Research of Inspection Firewall Fine Grained Access Control on SDN State

Hu Yang and Lu Ling

Computer Department, Guangzhou Civil Aviation College, Guangzhou 510470, China

huyangfox@163.com

Abstract. A new SDN state inspection firewall system base on OpenFlow protocol is proposed. In this scheme, state tables and shifted flow tables are added into SDN controller and switch, and corresponding state transition rules are formulated on basis of packet type, so firewall is able to inspect SDN network state. SDN stat inspection firewall is implemented on opensource controller Floodlight and Open vSwitch. The performance evaluation result shows that the SDN state inspection firewall can recognize the type of packets, moreover, it achieves the fine-grained access control which present SDN firewall cannot offer. The innovation of this paper is to put forward of inspection firewall fine grained access control on SDN State. by comparing the packet delay and connection establishment speed of the two firewalls, it can be concluded that the firewall system based on state detection has shorter delay time, in general, the connection establishment time takes up less proportion in the whole data transmission. Therefore, in summary, the overall performance of state-based firewall system is better than that of packet filtering firewall system.

1. Introduction

With the development of new businesses represented by cloud computing and big data in Civil Aviation information network, the traditional network is difficult to meet the flexible resource requirements, mainly because traditional networks have become too complex and can only operate in static mode. Therefore, a new network architecture is urgently needed to realize the flexible management of the network, thus SDN [1] came into being.

Open Flow [2] is the current mainstream protocol of SDN, which defines the communication standard between SDN controller and switch. Flexible network control is realized through the separation of network control plane and data plane. The control layer issues corresponding instructions to the switch by using the Open Flow protocol, so as to realize centralized control of the whole network. At present, SDN devices based on Open Flow have been deployed at a certain scale in practice [3-10]. To solve this problem, starting from the characteristics of SDN architecture, a SDN state detection firewall system based on Open Flow is designed and implemented by adding state table and state transition table in SDN switch and adding state table in controller. The main contributions of this paper are as follows.

(1) SDN firewalls are mostly traditional packet filtering firewalls, lacking the function of state detection. However, this paper realizes the detection of global network communication state through the way of switch and controller working together. This design scheme is relatively efficient and does not violate the purpose and original intention of SDN architecture design. On the basis of ensuring that the centralized control characteristics of the controller are not affected, the controller USES a small amount...
of communication between the controller and the switch to achieve the acquisition and control of the global network communication state by the controller. (2) realize the state-based data transfer. The existing Open Flow table is stateless, because it cannot complete the function of detecting the network communication state, which brings a lot of security threats, such as the inability to achieve fine-grained access control and the difficulty in defending against DDoS attacks. In this paper, the basic structure of the flow table defined in OpenFlow protocol is modified, and relevant status items are added, so that data flow and distribution are no longer irrelevant to the state, and the maintenance and update function of the state table is completed. At the same time, administrators can define rules based on state tables to achieve fine-grained access control. (3) performance analysis shows that compared with the traditional firewalls based on packet filtering, the state-based firewalls proposed in this paper have better comprehensive performance.

2. Related work and analysis
At present, the related work of SDN firewall includes: Shin et al. put forward an Open Flow security application development framework FRESCO[7], which is the secondary development of NOX controller. On the basis of NOX kernel security reinforcement, external interfaces of legacy network security system are provided to ensure the maximum compatibility of the platform. FRESCO "contains 16 basic module, each basic module has five interface: input, output, and events, parameters and operation, its biggest highlights of the basic module can be through the five interface into the safety of the complex application modules, many common network security platform and the function, so as to replace the firewall, IDS, and traffic management tools. Porras et al. proposed a reinforced control plane operating system Fort NOX. Policy conflict resolution was added by extending the Send_OpenFlow_Command module of the open source NOX operating system. Policies from different applications set different security levels, such as from security applications (such as firewall, intrusion detection, intrusion prevention), trusted applications and other application policies to provide security services have the highest priority; The local application strategy of control layer operating system has medium priority; Other business-providing applications are assigned the lowest priority. Extended FT_Send_OpenFlow_Command gathers all the application policies, authenticates the source of the policies by verifying the digital signatures carried by the application policies, checks whether there are policy conflicts, and determines the actions when policy conflicts occur according to priority.

3. The Firewall System of SDN State Base on OpenFlow
The firewall system of SDN state detection aims to build a fine-grained state-based access control mechanism for SDN network. Based on open source SDN controller Floodlight function extension to the original packet filter firewall, intercepted packet in switch and pull out information related to the state of the application layer, one important information sent to the controller, and a decision based on data forwarding, so as to realize state based access control, at the same time has good adaptability, scalability and low cost.

3.1 System Framework
In order to realize the function of state firewall, satisfy the centralized control of controller and facilitate the overall management and operation of network administrators, this paper proposes the system architecture shown in figure 1, including the following 5 modules. (1) Key Extractor module: the packet header information extraction module in the switch extracts the Key information in the packet header. (2) State table-sw module: the establishment and update module of the State Table. The function of this module is to establish the connection State Table in the switch and simultaneously update the connection State Table to the controller. The update of the State Table is controlled by the transformation flow Table through the set_state instruction. (3) Shifted Flow module: transform Flow Table (Shifted Flow Table)
Set up and update the module. The controller sends instructions to set up the transformation flow table in the switch, and is responsible for the state transition process and packet forwarding operation. 

(4) Legitimacy module: Legitimacy detection module is responsible for determining which connection the data packet arrived at the switch belongs to.

(5) the State Table - C modules: the controller command issued, the State Table synchronization module, a connection is established in the controller status Table, and with the switch in the State Table, when switches send Packet_In messages are received at the same time, the module through the header information, State information and links, comparing the status Table or firewall rule sets assigned corresponding State, at the same time issued transform flow chart into the switch. The first four modules are implemented by adding corresponding functions in the switch for packet matching. The State table-c module is implemented in the controller to determine the validity of the connection. The State table-c module needs to use the Open Flow Core Service provided by the controller to communicate with the switch, and use the Firewall module provided within the controller to query the Firewall rules to determine whether the packet is legal.

3.2 Key technologies of the system

The key technologies involved in this system mainly include: establishment of connection state table, management of transformation flow table and update management of state table.

3.2.1 Establishment of Connection status table. This system defines two message structures, STATE_IN and OFP_STATE_MOD, to complete synchronous operation of connection state table in controller and switch. When the state table in the switch is updated, it will send a STATE_IN message to the controller to inform the controller to update the corresponding state record. When the state table in the controller is updated, the controller will send an OFP_STATE_MOD message to the switch, which commands the switch to update the corresponding state record.
For the actual problems to be solved by this system, the connection status table is composed of quaternions (Match Field, State, Timeout, Packet_count).

The Match Field includes IP (source address and destination address, but only for connection state, no distinction is made when matching connection state table) and protocol type (including TCP/UDP, ICMP, etc.) of packet; State represents the connection State; Timeout indicates the Timeout time for a connection; Packet_count represents the number of packets passed.

3.2.2 Transform flow table management. In order to update, maintain and manage the connection State table, the system modified the OpenFlow flow table, added the State attribute (State) and the next State attribute (Next_State), and redefined the matching process of data packet and flow table item. The result of the match depends not only on the packet header information, but also on the packet state. When the match is successful, the OFPIT_SET_STATE instruction will be executed, which will transform the next state value (Next_State) in the corresponding record in the flow table to the state attribute value in the state table. At the same time, the packet is processed according to the ACTION instruction. If the match is unsuccessful, the switch will send a Packet_in message containing packet header and state information to the controller. In response, the controller returns a Flow_mod message to add a flow table to the switch.

3.2.3 Status update management. Another key technology in this system is status update management, that is, what the switch will do when a packet passes through it.

(1) status query: query the connection status table with packet packet header information as the query key (such as source IP address). If there is no relevant query record, add the record in the status table and set its status as DEFAULT.

(2) transform the flow table to achieve state transition. When the packet and the state connection table are successfully matched, state information will be added in the packet header; The state information will also be matched as a match and the table transform table. If the match is successful, the next state information in the packet header will be written back to the connection state table, the state table will be updated, and the packet will be processed according to the ACTION table item in the flow table.

(3) status updates, which include additions, deletions, and modifications, are usually done through the OFPIT_SET_STATE directive, in some cases using State_mod.

4. System implementation process

4.1 Specific process

In the network, most of the data transmission is based on the connection oriented TCP protocol, and once the control of the connection establishment process, the whole communication process is considered to be reliable. In this system, TCP packet processing process is typical and representative, so this paper will focus on TCP packet state detection process for design and implementation. The main task of the TCP packet state detection module is to create a TCP connection state table. The specific process is shown in figure 2:

(1) when the packet arrives at the switch, the key information extraction module of the switch is started and matched with the state table in the switch.

(2) if no match is successful, add the record in the state table, set the state as DEFAULT, and inform the controller to update the corresponding state table (4).

(3) if the match is successful, the status information will be written to packet header.

(4) match with the transformation flow table in the switch. If the match fails and no corresponding flow table information is found, then judge whether the packet is SYN packet or not.

(5) if the SYN packet, for the establishment of a new connection with the controller module of firewall rule set match, after the success of the match, Flow_mod messages sent to the switches, changing flow Table, add the record, the action for the forward, assign a status symbol, receipt of switches, immediately perform SET_STATE operations, that is, update the connection State Table (including controller connection State of the Table (the State Table-C, and switches the connection State Table
(State Table-SW), The two are synchronized by State_mod), and the packet is forwarded after legitimacy detection.

(6) if it is not SYN data packet, it indicates that it may be part of the original connection, and no rule matching is required. Directly query the controller connection status table. Upon receipt by the switch, SET_STATE operation shall be performed immediately, that is, the connection status Table (including the connection status Table in the controller (State table-c) and the connection status Table in the switch (State table-sw), which are synchronized by State_mod) shall be updated, and the packet shall be forwarded after legitimacy detection.

(7) if the state Table in a packet with switches (state Table - SW) and transform Flow Table (Shifted Flow Table) match, the packet header information is sent to the controller and directly handled by switches.

4.2 Specific problem analysis

Packet-filtering firewalls allow or block communication between internal and external networks. However, this requirement cannot be satisfied: the inner network is allowed to send connection requests to the outer network, while the outer network cannot send connection requests to the inner network. This can better protect the internal network, prevent attackers from sniffing, scanning, and so on. Here, the firewall is detected based on the SDN status mentioned above to illustrate how to solve this problem (in order to fully demonstrate the functions of each module, the connection status table and transformation flow table do not contain relevant record information). The inner network h1 (172.0.1.1) can actively establish a connection with the outer network h2 (10.65.10.1), while the h2 in the outer network cannot actively initiate a connection with the inner network. First, define the rules in the firewall as shown in table 1.

When h1 sends connection request, after SYN packet arrives at the switch, the key information extraction module in the switch starts to extract its packet header information and conduct matching operation with the connection state table in the switch. Obviously, there is no relevant record in the switch at this time, so add relevant records in the state table, as shown in table 2.

At the same time, the status information is written to the packet header of the SYN packet and matched with the exchange flow table. If no relevant records are found in the transform flow table, the key information will be sent to the controller. After the controller determines that it is the SYN request package, it matches the firewall rules and gets the allowed connection result. Then the controller will send Flow_mod message to the switch and add new records in the transform flow table, as shown in Table 3.

After passing the detection, the switch needs to check the legitimacy of packet packet header. After passing the detection, SET_STATE operation will set the State value of State Table -sw related records to Next_state in the switching flow Table, and synchronize the information to the State Table -c of the controller.

Through the above operation, the outer network host h2 successfully receives the connection request sent by h1 and returns a SYN+ACK packet. When the packet arrives at the switch, it is successfully matched to the relevant records in the STATE table. After the STATE value SEND_SYN is written into the packet packet header, it is matched with the exchange flow table. Obviously, the match fails (in the transformation flow table, there is no STATE field for SEND_SYN item). After the packet header...
information is sent to the controller again and successfully matches with the controller's State table-c, the controller sends Flow_mod message to the switch to add new records in the transformation flow Table, as shown in Table 4, and writes the Next_State value to the switch State Table and synchronizes it to the controller.

After the legitimacy detection, the packet is sent to the host of Intranet h1, which replies an ACK packet. Same as the above steps, the update of the transformation flow table and the update of the status table are shown in Table 5 and Table 6.

### Table 5. Update of the transformation flow

| A       | B            | Next_state | Action |
|---------|--------------|------------|--------|
| 172.0.1.1 | 10.65.10.1   | SEND_SYN   | Forward|
| 10.65.10.1 | 172.0.1.1   | REC_SYN    | Forward|
| 172.0.1.1 | 10.65.10.1   | ESTABLISHED | Forward|

### Table 6. Update of the status

| A       | B            | STATE      | Status related information |
|---------|--------------|------------|----------------------------|
| 172.0.1.1 | 10.65.10.1   | Established | Skip without examples     |

At the same time, when the controller STATE table STATE value is ESTABLISHED - that is immediately to the switches in the transform flow Table to add the following flow chart information, as shown in Table 7.

When the connection is established, the inner network h1 can communicate with h2 normally without being blocked. Once the switch receives information with RST or FIN flags in the relevant connection, the controller will delete the connection records in the State Table (including State table-sw and State table-c) and delete the relevant items in the transformation flow Table.

With the SDN state firewall, administrators can achieve more fine-grained access control. To achieve this in a packet filtering firewall, an administrator might configure the following firewall rules, as shown in Table 8.

### Table 7. Change flow table state after connection establishment

| A       | B            | Next_state | Action |
|---------|--------------|------------|--------|
| 172.0.1.1 | 10.65.10.1   | ESTABLISHED | Forward|

### Table 8. Packet filtering firewall rules

| Protocol type | The source IP address | Destination IP address | operation |
|---------------|-----------------------|------------------------|-----------|
| TCP           | 172.0.1.1             | 10.65.10.1             | ALLOW     |
| TCP           | 10.65.10.1            | 172.0.1.1              | DENY      |

The status detection firewall can prevent the external network from sending connection requests to the internal network while allowing the normal communication between the internal network host and the external network. Therefore, SYN packets on the external network will be rejected by the firewall, which makes it very easy to defend against SYN flooding attacks.

### 5. System test and performance analysis

#### 5.1 Access control function test

The system is deployed in Floodlight and Open v Switch, and the test environment configuration is shown in Table 9.

### Table 9. Test environment

| entity            | configure                        |
|-------------------|----------------------------------|
| CPU               | Intel Core i5-4200m main frequency 2.5ghz |
| memory            | 8 GB                             |
| The operating system | Ubuntu 12.04 X64               |
| switches          | Open v Switch 1.9.3              |
| SDN controller    | Floodlight 9.0                   |
| Virtual machine manager | KVM                             |
| Virtual machine operating system | Cent OS7.1                       |

This test USES the network topology shown in figure 3, including 3 switches (S1, S2, S3) and 4 hosts (VM1, VM2, VM3, VM4).

Set the IP address of VM1 to 172.0.1.1 and VM3 to 10.65.10.1. Add rules in Floodlight firewall module as shown in Table 10.
Table 10. Firewall module rules

| Protocol type | The source IP address | Destination IP address | Operating collection |
|---------------|-----------------------|------------------------|----------------------|
| TCP           | 172.0.1.1             | 10.65.10.1             | ALLOW                |
| TCP           | 10.65.10.1            | 172.0.1.1              | DENY                 |

Figure 3. Network Topology Test

5.2 System performance analysis

5.2.1 Packet transmission latency. The test network topology is built based on Open v Switch and virtual VM, as shown in figure 3. Under the SDN architecture, the comparison of transmission delay between two firewalls based on packet filtering and state-detecting is shown in figure 4. It can be concluded that, with the increase of data volume, the transmission delay of state-detecting firewalls is gradually lower than that of packet filtering firewalls.

5.2.2 TCP connection establishment speed. At the same time, the speed of connection establishment between firewalls based on packet filtering and state detection is compared under SDN architecture. The total time taken to establish 100~900 TCP connections was used to compare the performance of the two firewalls, as shown in figure 5.

As can be seen from figure 5, the speed of connection establishment is generally slower than that of the firewalls based on state detection, because the firewalls based on packet filtering need to carry out various operations such as state establishment and synchronization at the initial stage of connection establishment. When the number of connections is low, the difference is not significant because each connection takes milliseconds. The more connections, the overall state of the firewall slower speed. The results were the same at individual points, mainly because the speed was not stable and the difference was not significant.

Under the SDN architecture, by comparing the packet delay and connection establishment speed of the two firewalls, it can be concluded that the firewall system based on state detection has shorter delay time, but slower connection establishment speed. However, in general, the connection establishment time takes up less proportion in the whole data transmission. Therefore, in summary, the overall performance of state-based firewall system is better than that of packet filtering firewall system.

Figure 4. Transmission Delay Test
Figure 5. TCP establish connection speed test
6. Conclusion
To solve the problem that it is difficult for SDN firewall to realize state-based fine-grained access control, this paper designs and implements an Open flow-based SDN state detection firewall system, proposes the modification and design method of Open Flow Flow table and protocol, as well as the key technology and implementation process of connecting state table and transformation Flow table. The system is implemented with Floodlight Open source controller and Open v Switch Open source Switch. By modifying the Open Flow table structure and protocol and adding state detection modules to the switch, the system can detect the state of packets, thus enabling network administrators to define rules to achieve fine-grained state-based access control functions. Finally, the function and performance of the system are evaluated, and the results show that the system can achieve the state detection function that the existing SDN firewall does not have, and has a low overhead.

7. References
[1] Foundation O N. Software-defined networking: The new norm for networks[J]. ONF White Paper, 2012, 2.
[2] Mckeown N, Anderson T, Balakrishnan H, et al. Open-Flow: Enabling innovation in campus networks[J]. ACM Sigcomm Computer Communication Review, 2008, 38 (2): 69-74.
[3] Porras P A, Cheung S, Fong M W, et al. Securing the software defined network control layer [C]// Proceedings of the Network and Distributed System Security, San Diego, 2015.
[4] Dai B, Wang H Y, Xu G, et al. Opportunities and threats coexist in SDN security[J]. Application Research of Computers, 2014, 8: 2254-2262.
[5] Martins J, Ahmed M, Raiciu C, et al. Click OS and the art of network function virtualization[C]// Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation, 2014: 459-473.
[6] Sherry J, Hasan S, Scott C, et al. Making middleboxes someone else’s problem: Network processing as a cloud service [J]. ACM SIGCOMM Computer Communication Review, 2012, 42 (4): 13-24.
[7] Shin S, Porras P A, Yegneswaran V, et al. FRESCO: Modular composable security services for software-defined networks [C] // Proceedings of the Network and Distributed System Security, San Diego, 2013.
[8] Kreutz D, Ramos F M V, Verissimo P E, et al. Software-defined networking: A comprehensive survey [J]. Proceedings of the IEEE, 2015, 103(1): 14-76.
[9] Son S, Shin S, Yegneswaran V, et al. Model checking invariant security properties in Open Flow [C] // 2013 IEEE International Conference on Communications (ICC), 2013: 1974-1979.
[10] MATIAS J, GARAY J, M ENDIOLA A, et al. Flow NAC: Flow-based Network Access Control [C] // IEEE. 3rd European Workshop on Software Defined Networks, September 1-3, 2014, London, UK. New Jersey: IEEE, 2014: 79—84.

Acknowledgments
This work is partially supported by Natural Science Foundation in University of Guangdong (No.GDJS2019303)