Journal of Computer Science and Technology Studies
ISSN: 2709-104X
DOI: 10.32996/jcsts
Journal Homepage: www.al-kindipublisher.com/index.php/jcsts

JCSTS
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

| RESEARCH ARTICLE

DoS Attack Detection and Defense on SDN Controller

Yalan Zhang[1] ✉ and Huiyun Ning[2]
[1]School of Information Technology and Electrical Engineering, University of Queensland, St Lucia QLD 4067, Australia.
[2]Assistant Engineer, Department of Communication and Signal, China Railway Siyuan Survey And Design Group, Wuhan, China.
Corresponding Author: Yalan Zhang, **E-mail**: yalan.zhang@uq.net.au

| ABSTRACT
With the development of the Internet, the flexibility and manageability problems of traditional network architecture have become increasingly prominent. To solve this problem, Software Defined Network (SDN) was born in recent years. The core idea of SDN is to decouple the data forwarding layer and the control layer, which makes SDN centralized, expandable, and programmable. The core services, important configuration, and other functions of SDN are deployed on the SDN controller, which is conducive to the centralization of network management but also introduces threats to network security. If the SDN controller is attacked, it will affect the coverage of the controller. The entire network, in extreme cases, will be paralyzed. Based on the analysis of the characteristics of the SDN network architecture, OpenFlow protocol, and the principle of DoS attacks, this project proposed and implemented two DoS detection and mitigation methods. The first one uses the Ryu controller, sFlow, and Postman to visualize and drop DoS attack traffic. The second method is a DoS attack automatic detection algorithm with a POX controller.

| KEYWORDS

Software Defined Network (SDN), SDN controller, OpenFlow, DoS attacks

| ARTICLE DOI: 10.32996/jhsss.2022.4.2.2

1. Introduction

Software Defined Networking (SDN) has quickly emerged as a new networking paradigm that greatly changes the traditional network architecture. With the help of the concept of decoupling the control plane from the data plane, SDN uses the logically centralized controller to maintain a network-wide view and performs forwarding decisions to support fine-grained network management policies, which gives our full programming flexibility and reduces the complexity of network configuration and operation.

However, it's obvious that the centralized controller carries considerable overhead and would be easy to become a bottleneck. This leads to issues in both scalability and security. While there are many studies and solutions on the scalability issue, there is very little research on even more challenging security issues. For example, attackers may simply mount saturation attacks on the SDN controller by sending massive useless packets; consequently, the controller will handle every useless new packet for flow entry creation, which greatly occupies compute resources and overwhelms the controller.

Therefore, a practical DoS attack detecting application is needed to protect the SDN system from malicious users.

2. Background

2.1 Software Defined Network (SDN) Architecture

The layered decoupling architecture of SDN mainly consists of three layers: infrastructure layer, control layer, and application layer ("Software-Defined Networking: The New Norm for Networks - Open Networking Foundation", 2021). The upper two layers are responsible for the control function, and the lower layer is responsible for the data forwarding function. The connection between the control layer and the infrastructure layer is a southbound interface, which is currently implemented using the OpenFlow

protocol and is mainly responsible for the routing, access control, and security functions of the network. The connection between the control layer and the application layer is a northbound interface, which issues instructions to the applications, thereby making the application logic controllable (McKeown et al., 2008).

Such network architecture makes SDN different from traditional networks in the following ways. First, in traditional networks, the control and forwarding strategies of the network are implemented in one device, while SDN separates the control layer and forwarding layer. It greatly reduces the load of the network and improves forwarding efficiency. Second, in traditional networks, different manufacturers lead to differences in hardware and software of the equipment, which results in an isolated network, and only fixed unified standards can be used to communicate with others. SDN logically centralizes the control layer, controls the forwarding of the underlying hardware, monitors the status of the entire network, and realizes the ability to control the global network through the southbound and northbound interfaces.

In addition, the SDN network is programmable, with open and standardized interfaces. The interface includes two aspects. One is the northbound interface of the control layer and the application layer. Users can program multiple applications to access the SDN network through the northbound interface. The other is the southbound interface between the control layer and the forwarding layer. The unified standard of the interface eliminates the differences in equipment, and the controller does not need to monitor every forwarding device ("What is SD-WAN?", 2021, Haji et al., 2021, HumayunKabir, 2021).

2.2 OpenFlow

The earliest paper on OpenFlow was published in 2008 by a team led by Professor Nick McKeown of Stanford. Previously, because countless devices and protocols existed in networks, it was very difficult for new protocols to test or operate in a real network environment. And it was very impractical to persuade various hardware manufacturers to unify the specifications and standards of the production equipment. The OpenFlow switch came into being. The OpenFlow switch not only has the characteristics of high performance and low cost but also can complete a large range of experimental research.

The flow table is a very important concept in OpenFlow. In the traditional network architecture, the control and forwarding of the network are completed through the two-layer MAC address mapping table and the three-layer IP address routing table. OpenFlow also uses the same concept, except that the keywords of each layer are concentrated. The flow table is mainly composed of instructions and flow table entries to inform the switch on what to do with the data flow.

The OpenFlow channel provides a channel for the exchange of messages between the OpenFlow switch and the OpenFlow controller and transmits the instructions and data issued by the controller to the switch.

The OpenFlow protocol includes three message types: controller-to-switch, asynchronous and symmetric, each of which has several sub-messages. Controller-to-Switch is sent by the controller to the switch. Asynchronous messages are used to synchronize the switch and the controller and are initiated by the switch. Symmetric messages can be initiated by both parties of the interaction.

An OpenFlow switch refers to a switch that supports the OpenFlow protocol. The pipeline of the switch contains multiple flow tables. When the packet sent by the user is received by the OpenFlow switch, the switch will refer to the flow table for matching one by one. When matching, the OpenFlow switch will start with a flow table with the highest weight value. If a matching flow table entry is not found, the current data packet will be sent to the SDN controller, and the controller will then decide whether the data packet shall be discarded or passed to the subsequent flow tables ("OpenFlow Switch Specification Version 1.3.1 (Wire Protocol 0x04)", 2012, Chen et al., 2021).

2.3 Denial of Service (DoS) Attacks

Denial of Service (DoS) is an attack that uses existing hacking techniques to specifically target network protocol flaws, causing the target service system to stop service or be delayed. One of the DoS attack methods is flooding-based attacks, which mainly include UDP, TCP SYN, and ICMP.

Attackers send massive amounts of deceptive data packets to the target host, causing the target host to exhaust its computing resources, such as CPU, memory, and network bandwidth, and eventually, the entire network would be paralyzed.

In the OpenFlow protocol, if the OpenFlow switch receives a new packet and it has no relevant information in the flow table, the controller will not process matching but directly calculate the routing path. If the attacker pretends to be a switch and sends a large number of new packets to the controller, this will cause the controller to be busy calculating the routing path and consume a lot of resources and bandwidth, which would bring a serious impact on normal users (Kandoi & Antikainen, 2015).

For potential vulnerabilities in the SDN system, an attacker may launch a DoS attack or a Distributed Denial of Service (DDoS) attack on the controller. For example, an attacker may create a large amount of new traffic in a short period of time, requesting the controller so that normal users' requests cannot be responded to and causing network paralysis (Kandoi & Antikainen, 2015).

3. Methodology

3.1 Approach A Using Ryu Controller

Ryu is an open source SDN controller written in Python; it supports OpenFlow protocol and is easy to use with well-defined APIs. This approach uses Ryu as the SDN controller, along with the sFlow as a traffic monitoring method to realize the function of DoS attack detection, then uses Postman to send a flow table to drop packets. The detailed experimental process is shown below:

1. Launch the Ryu controller by running the provided python script.
2. Set up Mininet simulated network as below, with two hosts connected to one switch.
3. Set up sFlow Agent and monitor this switch; it should be shown as the following screenshot when no DoS attack occurs.
4. Create a flood attack on Mininet using the following command:
5. Check the sFlow again, and now the DoS attack can be found.
6. Launch the Postman and send a flow table to the switch to drop DoS traffic (ICMP packets).
7. Then check sFlow and DoS attack was successfully blocked.

3.2 Approach B Using POX Controller

POX is an open source SDN controller written in Python; it supports OpenFlow 1.0 protocol. This approach uses POX as the SDN controller, along with the hping3 as the DoS attacking method to implement the function of DoS attack detection. The controller will block the source IPs that have sent more packets than the threshold limits, which are set to be 50 in the demo below. The detailed experimental steps are shown below:

1. Running the topo script and opening any host's terminal by using the command xterm.
2. Launch the POX controller by running the DoS detection python script.
3. On the host, we run hping3 to send 45 packets to 10.10.1.3 with a fake src_ip. Since the packet count hasn't exceeded the threshold (50), this traffic won't be blocked.
4. When we run hping3 to send 65 packets to 10.10.1.3 with another fake src_ip, the traffic will be detected as malicious and blocked.
5. Similarly, sending a packet constantly with the same source IP over 50 times will also cause the IP address to be blocked.

4. Results and Discussion

The methods were then tested to see if they successfully passed the requirements proposed as the project aim; the following table shows the result of each test. Four out of five tests passed, and there is one requirement that failed to be fulfilled.

Table 1: Project Result

ID	Requirement	Description	Result
1.0	Ability of DoS attack detection	**Primary Functionality Requirement:** The system should be able to detect the DoS attack when it occurs.	**Pass**: This method can clearly indicate when a DoS attack occurs.
2.0	Ability of DoS attack mitigation	**Primary Functionality Requirement:** The system should be able to mitigate the DoS attack.	**Pass**: DoS attack can be blocked with attack packets being dropped.
3.0	DoS attacker identification	**Sub-Functional Requirement:** The system should be able to determine which switch the DoS attacker was sending packets through; this is also essential for the DoS attack mitigation.	**Pass**: This method can identify the switch under attack by monitoring each switch.
4.0	DoS attack type determination	**Sub-Functional Requirement:** The system should have the ability to distinguish different DoS attacks, including application layer flood, DDoS, and unintended DoS ("Types of Denial of Service Attacks - DOS Mitigation Strategies", 2021).	**Failed**: The method can not determine the DoS attack type.
5.0	Usage of Mininet	**Low-Level Requirement:** The Mininet should be used as simulation platform for the project.	**Pass**: Mininet was used for network setup and Dos attack simulation.

5. Conclusion

The main content of this project has the following aspects. The principle of the DoS attack is analyzed, and the software-defined network (SDN) architecture and OpenFlow protocol mechanism are briefly introduced. Two DoS attack detection methods are proposed and implemented, one of which is to configure the sFlow Agent on the virtual machine and then view the monitoring status in sFlow WebUI. The other is to automatically detect dos attack by counting whether the number of packets exceeds the threshold. Then, this project proposes two countermeasures for the detection of DoS attacks. One is to instruct the switch to drop DoS traffic by adding a flow table to the switch. The other is to stop the service.

This project has completed most of the project requirements. However, due to a lack of knowledge and an experimental environment, this project has the following limitations:

1. Although method 2 can automatically detect dos attacks, due to the diversity of network status and DoS attacks, simple counting packets instead of using flow table characteristics to determine whether the traffic is a DoS attack has a certain error.
2. Due to the experimental environment and software problems, the DoS detection algorithm was not simulated with the ONOS controller.

Although there are many detection and defense methods for DoS attacks in SDN, with the popularization and application of SDN networks, more new types of DoS attacks will certainly appear. Therefore, the research on SDN oriented DoS attack detection and defense methods are extremely valuable. Only by conducting in-depth research we can get a safer network environment.

Funding: This research received no external funding.
Conflicts of Interest: The authors declare no conflict of interest.
ORCID iD: Yalan Zhang https://orcid.org/0000-0001-8716-1306

References

[1] Chen, Y., Zhang, P, Zhou, S. (2019). Study on SDN Technology Based on OpenFlow and Its Application Prospect. Itm-conferences.org. Retrieved 29 August 2021, from https://www.itmconferences.org/articles/itmconf/pdf/2019/02/itmconf_icicci2018_01017.pdf.

[2] Haji, S., Zeebaree, S., Saeed, R., Ameen, S., Shukur, H., & Omar, N. (2021). Comparison of Software Defined Networking with Traditional Networking. *Asian Journal Of Research In Computer Science*, 1-18. https://doi.org/10.9734/ajrcos/2021/v9i230216

[3] HumayunKabir, M. (2021). *Software-Defined Networking (SDN): A Revolution in Computer Network*. Retrieved 29 August 2021, from http://www.iosrjournals.org/iosr-jce/papers/Vol15- issue5/Q0155103106.pdf.

[4] Kandoi, R., & Antikainen, M. (2015). *Denial-of-service attacks in OpenFlow SDN networks*. Ieeexplore.ieee.org. Retrieved 29 August 2021, from https://ieeexplore.ieee.org/abstract/document/7140489.

[5] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., & Rexford, J. (2008). OpenFlow. *ACM SIGCOMM Computer Communication Review,* 38(2), 69-74. https://doi.org/10.1145/1355734.1355746

[6] *OpenFlow Switch Specification Version 1.3.1 (Wire Protocol 0x04).* Opennetworking.org. (2012). Retrieved 29 August 2021, from https://opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.3.1.pdf.

[7] *Software-Defined Networking: The New Norm for Networks* - Open Networking Foundation. Open Networking Foundation. (2021). Retrieved 28 August 2021, from https://opennetworking.org/sdnresources/ whitepapers/software-defined-networking-the-new-norm-for-networks/.

[8] *Types of Denial of Service Attacks - DOS Mitigation Strategies. Developer.okta.com.* (2021). Retrieved 29 August 2021, from https://developer.okta.com/books/api-security/dos/what/.

[9] *What is SD-WAN?*. Ibm.com. (2021). Retrieved 29 August 2021, from https://www.ibm.com/services/network/sdn-versus-traditional-networking.