cause of the inconsistent execution results of the abnormal executor may be due to the systematic error of the executor itself or the attacker’s attack, while the normal executor may only be caused by the systematic error, so:

\[v^* = \arg \max_v \sum_{i=1}^{M} \left( \prod_{k=1}^{n} (1 - \alpha)^{n^*(i)} \alpha^{1 - n^*(i)} \right)^2 \right) \cdot P(b_i) \tag{8}\]

Among them, \(l_i\) represents the number of bits occupied by the \(i\)-th field of the execution result, and \(n^*(i)\) represents the same number of bits in the \(i\)-th field of the execution result of the \(s\)-th execution body as the corresponding bit in the \(i\)-th field of the correct execution result \(v\), \(\beta\) is

\[\beta = P(w' \neq v \mid b_i = -1) = P(w' \neq v \mid b_i = -1) \tag{9}\]

In order to maximize the value of formula (1), it is assumed that the degree of attack by the attacker on each of the executors in the set of executors is the same. This article assumes that the attacker’s degree of attack on the executor is represented by \(\lambda\), because each executor in the executor set is independent of each other. Therefore, \(P(b_i)\) can be expressed as the following formula:

\[P(b_i) = \prod_{s=1}^{M} P_s(b_i) \tag{10}\]

The following formula can be obtained:

\[v^* = \arg \max_v \prod_{s=1}^{M} \left( \lambda(1 - 1 - \beta) + (1 - \lambda) \alpha^{1 - n^*(i)} \right) \tag{11}\]

4.2. Decision Algorithm

The decider in this paper needs to make judgments on the flow table information so as to select relatively correct flow table information. This paper uses the HASH table method to implement the determination. The specific determination algorithm is as follows:

Decision Algorithm

```
{IP
Cipher
}
input MSG msg[i] i<-[0,M]
output The randomly selected correct controller executor is connected to the network layer, and the executor that needs to be offline or cleaned up is sent to the scheduler
1) MSG TempSet[]
2) for(i<0; i<=M; i++)
3) if (msg[i].Cipher not in TempSet[])
4) TempSet[].Cipher.append(msg[i].Cipher)
5) Count[j]++;
6) for(j<0; j<=len(TempSet[].Cipher); j++)
7) if(len(TempSet[].IP) != max(count))
8) if(msg[i].Cipher == TempSet[j].Cipher)
9) TempSet[j].IP.append(msg[j].IP)
10) count.append.len(TempSet[j].IP)
11) for(j<0; j<=len(TempSet[j].Cipher); j++)
12) if(len(TempSet[j].IP) != max(count))
13) Send TempSet[j].IP to Schedule
14) else
15) Random(TempSet[j].IP) to Network
```

4.3. Efficiency analysis

In the mimic defense system, the decider makes a pairwise comparison when deciding the output results of each executor in the executor set. Assume that the total time it takes for the decider to complete the comparison of all the output results is denoted by \(T^*\), and the time for the comparison of
any two output results is denoted by \( t \), then
\[
T^* = C^* t
\]

This decision mode, which directly compares the output of the executor, consumes a lot of time, and may even affect the performance of the entire system. Therefore, this paper proposes to perform MD5 encryption on the output results of each executor in the execution set.

After the output result is encrypted by MD5, all the parameters that the decider needs to compare are encrypted into an MD5 encrypted ciphertext. This reduces the parameters that need to be compared, thereby shortening the comparison time. Therefore, the time for any two execution results comparison can be expressed by \( t / \omega \) (\( 0 < \omega < 1 \)). For different service chain requests from users, the value of \( \omega \) is different, and the total time required by the decider is
\[
T^* = C^* \frac{1}{\omega} t
\]

Therefore, the time saved by the decider is
\[
\Delta T = T^* - T^*
\]
\[
= C^* t - C^* \frac{1}{\omega} t
\]
\[
= C^* t (1 - \frac{1}{\omega})
\]
\[
= C^* t (\omega - 1 \omega)
\]

After MD5 encryption, the decision time of the decider is shortened, the decision efficiency of the decider is improved, and the performance of the entire system is improved.

5. Simulation and experiment

5.1. Simulation analysis

In this paper, five SDN controllers are set as executor, and a flow table that meets the service function chain is output on the controller for judgment test. First test the non-MD5 encryption processing and non-hash processing judgments and record the time required. Then test the non-MD5 encryption processing and hash processing to record the time required. Then test the MD5 encryption and non-hash processing to record the time required. Finally, test the MD5 encryption processing and hash processing to record the time required. In order to avoid the influence of the number of wrong results in the decision process on the time of the decision process, this article assumes that one of the five executor output results is attacked for experiments. The experimental results are shown in figure 3.

The data show that as the length of the flow table increases, the decision time becomes longer. If no processing is performed on the flow table, the judgment takes the longest time. If only the hash table is processed, the decision time can be reduced to a certain extent. If only the MD5 encryption process is performed on the flow table, the time consumed by the decision will hardly fluctuate as the flow table length increases. If MD5 encryption and hash processing is performed on the flow table, the decision consumption time can be effectively reduced.

5.2. Experimental test

The tools used in the experiments in this chapter include three operating systems: Windows, Ubuntu,
and CentOS; three SDN controllers: RIU, ONOS, and Floodlight; Open vSwitch switches, KVM virtualization tools, and Nginx servers.

Install the OVS virtual switch on the host Ubuntu 16.04 LTS system, and use the ovs command to create the switch connection topology shown in figure 4.

![Figure 4. Network topology diagram](image)

5.3. **Offense and defense test**

By simulating an attacker to attack the mimic defense system, it is observed whether the result of the system using the decision algorithm in this paper is consistent with the request issued by the user. If the result obtained after using the decision algorithm is consistent with the request issued by the user, it indicates that the mimic defense system has achieved the effect of mimic defense, otherwise it indicates that the effect of mimic defense has not been achieved. In this section, in order to observe the output of the executor intuitively and to facilitate the tampering attack on the output result of the executor, it is not MD5 encrypted.

First, mimicry is turned on, and the decider randomly selects a controller's flow table and sends it to the switch. Simulate the user to perform the corresponding operation. Simulate the attacker to tamper with the target system, simulate the attacker to tamper with the flow entries delivered by the executor to the switch. This article uses manual modification to modify the key field values of the executor output flow entries, as shown in figure 5. As shown, the output port of the onos output flow entry is changed from 1 to 3 (red box in the figure), ryu and floodlight both issued the same flow entry (the same output value, blue box in the figure), but the verdict The router can still make correct decisions and output the correct flow entries, as shown in figure 6. Finally, the executor sends normal flow entries to the switch, as shown in figure 7.

![Figure 5. Executor tampered with output stream table entry](image)

![Figure 6. The executor is tampered with the output stream table entry of the decision maker](image)

![Figure 7. Switch flow table after tampering](image)
The experimental results before and after prove that the mimic defense system in this paper successfully achieved the effect of mimic defense.

6. Concluding remarks
This paper takes the mimic defense mechanism proposed by Academician Wu as a guide and optimizes the flow table decision algorithm in a mimic environment. Simulation analysis shows that the decision algorithm proposed in this paper can efficiently and correctly select the correct flow table.

This architecture proposes a heterogeneity-guaranteed scheduling mechanism and MD5 encryption decision mechanism on the original mimic defense architecture. This technology not only improves the security of the mimic defense system, but also improves the decision efficiency of mimic defense.

References
[1] Porras P, Seungwon S, Vinod Y, Maetin F, Mabry T and Guofei G 2012 J. A security enforcement kernel for Openflow networks (New York: Association for Computing Machinery) PP 121-126
[2] Porras P, Cheung S, Fong M, Skinner K and Yegneswaran V 2015 Securing the Software-Defined Network Control Layer (California: Association for Computing Machinery)
[3] Sonchack J, Aviv A J, Keller E and Smith J M 2015 POSTER:OFX: Enabling Openflow Extensions for Switch-Level Security Applications (New York: Association for Computing Machinery) PP 1678-80
[4] Zhang X, Hou W G, Han P C and Guo L 2014 Design and Implementation of the Routing Function in the NOX Controller for Software-Defined Networks J. Applied Mechanics and Materials. 635-637 1540-43.
[5] Nelson T, Ferguson A D, Da Y, Fonseca R and Krishnamurthi S 2015 Exodus:toward automatic migration of enterprise network configurations to SDNs (New York: Association for Computing Machinery) PP 1-7
[6] Shin S, Song Y, Lee T, Lee S, Chung J and Porras P 2014 Rosemary: A Robust, Secure, and High-performance Network Operating System (New York: Association for Computing Machinery) PP 78-89
[7] Jie C and Zhao L 2014 Design and implementation of firewall policy auditing scheme based on improved policy tree J. Applied Mechanics and Materials. 736 179-182
[8] Vanaubel Y, Luttringer J R, Merindol P, Pansiot J J and Donnet B 2019 TNT, Watch me Explode: A Light in the Dark for Revealing MPLS Tunnels (Paris: Network Traffic Measurement and Analysis Conference) PP 65-72
[9] Qing T, Huan Z, Weihua Z and Jiangxing W. 2017 Design and Implementation of Mimic Defense Web Server J. Journal of Software TP393.08
[10] Yawen W, Jiangxing W, Yunfei G, Honghao H, Wenyan L and Guozhen C. 2018 Scientific workflow execution system based on mimic defense in the cloud environment J. Frontiers of Information Technology & Electronic Engineering 19 1522-1536
[11] Zeyu G, Xingming Z and Senjie L 2017 Research on sdn control layer security mechanism based on mimic defense theory J. Application Research of Computers
[12] Jabeur N, Moh A N S and Barkia M M. 2016 A Bully Approach for Competitive Redundancy in Heterogeneous Wireless Sensor Network J. Procedia Computer Science 83 628-635.