Building Stateful Firewall Over Software Defined Networking

Karamjeet Kaur and Japinder Singh

Abstract Current network architectures are ill suited to meet today's enterprise and academic requirements. Software Defined Networking (SDN) is a new way to Design, Build and Operate Networks. It replaces static, inflexible and complex networks with networks that are agile, scalable and innovative. The main idea is to decouple the control and data planes, allowing the network to be programmatically controlled. A key element of SDN architectures is the controller. This logically centralized entity acts as a network operating system, providing applications with a uniform and centralized programming interface to the underlying network. But it also introduces new security challenges. The challenge of building robust firewalls is the main challenge for protection of OpenFlow networks. The main problem with traditional firewall is that Network Administrator cannot modify/extend the capabilities of traditional vendor-specific firewall. Network Administrator can only configure the firewall according to the specifications given by the firewall vendor. To solve these problems we developed stateful firewall application that runs over SDN controller to show that most of the firewall functionalities can be built on software, without the aid of a dedicated hardware.

Keywords SDN · OpenFlow · Firewall · Packet filtering · Stateful firewall · Stateless firewall

1 Introduction

Firewall is used for preventing unauthorized access to and from the private network. Firewall examines all packets leaving or entering internal networks and block those who do not meet the defined security criteria. A firewall allows or rejects a specific

K. Kaur (✉) · J. Singh
Shaheed Bhagat Singh State Technical Campus, Ferozepur, India
e-mail: bhullar1991@gmail.com

J. Singh
e-mail: japitaneja@gmail.com

S.C. Satapathy et al. (eds.), *Information Systems Design and Intelligent Applications*, Advances in Intelligent Systems and Computing 434,
DOI 10.1007/978-81-322-2752-6_15

type of information. Implementation of a stateful firewall is the mandatory part of an effective information security program.

Most businesses and institutions deploy firewall as the main security mechanism. A traditional firewall is placed at the boundary of public and private network. It prevents attacks and unauthorized access by examining all incoming and outgoing traffic. In Traditional Firewall deployments, all Insiders in the private network are considered trustworthy. So internal traffic is not examined and filtered by the firewall. The assumption that insiders are trustworthy is not valid these days since insiders can perform attacks on others by bypassing security mechanisms. The main problem with traditional firewall is that they use dedicated hardware. That hardware is expensive and inflexible. This can be an even greater burden if a network needs more than one, as is common. Network administrators can not add new features since traditional firewalls are vendor locked, difficult to program, as discussed by the Hu et al. [1, 2].

To solve the problems of traditional firewall, we created SDN application which turned simple OpenFlow device into powerful firewall. Our firewall application runs on an SDN controller. To achieve our goal we used the Ryu controller which is implemented in Python. In this paper, we focus on implementation of robust stateful firewall. Firewalls are the most widely used security mechanism for OpenFlow based networks.

Our main contribution is the development and evaluation of stateful firewall solution for software defined network.

- To implement Stateful firewall to keep track the state of network connections passing through it. The stateful firewall is programmed to differentiate between packets from different types of connections. Firewall will allow only those packets which match with known connection state, others will be rejected.
- To Test our stateful firewall application in real environment.
- To compare our stateful firewall with stateless firewall using HTTP (Hyper Text Transfer Protocol), ICMP (Internet Control Message Protocol) traffic and observe that our firewall is more secure because it is able to block the fake packets as compared to stateless firewall.

The outline of our paper is as follow. Section 2 contains background and related work. Section 3 describes implementation details. Section 4 covers Experimental Evaluation and Sect. 5 contains conclusion and future work.

2 Background and Related Work

Current networks are very complex and hard to manage. These networks consists of different types of devices such as routers, switches, firewalls, network address translators, load balancers, and intrusion detection systems. These devices run software that are typically closed and vendor specific. New network protocols goes through years of standardization efforts and interoperability tests. Network administrators have to configure each individual network devices as specified by the

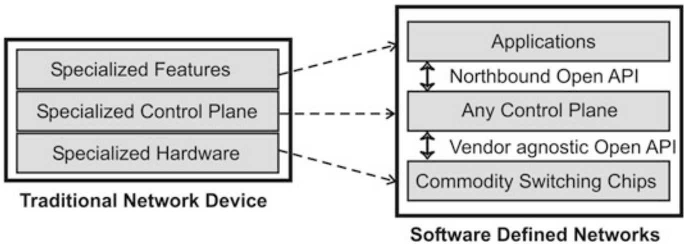


Fig. 1 Traditional and software defined network

vendor. Even more painful is that different products from the same vendor require different configuration. Traditional Networks has slowed down innovation, are very complex, and both the capital and operational expenses of running the network are high.

Mendonca et al. [3] discusses about the Software Defined Networking is emerging networking architecture in which the control plane is decoupled from data plane. Separation of control plane from data plane allows easier deployment of new applications and simplified protocol management as shown in Fig. 1. In SDN, the control plane is shifted into centralized place and shifted control plane is called SDN controller or OpenFlow controller. The network devices become simple packet forwarding devices that can be programmed via southbound interfaces such as ForCES (Forwarding and Control Element Separation), OpenFlow, as explained by Feamster et al. [4]. The research community and industry are giving significant attention to SDN [5]. The Open Network Foundation (ONF) has been created by service providers, network operators and vendors for promoting SDN and standardizing OpenFlow protocol. Although the concept of software defined networking is quite new, still it is growing at a very fast speed.

OpenFlow architecture basically consists of three components, OpenFlow switch, controller and Openflow protocol, as explained by Lara et al. [6]. OpenFlow protocol is used for communication between switch and controller through a secure channel. OpenFlow switch contains flow tables entries that consist of match fields, statistics, actions. When packet arrives at switch it is matched against rules. If match is found, corresponding action is performed. If no match is found, action is taken as specified in table miss entry. Controller is responsible for adding, updating or deleting flow entries, as discussed by Suzuki et al. [7].

Tariq et al. [8] successfully implemented layer 2 firewall by modifying layer 2 learning switch code in the POX controller. Limitation of this implementation is that this blocks only layer 2 traffic. Michelle et al. [9] Created simple user interface for the firewall. Their design looks at few header fields. Kaur et al. [10] implement stateless firewall that watch network traffic, and deny or drop packets based on source IP, destination IP, source MAC, destination MAC or other static parameters.

All previous work implemented stateless firewall that treats each packet in isolation. Stateless firewall is not able to determine whether packet is attempting to establish new connection or is a part of existing connection or just a fake packet.

Second problem of these implementations is that they were tested using only ICMP traffic. TCP traffic was not used for testing. To solve these problems, we design and implement a stateful firewall that is more secure than the stateless firewall.

3 Implementation

Firewall is used for preventing unauthorized access to and from the private network based on packet filtering rules. Firewall examines all packets leaving or entering internal networks and block those who do not meet the defined security criteria. When configuring filter rules for TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) services such as File Transfer Protocol (FTP), Trivial File Transfer Protocol (TFTP), http, we have to allow traffic in both directions since these services are bi-directional. A TCP or UDP session involves two players, one is the client that initiates the session and the other is the server hosting particular service. As an example, Fig. 2 the rules are added in such a way so that traffic can cross the firewall between web client with address 192.168.0.2/24 and web server with address 172.16.0.31/16.

The server IP address, the client IP address, the server port also known as destination port and client port known as source port are the 4 attributes that defines a TCP or UDP session. Generally, the server port helps in identifying the type of service that is being offered. For example, port 80 is associated with web service and ports 20, 21 are associated with ftp service. Client port which is mostly greater than 1023 is dynamically chosen by client host's operating system. It also means that client's ports are mostly unpredictable and firewall has to allow all the source ports so that session can be successful. As a result, this type of approach introduces serious security problem. It enables malicious hosts to launch Denial of Service (DOS) attack by flooding the servers with unwanted traffic. The Fig. 2 shows network architecture used in experiment.

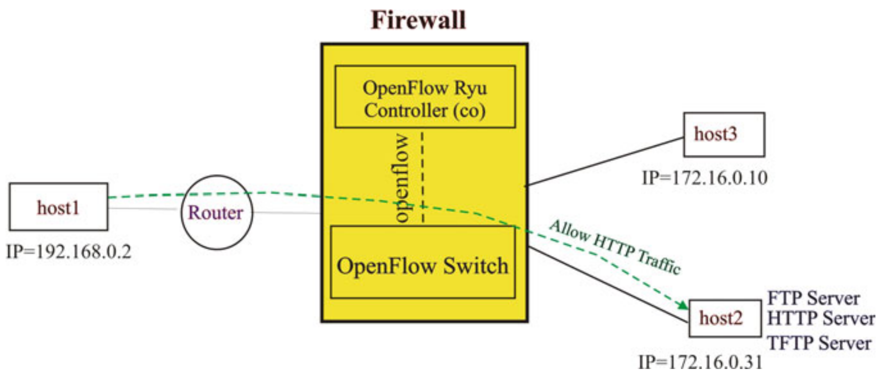


Fig. 2 Network topology

For the experimental evaluation we create a real topology which consists of 4 computers. We implement the OpenFlow switch and Ryu controller on one computer. On second computer we are implemented apache web server, FTP server and TFTP server. Other two systems act as clients on which we implemented Ostinato packet generation tool. Ryu controller is started and connectivity between the openflow network and controller is verified. OpenFlow switch and controller exchange series of messages for connection establishment and setup. We have got two hosts host1 with IP address 192.168.0.2/24 and a host2 with IP address 172.16.0.31/16 connected using of the OpenFlow switch.

3.1 Stateful TCP Packet Filtering

Stateful firewall handles the security issue by enabling connection tracking. The connection tracking keeps track of established TCP sessions. Firewall maintains entries of open TCP sessions in cache or in connection tracking table. An entry contains information regarding server ip address, client ip address, server port and client port. The firewall administrator does not have any information regarding client port at the time of configuration of rules. But at the time of connection setup, both server and client port information is available in TCP header. Stateful firewall allows traffic in both directions for packets belonging to existing TCP connection.

After connection establishment, the decision to allow or block subsequent packets will be based on contents of connection tracking table. When a subsequent packet reaches the firewall with flag ACK set and flag SYN unset, its information entry is checked in connection tracking table. If entry exists, the packet is allowed to pass through immediately. The packet gets rejected if no such entry is found.

On the completion of 3-way handshake, TCP connection state turns to ESTABLISHED state. On connection termination, the entry is removed from the connection tracking table. If TCP connection is inactive for a long time, a timeout value is kept to flush out the inactive entries from connection tracking tables and thus blocking the connection.

3.2 Stateful ICMP Packet Filtering

It is easy to track ICMP sessions. It involves 2 ways communication. For every response ICMP message, there should be corresponding ICMP request message. ICMP tracking is done on basis of Sequence number, Type Identifier, source address, destination address of reply and request messages. The sequence number and type identifier cannot alter in an ICMP session when returning a message to the sender. The sender matches each echo request with echo request by using these. These parameters should have the same values for echo request and echo reply. This is the only way for tracking ICMP sessions.

Upon receiving ICMP request packet, an entry is made in connection tracking table by stateful firewall. Stateful firewall will accept the echo reply if parameter value are the same as in request packet. Fake ICMP reply will be rejected since connection tracking table does not contain any entry the same parameter values.

4 Experimental Evaluation

The purpose of this evaluation is to ascertain the functionality of our Stateful firewall application. The key components involved for experimental setup are, Ryu OpenFlow Controller, OpenFlow virtual switch, Ostinato packet generation tool.

Ryu OpenFlow Controller
Monaco et al. [11] discusses about the Ryu controller that is a component-based OpenFlow controller in Software Defined Networking [11]. Ryu provides well defined Application programming interface that make it easy for developers to create new network control applications like Firewall, Load Balancer. Ryu supports different protocols for controlling network devices, such as OpenFlow, Netconf, OF-config, etc. Ryu supports OpenFlow versions 1.0, 1.2, 1.3. The code of Ryu is available under the Apache 2.0 license. Ryu is completely written in Python language.

OpenFlow Virtual Switch
An OpenFlow switch is a virtual or physical switch that forwards packets in a software defined networking (SDN) infrastructure. OpenFlow switches are either pure SDN switches or hybrid switches as explained by Bianco et al. [12].

Ostinato
Ostinato is GUI based network packet analyzer and generator tool. It is open source and cross platform tool. By using it, we can create and send several streams of packets at different rates having different protocols as explained by Botta et al. [13], Srivastava et al. [14].

Wireshark
Orebaugh et al. [15] discusses about the Wireshark that is a best available network protocol analyzer tool that is open source and multi-platform. It permits you to analyze information from stored captured file or using live network. Captured information can be scanned interactively, as explained by Sanders et al. [16].

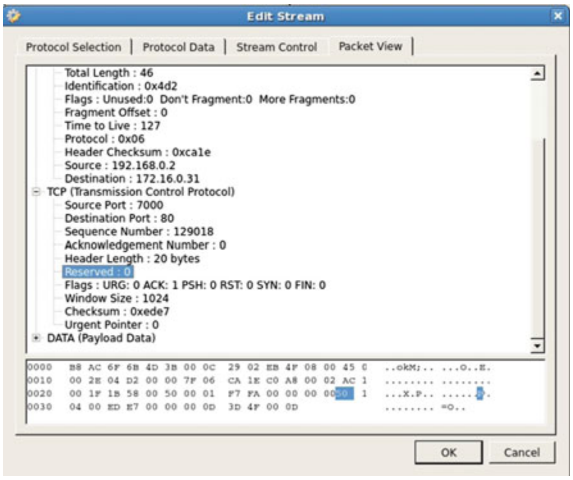
The experiment includes two parts.

4.1 Stateful TCP Packet Filtering Testing

(1) We tested number of scenarios on firewall application. In first scenario we applied rule in which host1(“192.168.0.2/24”) was allowed to access the web

Table 1 Three-way handshake of web session

Source IP	Destination IP	Source port	Destination port	SYN	ACK
192.168.0.2	172.16.0.31	35591	80	1	0
172.16.0.31	192.168.0.2	80	35591	1	1
192.168.0.2	172.16.0.31	35591	80	0	1



```
Total Length : 46
Identification : 0x4d2
Flags : Unused:0  Don't Fragment:0  More Fragments:0
Fragment Offset : 0
Time to Live : 127
Protocol : 0x06
Header Checksum : 0xca1e
Source : 192.168.0.2
Destination : 172.16.0.31
TCP (Transmission Control Protocol)
  Source Port : 7000
  Destination Port : 80
  Sequence Number : 129018
  Acknowledgement Number : 0
  Header Length : 20 bytes
  Reserved : 0
  Flags : URG: 0 ACK: 1 PSH: 0 RST: 0 SYN: 0 FIN: 0
  Window Size : 1024
  Checksum : 0xede7
  Urgent Pointer : 0
DATA (Payload Data)

0000   B8 AC 6F 6B 4D 3B 00 0C  29 02 EB 4F 08 00 45 0   ..okM;.. ...O..E.
0010   00 2E 04 D2 00 00 7F 06  CA 1E C0 A8 00 02 AC 1   ........ ........
0020   00 1F 1B 58 00 50 00 01  F7 FA 00 00 00 0050  1   ...X.P.. ......P.
0030   04 00 ED E7 00 00 00 0D  3D 4F 00 0D              ........ =O..
```

Fig. 3 Fake TCP packet generated by ostinato packet builder

server at host2 (“172.16.0.31/16”). This means that web traffic between host2 web server and host1 web client is allowed.

(2) At host2, wireshark has been used for capturing 3 way TCP handshake packets during web session. Table 1 represents the three-way handshake web session.

(3) At Ostinato Packet Builder was used for sending a fake TCP packet from host1 to host2 as shown in Fig. 3. The fake packet pretended that connection to TCP port 80 is already established (SYN = 0, ACK = 1). The fake packet was having a source port that was different from the source port of current active session as shown in Table 2.

(4) Wireshark at host2 was not able to capture the fake packet sent by host1. This happened because fake TCP packet was blocked by the stateful firewall. Stateful firewall blocks TCP packets which do not belong to established TCP sessions. But in case of stateless firewall the fake packets are reached at web server. This means that stateful firewall is more secure than stateless firewall.

4.2 *Stateful ICMP Packet Filtering Testing*

(1) We tested number of scenario on firewall application. In this scenario we applied rule that allow host2 (“172.16.0.31/16”) to ping host1 (“192.168.0.2/24”).

Table 2 Fake TCP packet parameters

Source IP	Destination IP	Source port	Destination port	SYN	ACK
192.168.0.2	172.16.0.31	7000	80	0	1

Table 3 ICMP exchange packet parameters

Source IP	Destination IP	Type	Identifier	Sequence number
172.16.0.31	192.168.0.2	8	314	1
192.168.0.2	172.16.0.31	0	314	1

Table 4 Fake ICMP echo reply packet

Source IP	Destination IP	Type	Identifier	Sequence number
192.168.0.2	172.16.0.31	0	314	10

(2) Wireshark at host2 is used for capturing ICMP packets. Table 3 show the ICMP exchange packets between the two systems.

(3) Ostinato Packet Builder is used for sending a fake ICMP echo reply packet from host1 to host2. This fake packet pretended that ICMP echo request packet was received from host2 previously. The fake packet contains different sequence number as shown in Table 4.

(4) Wireshark at host2 is not able to capture the fake ICMP packet sent by host1. This happened because fake ICMP packet was blocked by the stateful firewall. Stateful firewall blocks fake ICMP echo reply packet. This means that stateful firewall is more secure than stateless firewall.

After stateful firewall implementation, Fig. 4 shows that latency got increased which means that firewall provide security with little more overhead than stateless firewall.

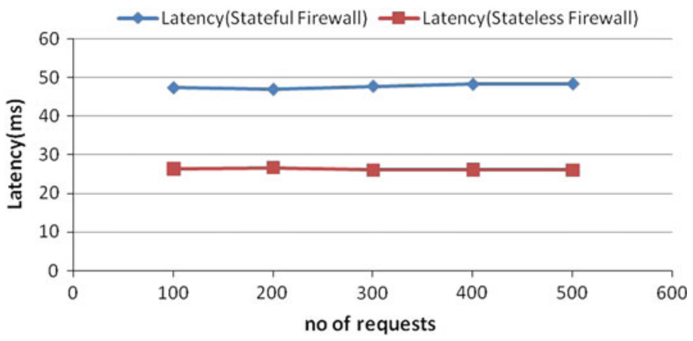


Fig. 4 Latency in Stateful and stateless firewall

5 Conclusion and Future Work

The popularity of SDN is increasing day by day. Although there exist many quality firewalls but they are very costly. Another limitation is that Network Administrator cannot modify/extend the capabilities of the traditional vendor-specific firewalls. They can only configure the firewall according to the specification given by the firewall vendor. We have implemented a stateful firewall and compared it with the stateless firewall and observed that our firewall is more secure (means able to block fake packets) than the stateless firewall. Our firewall is also able to block the SYN flooding attack by keeping a record of each connection.

Future direction can be to design and implementation Intrusion Detection system and combine it with our firewall for creating Intrusion Prevention system.

Acknowledgments We thanks Mr. Vipin Gupta of U-Net Solutions, Moga, India for his valuable help.

References

1. Hu, Hongxin, Wonkyu Han, Gail-Joon Ahn, and Ziming Zhao. "FLOWGUARD: building robust firewalls for software-defined networks." In *Proceedings of the third workshop on Hot topics in software defined networking*, pp. 97–102. ACM, 2014.
2. Hu, Hongxin, Gail-Joon Ahn, Wonkyu Han, and Ziming Zhao. "Towards a Reliable SDN Firewall." *Presented as part of the Open Networking Summit 2014 (ONS 2014)}* (2014).
3. Mendonca, Marc, Bruno Astuto A. Nunes, Xuan-Nam Nguyen, Katia Obraczka, and Thierry Turletti. "A Survey of software-defined networking: past, present, and future of programmable networks." *hal-00825087* (2013).
4. Feamster, Nick, Jennifer Rexford, and Ellen Zegura. "The road to SDN: an intellectual history of programmable networks." ACM SIGCOMM Computer Communication Review 44, no. 2 (2014): 87–98.
5. N. Feamster, "Software defined networking," Coursera, 2013. [Online]. Available: https://class.coursera.org/sdn-001.
6. Lara, Adrian, Anisha Kolasani, and Byrav Ramamurthy. "Network innovation using openflow: A survey." (2013): 1–20.
7. Suzuki, Kazuya, Kentaro Sonoda, Nobuyuki Tomizawa, Yutaka Yakuwa, Terutaka Uchida, Yuta Higuchi, Toshio Tonouchi, and Hideyuki Shimonishi. "A Survey on OpenFlow Technologies." IEICE Transactions on Communications 97, no. 2 (2014): 375–386.
8. Javid, Tariq, Tehseen Riaz, and Asad Rasheed. "A layer2 firewall for software defined network." In *Information Assurance and Cyber Security (CIACS), 2014 Conference on*, pp. 39–42. IEEE, 2014.
9. Suh, Michelle, Sae Hyong Park, Byungjoon Lee, and Sunhee Yang. "Building firewall over the software-defined network controller." In *Advanced Communication Technology (ICACT), 2014 16th International Conference on*, pp. 744–748. IEEE, 2014.
10. Kaur, K.; Kumar, K.; Singh, J.; Ghumman, N.S., "Programmable firewall using Software Defined Networking," *Computing for Sustainable Global Development (INDIACom), 2015 2nd International Conference on*, vol., no., pp. 2125, 2129, 11–13 March 2015.
11. Monaco, Matthew, Oliver Michel, and Eric Keller. "Applying operating system principles to SDN controller design." In *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, p. 2. ACM, 2013.

12. Bianco, Andrea, Robert Birke, Luca Giraudo, and Manuel Palacin. “Openflow switching: Data plane performance.” In *Communications (ICC), 2010 IEEE International Conference on*, pp. 1–5. IEEE, 2010.
13. Botta, Alessio, Alberto Dainotti, and Antonio Pescapé. “A tool for the generation of realistic network workload for emerging networking scenarios.” Computer Networks 56, no. 15 (2012): 3531–3547.
14. Srivastava, Shalvi, Sweta Anmulwar, A. M. Sapkal, Tarun Batra, Anil Kumar Gupta, and Vinodh Kumar. “Comparative study of various traffic generator tools.” In *Engineering and Computational Sciences (RAECS), 2014 Recent Advances in*, pp. 1–6. IEEE, 2014.
15. Orebaugh, Angela, Gilbert Ramirez, and Jay Beale. *Wireshark & Ethereal network protocol analyzer toolkit*. Syngress, 2006.
16. Sanders, Chris. *Practical Packet Analysis: Using wireshark to solve real-world network problems*. No Starch Press, 2011.