Parallel Section SEC. Security, Fault Tolerance

UDC 004

Artem Volokyta, Artem Dremov

METHODS AND SOLUTIONS FOR TRAFFIC MONITORING AND CONTROL IN SDN ARCHITECTURES

Abstract: The article aims to provide an analysis of modern computer network traffic monitoring solutions and technologies. In this article an overview of network traffic monitoring solution is provided, including general firewall designs and the use of firewalls in SDN networks. The article also touches on general architecture of SDN networks and SDN controllers. An example firewall solution application is presented in the article.

Keywords: firewall, cybersecurity, network security, traffic monitoring, SDN.

Introduction

While computer networks are an integral part of modern infrastructure, this has also made them susceptible to remote attacks, such as malware injections, DNS poisoning, DoS attacks, TCP hijacking, and other attacks. As such, the topics of network security and traffic monitoring remain some of the most crucial topics in cybersecurity. Additionally, emerging architectures and concepts, such as SDN networks, require individual solutions, that can provide the best results for a specific problem [1]. One of the most common techniques for traffic monitoring and control is a firewall, which allows to control incoming and outgoing traffic based on a table of security rules. These systems allow individual hosts or a controller host to block out malicious traffic while allowing communication between computers in the local network. However, traditional firewall solutions are proving hard to adapt in SDN environment. As such the topic of SDN network security remains one of the most forward-looking and important topics in the area of research [2].

Problem Definition

General problem explored in this article is of network security. In particular, the article explores subjects of defence against malware attacks, DoS attacks, DHCP, DNS attacks, etc. This is a problem prominent in modern day computer network infrastructure for organisations as well as networks designed for personal use. Coincidentally, the use of SDN networks is becoming more popular with businesses and for personal use. As such the problem of data security and traffic control in SDN networks is prominent and is the main topic of this article.

SDN network architecture

A typical SDN network consists of a number of hosts, switches (routers, other connection devices) and a controller device. SDN controller is software that typically runs on a unix based system. In a SDN network infrastructure we can define data plane, control plane and management plane (fig. 1).

Data plane is mainly responsible for the movement of data, in this case packets, in the network. The movement of packets is performed by a network device, such as a switch, that support openflow protocol. Unlike traditional networks, data plane devices do not implement logic on where to forward packets, as that is handled by the control plane.

Management plane is the part of the network that houses network applications. These applications can be running on a host device, or, occasionally, a controller. The applications allow host devices to communicate with the controller and can implement networking logic.

Control plane contains controller devices that implement the bulk of routing logic. A SDN controller is responsible for constructing and maintaining flow tables (similar to routing tables), ARP mapping, network devices discovery, STP protocols, etc. SDN controller is also responsible for communicating routing rules to the data plane, via southbound API (such as openflow protocol). The controller is also responsible for communication with management plane applications via northbound API, such as REST API, python API, programming languages such as frenetic and pyretic, and others [3]. The specific API's available depend on the controller software implementation. Popular SDN controller software includes opendaylight, floodlight, pox, nox, cisco ACI, vmware NSX, and others.

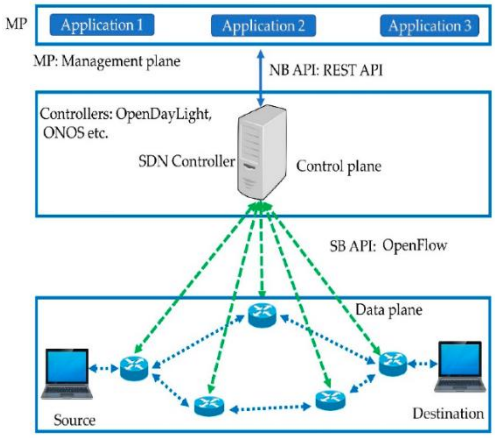

Fig. 1. Architecture of a SDN network

Additionally, the control plane can be centralised (with one controller), hierarchical (with multiple controllers organised in a hierarchy) or distributed (with multiple controllers that control different parts of the data plane).

For the topic of this article, we are mainly interested in management plane network applications for firewall implementation. The application itself may be hosted on the controller itself (fig. 2). As described in previous paragraphs, the SDN controller is responsible for constructing and maintaining the flow tables for the network. These flow tables are copied on to switches and are used to determine whether to drop or forward packets. The switch uses packet header data, such as destination ip, source ip, port, etc to match to flow table rules (fig. 3) [4]. Dropped packets, or packets that do not have a matching flow rule can be configured to be forwarded to the controller for inspection. As such it is possible to implement a variety of firewall designs, including flow-based firewalls, tcp firewalls. The controller can then update the flow tables for controllers.

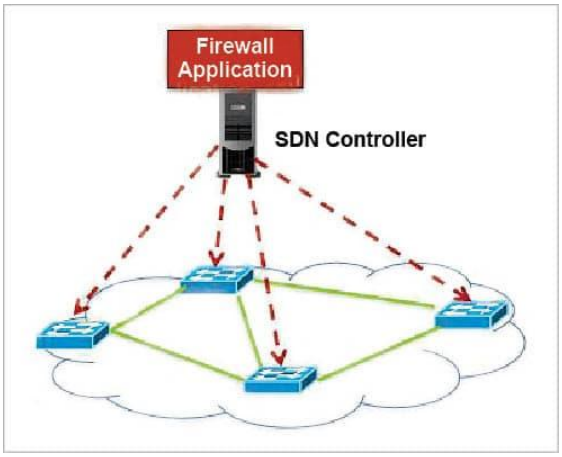

Fig. 2. A variant of integrating a firewall application in a SDN network

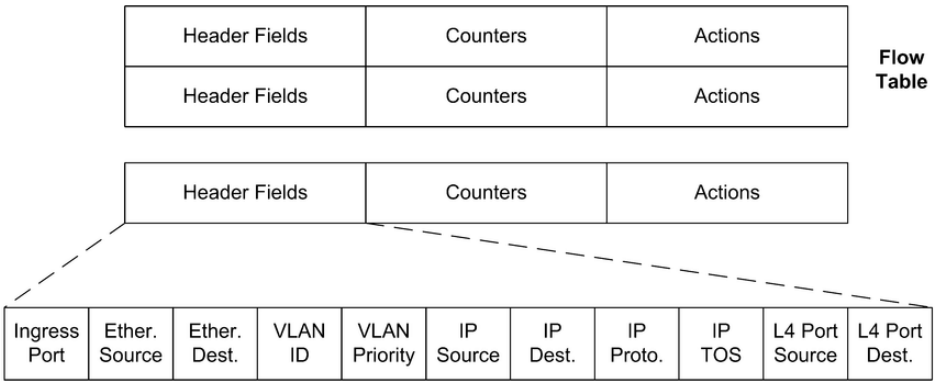

Fig. 3. Example of a flow table and packet header data used

Firewall design

A firewall is a system that is designed to monitor and control incoming traffic. In context of this article, we are primarily focusing on software-based firewalls. A software firewall as opposed to hardware firewall, can be deployed on any compatible device as an application. For example, in case of a SDN controller, a firewall can be a python-based application (in case of pox/ryu controllers) or Rest API and java application (odl, floodlight).

A simple implementation can operate as a packet filtering firewall. This design uses an access control list (ACL) table to match incoming traffic to the table rules. An example of an ACL table in figure 4. The controller can then construct a flow table based on incoming packets and their ACL table matches [5].

order	protocol	src_ip	src_port	dst_ip	dst_port	action
1:	tcp,	140.192.37.20,	any,	*.*.*.*,	80,	deny
2:	tcp,	140.192.37.*,	any,	*.*.*.*,	80,	accept
3:	tcp,	*.*.*.*,	any,	140.192.37.40,	80,	accept
4:	tcp,	140.192.37.*,	any,	140.192.37.40,	80,	deny
5:	tcp,	140.192.37.30,	any,	*.*.*.*,	21,	deny
6:	tcp,	140.192.37.*,	any,	*.*.*.*,	21,	accept
7:	tcp,	140.192.37.*,	any,	140.192.37.40,	21,	accept
8:	tcp,	*.*.*.*,	any,	140.192.37.40,	21,	accept
9:	tcp,	*.*.*.*,	any,	*.*.*.*,	any,	deny
10:	udp,	140.192.37.*,	any,	*.*.*.*,	53,	accept
11:	udp,	*.*.*.*,	any,	140.192.37.*,	53,	accept
12:	udp,	*.*.*.*,	any,	*.*.*.*,	any,	deny

Fig. 4. Example of an ACL table of a typical firewall

It is also possible to implement stateful packet inspection system. Stateful packet inspection is a technology that allows the firewall application to construct a state table of inspected packets. The state table allows the software to keep track of and log incoming connections (fig. 5). Incoming packets can be matched to the state table to be allowed through, according to the security policy. The state table entries can be deleted with appropriate packet for session end, or as a timeout. The benefits of this approach including traffic logging capabilities, more thorough packet inspection and finer control over incoming traffic. In context of SDN networks, it is possible to direct all incoming packets towards the controller for inspection. This, however, creates additional round trips from switch to controller back to switch. This can also be demanding of the controller’s device hardware. Alternatively, the Open vSwitch software has the capability to track incoming connections [4].

Source Address	Source Port	Destination Address	Destination Port	Connection State
192.168.1.100	1030	192.0.2.71	80	Initiated
192.168.1.102	1031	10.12.18.74	80	Established
192.168.1.101	1033	10.66.32.122	25	Established
192.168.1.106	1035	10.231.32.12	79	Established

Fig. 5. Example of state table of a stateful firewall design

Use cases and vulnerabilities

As discussed earlier, a SDN based firewall software can be used as a software-type firewall operating at layers 3 and 4 of OSI model, and can be implemented with features such as stateful inspection and tcp/openflow packet filtering. It is also possible to implement a solution with the usage of application firewall technology, which would allow the firewall software to perform deeper packet inspection at layer 7 of OSI model. Due to architecture of SDN controllers discussed earlier, it is also possible to implement additional applications, such as anti-malware software in order to inspect traffic payloads for possible malware, as well as load balancing and IPS/IDS solution. The firewall software can be used for centralized policy enforcement or as a distributed design [6][7][8]. A firewall can be implemented with one of northbound APIs, which allows it to be used on a variety of controllers, depending on API used. As such a SDN based firewall can be used in a wide variety of network architectures and topologies.

Overall, the main advantages of using firewalls in SDN networks come down to several factors. One of the main factors is the separation of control and data planes via SDN controllers, which allows to use a single firewall solution running alongside the controller to monitor the traffic a part of the network. This also allows the usage of simpler switches and other connection devices, that may be less susceptible to DoS attacks. Additionally, by running several controllers in a network, it is possible to divide the network into segments, where traffic between segments goes directly through the controller. This means that if a single segment gets compromised, the other segments can still be operational and can adjust security policies. Lastly, the use of software-based solution and network virtualisation, as well as programmable connection devices, allows for a highly scalable architecture, in particular in terms of firewalls designs.

Common attack vectors on SDN networks include switch device DoS attacks, spoofing attacks, for example, by imitating network switches, as well as man-in-the-middle attacks. Another vulnerability lies in the management plane applications, which, if compromised (for example with malware) can allow the insertion of forged rules. Another

big vulnerability of SDN networks is the importance of SDN controllers to the network's operation, since the controller implements most of the networking and packet forwarding logic [2]. This means that SDN controllers can be vulnerable to DoS attacks in particular, and a compromised controller can make the entire network inoperable. This, however, can be mitigated with hierarchical and distributed controller designs.

Example solution

For an example implementation, a mininet environment will be used. We will also be using POX controller and provided python API. To store the rules it is possible to use a python Dictionary, using values from packet header fields, such as destination IP address and source IP address as keys (fig. 3), and security policy, such as drop and forward as values. Alternatively, it is possible to store firewall rules in a separate configuration file instead. To check the rule, we can match one or several values of the packet header to the rules and determine the policy from there. It is then possible to employ openflow API to instruct the switch to forward or drop the packets as needed. Alternatively, it is possible to instruct the openflow API to forward packets to the controller for inspection by firewall software, or other network applications. For the topology a simple linear topology with 4 switches and 4 hosts was used (fig. 6). The rule inserted specifies that no traffic can be transmitted between hosts 1 and 2(fig. 7). Additionally, it is worth nothing that the implementation of a firewall is tied to controller's available northbound API. Since there is no common API for all available controller software, the firewall application cannot be deployed on certain controllers, for example, this implementation cannot be deployed on java/REST based controllers such as opendaylight or floodlight.

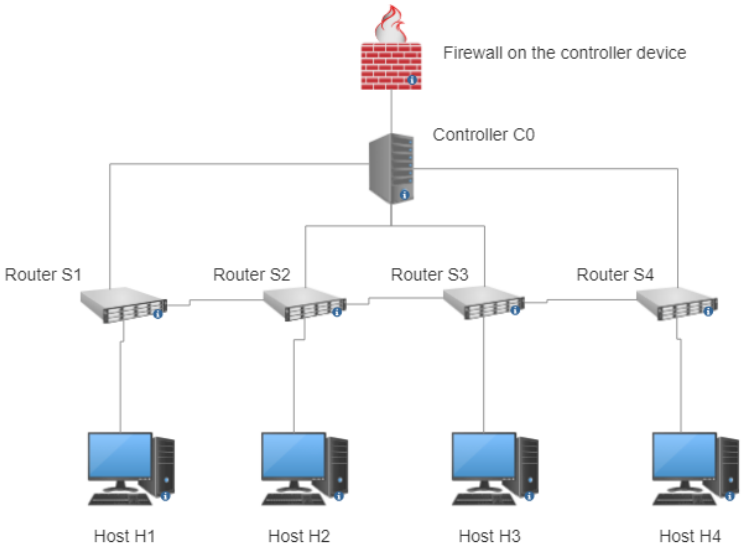

Fig. 6. Topology used for testing the example firewall

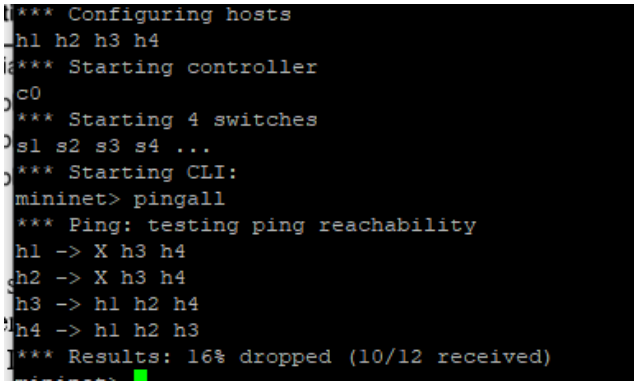

Fig. 7. Demonstration of packets being dropped via ping command using example implementation

Conclusions

In this article we have reviewed the problem of traffic monitoring and control and how it applies to SDN networks. We have reviewed the architecture of modern SDN networks. We have provided an analysis of firewall's role in a SDN network. SDN networks are becoming more and more widely used and this facilitates the adaptation of conventional firewall designs to SDN specific architectures. Additionally, SDN networks and controllers provide a wide range of tools, that can allow to implement additional security features, such as stateful packet inspection, distributed firewalls and application firewalls. SDN architecture also provides a number of potential security and network management advantage. We have also highlighted possible security vulnerabilities in SDN networks, that can be covered with usage of firewalls. We have reviewed the design of a typical software-based firewall. We have pointed out common firewall technologies that can be implemented with the use of modern SDN controllers.

Additionally, we have identified potential benefits and drawbacks of using SDN-based firewalls. We have identified potential use cases, including centralized and distributed network designs. We have also discussed potential benefits and vulnerabilities of reviewed firewall designs.

We have presented a simple implementation of a firewall on POX controller with Python API. The implementation can be configured to apply drop rule to packets between specified devices, or to apply forward to specific packets and drop all other packets. The example is used to demonstrate a way that a firewall can be implemented as a network application and to demonstrate the workings of a SDN-based firewall. It is possible to further extend the solution to implement additional features, such as stateful inspection and application layer design.

References

1. Mousa M., Bahaa-Eldin A., Sobh M. Software Defined Networking concepts and challenges // 2016 11th International Conference on Computer Engineering & Systems (ICCES). Cairo, Egypt, 2016. C.79-90. URL: https://ieeexplore.ieee.org/ document/7821979

2. Security Issues in Software Defined Networking (SDN): Risks, Challenges and Potential Solutions / M. Iqbal та ін. // International Journal of Advanced Computer Science and Applications. 2019. № 10 (10). C.298-302 URL: https://thesai.org/Downloads/Volume10No10/Paper_42-Security_Issues_in_Software_Defined_Networking.pdf

3. Paliwal M., Shrimankar D., Tembhurne O. Controllers in SDN: A Review Report // IEEE Access. 2018. № 6 (1). C.36256-36270 URL: https://ieeexplore. ieee.org/document/8379403

4. Braun W., Menth M. Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices // Future Internet. 2014. № 6 (2). C.302-336 URL: https://www.researchgate.net/publication/284696928_Software-Defined_Networking_Using_OpenFlow_Protocols_Applications_and_Architectural_Design_Choices

5. EFFICIENT ALGORITHMS AND ABSTRACT DATA TYPES FOR LOCAL INCONSISTENCY ISOLATION IN FIREWALL ACLS / S. Pozo та ін. // The International Joint Conference on e-Business and Telecommunications / Italy. Milan, 2009. C.42-53. URL: https://www.researchgate.net/publication/221436666_Efficient_Algorithms_and_Abstract_Data_Types_for_Local_Inconsistency_Isolation_in_Firewall_ACLs

6. Pena J., Yu W. Development of a distributed firewall using software defined networking technology // International Conference on Information Science and Technology (ICIST). N/A, 2014. C.449-452. URL: https://www.researchgate.net/ publication/286573159_Development_of_a_distributed_firewall_using_software_defined_networking_technology

7. Bannour F., Souihi S., Mellouk A. Distributed SDN Control: Survey, Taxonomy, and Challenges // IEEE Communications Surveys & Tutorials. 2017. № 20 (1). C.333-354 URL: https://ieeexplore.ieee.org/document/8187644

8. Obadia, M., Bouet, M., Leguay, J., & et al. (2014). Failover mechanisms for distributed SDN controllers. *2014 International Conference and Workshop on the Network of the Future (NOF)*, (1-6). Paris: IEEE. Retrieved from https://ieeexplore.ieee.org/ document/7119795

AUTHORS

Volokyta Artem – Ph. D., Associate Professor, Department of Computer Engineering NTUU “KPI”.

Scientific interests: real time systems, information security, distributed computations

Dremov Artem – student of the Department of Computer Engineering NTUU “KPI”.

Scientific interests: real time systems, cybersecurity, machine learning.