An SDN Controller Security Cluster Scheme Based on Intrusion Detection Technology

Xiao-feng XU\textsuperscript{1,2}, Jia-hao DAI\textsuperscript{1,2,*} and Gao-lei YANG\textsuperscript{3}

\textsuperscript{1}Science and Technology on Communication Information Security Control Laboratory
\textsuperscript{2}China Electronic Group Corporation NO.36 Institute, Zhejiang Jiaxing, China
\textsuperscript{3}Xidian University, Shanxi Xi’an, China

*Corresponding author

Keywords: Network security, Intrusion prevention, Software defined network, Cluster.

Abstract. Software Defined Network (SDN) is a new network innovation architecture that separates the control layer and the data layer of the network. In the SDNs, the control layer is the core of the entire network, and its security is very important. In practical applications, the control layer is usually composed of SDN controller clusters. In order to enhance the security of the SDN controller cluster, this paper first studies the development status of the current SDN technology, and then analyzes the current state of the defense technology of the network system. Through the analysis and summary of the two, we proposed the discovery and isolation scheme of the invaded controller in the cluster environment, and designed the intrusion detection module, control module and defense module. Finally, we implemented the scheme based on Opendaylight controller, and proved the effectiveness of the scheme by experiments.

Introduction

Traditional network devices (such as switches) consist of two main components: the control plane and the data plane. The control plane is responsible for performing complex network functions, such as calculating the forwarding path and creating a routing table to determine the control strategy for network traffic. The data plane is tightly coupled to the control plane, which forwards network packets based on processing strategies from the control plane. If a new feature or network protocol is to be added, the device provider must redevelop a new device. This process usually takes a long time and is a big obstacle to the network innovation. Software Defined Network (SDN) is a new network architecture, it separates the control layer and the data layer of the network. This separation makes the underlying network infrastructure abstract and unified with respect to network applications and services, so that network control can be implemented directly through programming, which accelerates the deployment of new services. It has also accelerated the innovation of the network\cite{10}. However, due to the centralized management, openness and programmability of SDN, it also brings certain security risks to SDN: The biggest security issue with SDN comes from the controller. As a central control software, the controller is the brain of the whole network. It has a topology map of the entire network resources and is responsible for the control strategy of network traffic. Once the entire network is attacked, it will be awkward. This is very different from the traditional network and should be paid enough attention\cite{8}.

In the protection of network devices, Intrusion Detection System (IDS) is a network security device that monitors network transmissions in real time and alerts or takes active measures when suspicious transmissions are discovered. It differs from other network security devices in that IDS is a proactive security protection technology. Zhang et al.\cite{1} analyze the IDS technology, explain the technical principle of intrusion detection, and classify IDS from two angles. They discuss the implementation method of the system and the existing IDS system. Hu et al.\cite{2} summarize the research status of intrusion detection technology from the aspects of attack detection methods and research status at home and abroad, and combine its own research results to propose development trends and main
research directions. Zhang et al. [3] propose a more effective intrusion detection scheme based on the network architecture of SDN. The scheme makes full use of the characteristics of the SDN architecture and the unique self-similar characteristics of the network, and can timely determine whether any nodes in the WSN are invaded, which greatly solves the defects of the energy and computing power of the common nodes. Moreover, the detection scheme can be flexibly deployed according to the environment of the WSN, ensuring the diversity of the detection scheme, and actively responding to the intrusion suffered by the WSN, thereby improving the security. Xiang et al. [4] seek to build a defense system with intrusion detection, automatic response and flexible decision-making based on the SDN framework, and design a threat assessment scheme for alarm analysis, which is based on credibility assessment and alarm consolidation.

Based on the above references, we carefully compare the strengths and weaknesses of each intrusion detection scheme, and propose an SDN-based intrusion detection scheme. Using the programmable features of the SDN network, we deploy the intrusion detection module, control module and defense module on the application layer. We deploy intrusion detection scheme in a cluster environment to realize the discovery and isolation of malicious controllers in a cluster. We then prove the feasibility of this scheme by experiments.

Related Technology Research

SDN Technology

SDN originated from an internal research project at Stanford University in the United States, which was proposed to solve various problems in the traditional network architecture. Once the SDN technology was put forward, it has received widespread attention from academia and industry, and its research and application prospects have been widely recognized. SDN has also become one of the more researched in next-generation network technologies [5].

The SDN architecture decouples traditional closed network architectures into application, control, and data layers. It realizes the centralized control and management of the network logically. SDN provides a good platform for improving the development and innovation of network applications [9].

The data layer is mainly composed of various devices in the network, and these devices only have a forwarding function. The main network device is Openvswitch (OVS). Each switch has a flow table, and the switch performs corresponding actions on the corresponding data packet according to the flow table. The control layer is the core part of the SDN architecture, and the SDN controller works on this layer. The controller interacts with the network device of the data layer through the southbound interface, the typical southbound interface is OpenFlow protocol. The control layer masters all the information of the entire network topology, generates a flow table, and can send the flow table to each switch on the forwarding layer[6]. The application layer is a variety of network applications that implement control over the network based on the northbound interface provided by the control layer to meet specific needs [7].

Intrusion Detection Technology

IDS is an important part of the information security architecture. It can be divided into host-based, network-based and hybrid IDS according to the deployment locations. According to the different detection methods of the analysis engine, it can be divided into abnormal detection and misuse detection.

In the intrusion detection system, Snort has a good reputation in the industry and is powerful enough to detect multiple network attacks. Snort is an open source software written in C that conforms to the GNU General Public License (GPL) specification. Snort is a network-based misuse intrusion detection system, it uses libpcap software to capture packets on the network and analyze the packets with malicious data matching, operations such as alarms and logging are performed after
suspicious data is discovered. Because Snort supports multiple platforms, the structure is clear, and is easy to expand, it become a model for researching IDS/IPS.

Intrusion prevention system (IPS) is a proactive defense system deployed on the critical path of the network to detect traffic through itself. When traffic with threats is detected, it sends an alarm message and actively filters out the traffic. IPS is equivalent to the combination of IDS and firewall. It also has the IDS detection function and the firewall filtering function.

**Controller Cluster**

A cluster is a group of independent computing nodes that make up a system through a high-speed network. Consensus refers to the agreement of multiple servers in the state. In a cluster, it is difficult to ensure data consistency. In previous single-node services, locks were often used to achieve consistency. When a concurrency conflict occurs, the operation of holding the lock is an effective operation, thereby ensuring that the data is only allowed to be operated by one request at the same time. However, in the cluster, if the lock mechanism is also used, then one node is needed to manage the allocation lock. Before other nodes make the request, it first acquire the lock to obtain the execution right. This will make the whole cluster become a serialized single node, the meaning of the cluster is lost. Therefore, a consistency protocol is needed. Opendaylight cluster uses the Raft consistency algorithm for the election of the Leader. Each controller instance has three states, Follower, Candidate, Leader. The Raft protocol synchronizes the data of all nodes in the cluster through the leader node to achieve cluster consistency. When the client sends data, it is first sent to the leader node. After receiving the data, the leader node first marks the data as unsynchronized, and then the Leader node sends the data to all Follower nodes. The Follower node sends a response to the leader node that the data is successfully received. After obtaining a response to the successfully received data of more than N/2 Followers in the cluster, the Leader node marks the status of the data as synchronized, and then the leader node sends a data reception (synchronous) success response to the client. Then the leader node will tell all the follower nodes that the data has been synchronized.

**Intrusion Prevention Schemes in a Cluster Environment**

**Overall Scheme**

Programmability is one of the core features of SDN. Based on this feature, applications for various needs can be developed at the application level of SDN. The intrusion prevention scheme designed in this paper is mainly realized by developing corresponding application modules on the application level of SDN and interacting with the existing core modules on the Opendaylight controller. The overall architecture of the system is shown in Figure 1.

![Figure 1. System architecture diagram.](image-url)
As shown in the Figure 1, firstly, according to the role mechanism of the Opendaylight controller cluster, the Leader controller is regarded as the central controller, and the Follower controller is regarded as the member controller. We deploy a detection module and a forwarding module on the member controller, and deploy a defense module on the central controller. The main functions of each module are as follows:

1. The main detection module is used for crawling and security analysis of the data packets of the intranet, and generates an alarm when a security threat is discovered.

2. The forwarding module is mainly used to receive an alarm generated by the detection module and transmit the warning message to the central SDN controller.

3. The defense module is deployed on the central controller to receive security alarm messages sent by the forwarding module, and threat isolation of the corresponding member SDN controller according to the level of the alarm message.

In addition, this solution provides users with a web interface. On the web interface, users can configure controller cluster parameters and monitor the information of each module.

**Detection Module**

The detection module is used for crawling and security analysis of intranet data packets. There are many methods for the existing intrusion detection. This paper selects the Snort intrusion detection system. Snort can work in IDS mode, analyze data traffic, use rule base to alert the attacks, and has good scalability and portability.

Of course, Snort also has many shortcomings. First, Snort is based on rule base detection and can only be filtered and searched based on protocol, IP address, port number, etc. This strategy is powerless when dealing with some advanced attacks such as API utilization. Secondly, many of Snort's libraries are outdated. In the specific development process, we need to redevelop these libraries, especially the rule base, to adapt to the Opendaylight cluster environment. Snort can only passively send out an alert message when it detects an attack, but it cannot effectively defend against the attack itself. Therefore, Snort must work with other modules to achieve defense goals.

After detecting the alarm, the Snort detection module writes the alarm information to the file and then transmits the alarm information to the Opendaylight controller instance via the socket communication mechanism.

This system requires snort to work in network intrusion detection system mode. Need to use the --enable-control-socket parameter, otherwise, snort cannot use the unix socket function after compiling. Snort needs to specify the snort.conf file when working in network intrusion detection mode, this file is a description of the matching rules. Snort's intrusion detection relies heavily on this rule set. The quality of the rules directly affects the detection accuracy and performance of the system. Snort's rule description language is simple in syntax and low in execution overhead. Such as this example:

```plaintext
alert tcp any any -> 192.168.1.0/24 111 (content:"|00 01 86 a5|"; msg: "mountd access");
```

Most rules are a single line. The first item of the rule is the rule action, and the keyword alert indicates that this is a trigger warning rules. The next item of the rule is the protocol, and the keyword tcp represents that this rule applies to packets using the TCP protocol. Currently, protocols supported by snort include TCP, UDP, ICMP, and IP protocols. -> is the direction operator, indicating the direction of data transfer. Four parameters are required, namely source IP address, source port address, destination IP address, and destination port address. Any can be used to replace any address. In addition to this, there are <> two-way characters. The part before the parentheses is the rule header, and the part inside the parentheses is the rule option, there is a option that end with a semicolon (;) in the rule options.
Forwarding Module

Due to the architectural characteristics of the software-defined network, the control plane is separated from the data plane. Therefore, Opendaylight cannot directly read the alarm information stored locally. It is necessary to design a forwarding module, read the alarm information from the intrusion detection module, and then send the alarm information to the Opendaylight controller.

The system is designed with a forwarding module deployed on the host where the Snort intrusion detection system is located. The forwarding module communicates with the Snort intrusion detection system through the Unix Socket mechanism to obtain real-time alarm information. Socket is an interprocess communication mechanism provided by the operating system. In the operating system, a complete set of socket application interfaces (Socket APIs) is usually provided. Between different processes, you can communicate through socket technology and exchange data.

Through the socket mechanism, the forwarding module can get the alarm information. Then the forwarding module needs to parse the obtained alarm information to obtain key data to construct a new alarm information. The new alarm information structure is alarm type, source IP address, destination IP address, source MAC address, destination IP address.

The forwarding module then sends the new alarm packet to the Opendaylight controller via the RESTful API mechanism. The RESTful architecture uses URLs to locate resources, and the client can operate the resources of the server through the HTTP protocol. We need to write a data receiver in the Opendaylight controller to receive the alert information of the forwarding module. Using RESTful API technology, this program can be externally represented as a RESTful API interface, which can be located using a URL. In this way, in the programming of the forwarding module, only the POST request needs to be made to the URL, and the alarm information can be pushed to the Opendaylight controller instance.

In specific implementation, the forwarding module uses Unix socket to communicate with the Snort network intrusion detection system. The core communication code is as follows:

```python
def start_recv(self):
    logger.info("Start.......")
    if os.path.exists(SOCKFILE):
        os.unlink(SOCKFILE)

        self.unsock = socket.socket(socket.AF_UNIX, socket.SOCK_DGRAM)
        self.unsock.bind(SOCKFILE)
        logger.info("Unix Domain Socket listening...")
        self.recv_loop()
```

Defense Module

The defense module first needs to get the alert message from the forwarding module, and then use the Opendaylight cluster's Datastore mechanism to notify the central Opendaylight controller instance of this alert information. Then processing the alert information to obtain relevant intrusion information, and isolate the compromised controller instance.

Since the message sent by the forwarding module is mainly implemented by the ODL Datastore mechanism, the message listener of the Datastore needs to be registered on the defense module, and the defense mechanism is triggered after the corresponding alarm message is monitored. DataStore is an in-memory database of Opendaylight. The data structure stored in this database is a tree structure defined by YANG. DataStore transaction-based access and operations, support for data change notifications and transaction chains. After that, the subsystem of the controller, Distributed Data Store, synchronizes the Data Store of the controller cluster, and the consistency protocol used is the raft protocol. In this way, the data of all controllers in the entire cluster is synchronized, the central controller receives the alarm information. Its core implementation code is:
public interface DataBroker extends DataTreeChangeService {
    ReadOnlyTransaction newReadOnlyTransaction();
    ReadWriteTransaction newReadWriteTransaction();
    WriteTransaction newWriteOnlyTransaction();

    BindingTransactionChain
    createTransactionChain(TransactionChainListener listener);
}

public interface DataTreeChangeService {
    ListenerRegistration<L> registerDataTreeChangeListener(@Nonnull
    DataTreeIdentifier<T> treeld, @NonNull L listener);
}

We uses newReadOnlyTransaction() to read data from the datastore and
newWriteOnlyTransaction() to write the data to the datastore. After getting the alert message, the
compromised controller is shut down and the controller leaves the cluster, thus achieving isolation of
the compromised SDN controller.

Experimental Test

Experimental Environment Construction

In order to test the feasibility and function of the intrusion prevention solution, a test platform needs to
be set up. The selection and version information of each component of the platform are shown in
Table 1.

Table 1. Test platform components selection.

| operating system | Ubuntu-16.04LTS |
|------------------|-----------------|
| Controller       | Opendaylight-Carbon |
| switch           | Openvswitch-2.3.0 |
| Topology         | Mininet-2.2.0   |
| Snort agent      | Snort-2.9.9.0   |

First we create an experimental topology on mininet. The specific topology is shown in Figure 2:

![Network topology](image)

Figure 2. Network topology.

After creating a topology, we use the Docker container to build the ODL controller cluster. The
version of the Docker container selected in this article is 18.09.2. The Dockerfile for building the
Docker image of the ODL cluster is as follows:
FROM ubuntu:18.04
ADD sources.list /etc/apt/
RUN apt-get update &&
    apt-get install unzip -y &&
    apt-get install net-tools -y &&
    apt-get install iputils-ping -y &&
    apt-get install vim -y &&
    apt-get autoclean &&
    apt-get autoremove
COPY distribution-karaf-0.6.4-Carbon.zip jdk-8u201-linux-x64.tar.gz /home/
WORKDIR /home
RUN tar -zxvf jdk-8u201-linux-x64.tar.gz &&
    rm jdk-8u201-linux-x64.tar.gz &&
    unzip distribution-karaf-0.6.4-Carbon.zip &&
    rm distribution-karaf-0.6.4-Carbon.zip
ENV JAVA_HOME=/home/jdk1.8.0_201
ENV JRE_HOME=$JAVA_HOME/jre
ENV CLASSPATH=.:$JAVA_HOME/lib:$JRE_HOME/lib:$CLASSPATH
ENV PATH=$JAVA_HOME/bin:$JRE_HOME/bin:$PATH
EXPOSE 8181 6653 6633

Then we use the docker run command to start a number of ODL controller instances and configure the controller to build an ODL controller cluster.

**Feasibility Verification**

After the topology is set up, invade the member controllers in the controller cluster to test the feasibility of the solution. First we add a test alarm rule on the detection module

```
alert icmp any any -> 220.181.57.216 any (msg: " rule test ")
```

According to this rule, when a ping request is sent to any controller in the cluster, an alarm is triggered to simulate a network attack, and the alarm is output in the alert file in the /var/log/snort directory. After the ping command triggers the alarm, we can see the alarm shown in Figure 3.

![Alert message](image)

Figure 3. Alert message.

After receiving the alarm message, the re-issuing module will pass the message to the central controller to run the forwarding module by using the python command. The forwarding module output is shown in Figure 4. We can see that the forwarding module successfully forwards the alert message and outputs the source and destination IP addresses of the dangerous traffic.
After receiving the alarm message, the central controller will isolate the compromised controller and output the IP address of the compromised controller in the log, as shown in Figure 5.

Conclusion

The application and promotion of SDN technology will change the traditional network architecture, control traffic from the entire network layer, improve the deployment speed of new services, save operating costs, and provide a new solution for network virtualization technology. However, the security problem of SDN is the key factor that restricts the further development and application of SDN technology. In this paper, based on the security problem in the SDN controller cluster environment, we propose a SDN-based intrusion detection system to realize the detection and isolation of the invaded controller in the cluster environment.

This paper summarizes and analyzes the knowledge and technology of software-defined network, Opendaylight controller cluster and intrusion detection technology, designs the overall architecture of the system. We constructed three functional modules and compared the advantages and disadvantages of different technologies and then deploy the scheme on the real environment. However, the false alarm status of the detection module is not considered, and the isolation means of the controller is relatively simple. We leave exploration of this venue to future work.

References

[1] Huimin Zhang, Jun He, Houkuan Huang. Intrusion Detection System[J]. Journal of Computer Applications, 2001, 18(9): 38-41.

[2] Huaping Hu, Haitao Chen, Chenlin Huang, et al. Research Status and Development Trend of Intrusion Detection System[J]. Computer Engineering and Science, 2001, 23(2): 20-25.

[3] Yong Zhang, Shengsheng Ji, Yiyi Wang. WSN Intrusion Detection Technology Based on SDN Architecture[J]. Journal of Henan University: Natural Science, 2015(45):216.
[4] Lai Xiang. Research and implementation of intrusion prevention system based on SDN framework [D].

[5] Shujun Wei. Design and Implementation of Network Security Isolation System Based on SDN Technology [D]. Beijing University of Posts and Telecommunications, 2015.

[6] Wenting Ma. Research on Key Technologies of SDN Controller Based on OpenFlow[D]. Beijing University of Posts and Telecommunications, 2015.

[7] Xiang Zhang. Design and implementation of SDN-based traffic awareness and bandwidth control system [D]. University of Electronic Science and Technology of China, 2015.

[8] Yoon C, Lee S, Kang H, et al. Flow Wars: Systemizing the Attack Surface and Defenses in Software-Defined Networks[J]. IEEE/ACM Transactions on Networking, 2017, PP (99):1-17.

[9] Bosshart P, Gibb G, Kim H S, et al. Forwarding metamorphosis: fast programmable match-action processing in hardware for SDN[J]. Computer Communication Review, 2013, 43(4):99-110.

[10] Zuo Q Y, Chen M, Zhao G S, et al. Research on OpenFlow-Based SDN Technologies[J]. Journal of Software, 2013, 24(5):1078-1097.