ESKİŞEHİR TECHNICAL UNIVERSITY JOURNAL OF SCIENCE AND TECHNOLOGY
A- APPLIED SCIENCES AND ENGINEERING

2018, 19(3), pp. 555 - 564, DOI: 10.18038/aubtda.421939

IMPROVING SOFTWARE DEFINED NETWORK SECURITY VIA SFLOW AND IPSEC PROTOCOL

Babatunde Hafis LAWAL [1] , Nuray AT [1, *]

[1] Electrical and Electronics Engineering, Anadolu University, Eskişehir, Turkey

ABSTRACT

Software Defined Network (SDN) has found its footprints in modern networking practices thanks to its abstraction of the control plane from the infrastructural plane and its ability to enhance programmability in networking. Despite its aptness, security is still a major concern for this technology. This study proposes a secure method for the SDN network based on the sFlow and the IPsec protocol. The proposed method ensures real-time detection and mitigation of attacks such as Distributed Denial of Service (DDoS) attacks, Man in the Middle attacks (MITM), replay attacks, etc. on the SDN network. To prove the effectiveness of the proposed method, the SDN network was emulated on Mininet and analyzed. It was shown that attacks were detected and curbed early on the network before any harm could be done to the network.

Keywords: Software Defined Network (SDN), Mininet, sFlow, IPsec, DDoS

1. INTRODUCTION

The ability to abstract the control functions from the forwarding elements in networking has led to the emergence of Software Defined Network (SDN). Network administrators see SDN as a breakthrough by having total remote access and control of the network while network providers and operators believe that SDN will reduce capital expenditure (CAPEX), operational expenditure (OPEX) and energy consumption. Their arguments are all true as SDN allows the programmability of the network, modification of routing rules or data flows according to real-time necessities, monitoring the network performance from a dashboard and abstraction of the control plane from the data/infrastructural plane. Other benefits include implementation of new network services such as quality of service (QoS), enforcement of new policies, access control, traffic engineering, security enhancement, bandwidth management, etc.

SDN also solves the issue of proprietary that exists in traditional networks. It has proven to be the required technology to make networking more scalable, dynamic and it removes the data plane devices vendor proprietary [1]. SDN abstracts the control functions from the switches or routers in contrast to traditional networks, rendering the data plane devices as merely forwarding agents. The forwarding rules are logically generated by the controller and are communicated to the switches through the OpenFlow protocol [2, 3]. OpenFlow facilitates the communication process between the logically centralized software controller and the network forwarding devices; thus, making it possible to implement and deploy SDN technology in current networks [4]. The OpenFlow controller provides a centralized overview of the network with the ability to detect network vulnerabilities and intrusions, and implement security policies [5]. Figure 1 shows the SDN architecture.

*Corresponding Author: nat@anadolu.edu.tr
Received: 08.05.2018 Accepted: 13.06.2018

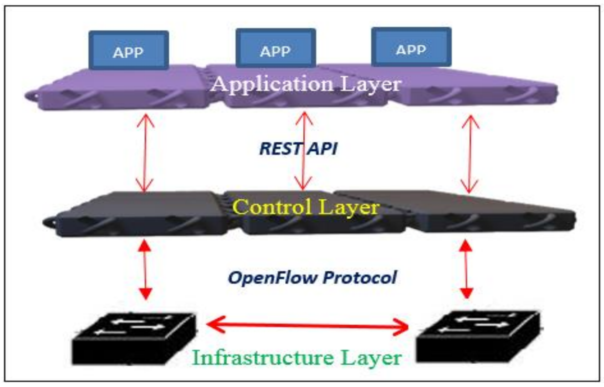


Figure 1. Software Defined Network Architecture

Regardless of the intelligence and aptness associated with SDN, security vulnerabilities remain a bottleneck in its deployment. Since SDN is an active network [4], it suffers from the challenges facing active networks, one of which is securing nodes from malicious injections. Active networks are programmable and allow real-time modification of network parameters based on necessities. They are generally prone to threats such as Distributed Denial of Service (DDoS) Attacks, Man in the Middle Attacks (MITM), Replay Attacks, etc.

DDoS attack has been on the rise ever than before. This attack aims to overwhelm the data and/or control resources of SDN network and render the network unavailable to eligible users. On the other hand, an MITM attack causes lack of integrity on the transmitted data, that is, an unauthorized user may alter or modify packets in transit. The growth in the rate of connected devices (Internet of Things) has also contributed to the exacerbation of the susceptibility of the network. The controller is one of the important components in the SDN network, if it is compromised then the whole network could be brought down. This has called for urgent defense mechanisms against these threats.

In this work, a defense mechanism that can detect and protect the SDN and its resources from the above mentioned vulnerabilities is proposed. The proposed method detects and prevents attacks before they harm or bring down the network. Specifically, the proposed method protects the SDN network against DDoS attacks, MITM attacks, etc. The system uses the sFlow technology and Internet Protocol Security (IPsec) protocol as a defense strategy. sFlow detects abnormalities in packet transmission by sampling fractions of all packets transiting the network. In case of anomalies in packet flow, sFlow generates handling rules and communicates with the controller for flow rule adjustment on the forwarding devices. IPsec on the other hand ensures that only authorized users are allowed on the network through an authentication protocol. It also ensures integrity on the network by encrypting the transmitted data and encapsulating the whole packets.

This paper is organized as follows: In Section 2, related works on the subject are reviewed. Section 3 explains how the sFlow technology and IPsec protocol can secure the network. In Section 4, the experimental model is setup and its performance is evaluated while Section 5 concludes the study.

2. RELATED WORKS

Many studies [1-5] have been carried out on SDN and its security issues, and many are ongoing. In the following, some of these studies and proposed techniques for detecting and mitigating vulnerabilities on the SDN network are reviewed.

In [6], Kaur et al. designed and applied an OpenFlow based firewall application using the open source POX controller based on python programming to secure the network. However, this design has two drawbacks: one is the firewall overload on the controller and the other one is the use of memoryless (stateless) firewall. Note that stateful firewalls keep information about past events which helps them to speed up decision making on the network against future threats. Kandoi et al. in [7] argued that improving configuration parameters may alleviate the harmful effects of DoS attacks. They claimed that configuring an optimal idle timeout value and flow aggregation would prevent the switch flow table from being overwhelmed. However, this is not the case if the attack is launched from multiple sources as in DDoS attacks. In [8], Haining Wang et al. setup a mechanism in a leaf router which connects the host to the internet. This method consumes many resources and much time to detect flood attacks because the detection mechanism can only be applied to the leaf router.

Bing Wang in [9] presented a graph model-based attack detection and mitigation system that stores (or, records) known traffic patterns as a relational graph between patterns and their labels to distinguish between normal and abnormal traffic. The prevention technique was based on blocking the source IP of the attack sending all packets to the controller via a path based on the graph model. The IPsec protocol and its applications were studied in [10-12]. In [13], security issues of SDN network were discussed and a security measure using IPsec was proposed to protect the traffic between hosts.

The required software (Mininet) to emulate an SDN network prototype is discussed in [14, 15]. Besides, [16-18] explains the basic components of the SDN network such as the OpenFlow switch and the Floodlight controller used in the experimental setup.

Unfortunately, none of the above mentioned methods was able to secure the SDN network in real-time. Motivated by the limitations of the existing works in the literature, we propose a method that is able to detect and prevent attacks in real-time and ensures integrity of the traffic transiting the network.

3. SECURITY IMPLEMENTATION

The use of sFlow technology and the IPsec protocol methodology is proposed and implemented to curb the security vulnerabilities and challenges facing the SDN network. The following subsections introduce the technology employed to secure the SDN network and the proposed secure SDN architecture.

3.1. sFlow Technology

sFlow is an open source statistical time-based real-time traffic sampling technology for monitoring traffic in data networks at high speed. Fractions of all packets transiting the network are collected by the sFlow agent(s) and are forwarded to the collector for analysis. The analyzer generates and communicates handling rules to the controller if it detects any anomalies on the network. The sFlow system architecture is given in Figure 2.

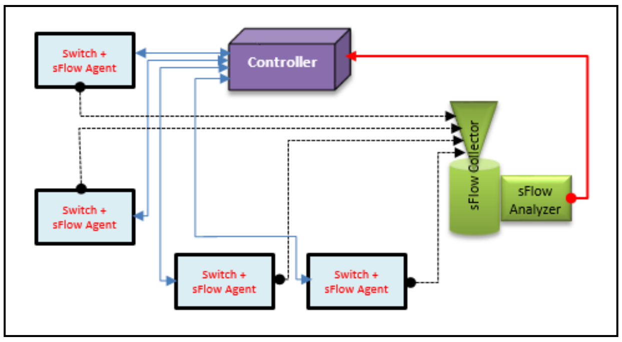


Figure 2. sFlow System Architecture

sFlow system architecture consists of:

The sFlow Agent: This can either be embedded in a switch or a router, or in a standalone probe. It takes samples from every packet transiting the network and sends as a datagram to a collector.

The sFlow Collector: It serves as the server where sFlow datagrams are collected and stored.

The sFlow Analyzer: It analyzes received datagrams and provides real-time or historical overview of the network traffic flow. This provides in-depth information about the network parameters and can be used to detect irregularities such as denial of service attacks, virus attacks, etc. on the network.

Samples are collected from all switches on the network by the sFlow agents which are pre-installed on them. The rate at which samples are collected is realized by assigning sampling rates and polling intervals. Sampling rate (S) is the fraction of packets that should be taken and sent for sampling from every packet transiting the network while polling intervals (I) is the time difference between successive sample collections. In this study, S is set to be 20 and I to be 30, i.e., at every interval of 30 seconds, one out of every 20 packets transiting the network will be forwarded to the collector for sampling.

Similarly, a threshold value T can be set to ensure that the number of packets per second, P, flowing through the switch(es) are below the threshold value. This can be done by starting the mitigation script usually written as an Application Program Interface (API) in JavaScript or Python and communicating to the controller through the Representational State Transfer (REST) protocol.

3.2. Internet Protocol Security (IPsec)

IPsec is a set of protocols and algorithms that is used to ensure a secure and private communication at the network layer in a public network using cryptographic keys. It is usually setup on the switches or routers and allows transmitted packets to be authenticated, encrypted and encapsulated. In addition, it negotiates between/among the gateways by checking the configuration in the network. If the keys are different during negotiation, the network will not be established [10]. IPsec ensures authentication between hosts and protects traffic between the gateways, or between the gateway and a host. It supports network-level data integrity, confidentiality, data authentication, and protects against replay attacks, man in the middle attacks, etc.

3.2.1. Flow-Based IPsec

In order to enhance SDN security, there is a need to introduce IPsec into the network. However, the traditional IPsec is too rigid to be implemented on the SDN network since it requires manual configuration. The flow-based IPsec protocol [11] is an SDN controlled security protocol which functions like the traditional IPsec with support for SDN services. The flow-based IPsec supports and creates a secured path for both host to gateway and gateway to gateway transmission [12]. Unlike the traditional IPsec, the flow-based IPsec can be programmatically built on the application layer and it communicates with the controller through the REST protocol. The controller then translates the security requirements and implements them on the forwarding agents.

The flow-based IPsec is a controller-based data protection service which allows the protection of data traffic based on predefined security policies. The controller establishes and manages the IPsec security associations by creating the required parameters for key negotiations. Some of the major algorithms IPsec uses for securing the network are shown in Table 1.

Table 1. Major IPsec Algorithms

Security Mode	Algorithm
Negotiation	DH series
Authentication	MD5, SHA1
Encryption	DES, 3DES, AES

The security rules do not need to be run on the gateways directly, they are automated by the controller and once implemented they do not need to be repeated. This reduces the overload and makes the implementation lighter.

3.2.2. IPsec Protocol Schemes and Modes

IPsec uses two protocols [13], namely, Authentication Header (AH) and Encapsulation Security Payload (ESP). They both provide authentication and data integrity. But in addition, ESP provides confidentiality on the transmitted data by encrypting the data payload and/or IP header. The level of encryption however depends on the adopted mode which can either be IPsec Tunneling or IPsec Transport Mode.

IPsec Transport Mode encrypts only the data payload while the IP header remains open. However, IPsec Tunnel mode encrypts both the data payload and the IP header. It wraps the whole encrypted traffic with a new IP header before transmission. Table 2 and Figure 3 give details on the IPsec protocols and modes and how they protect the network. Since SDN allows the programmability of the control layer, both protocols and modes can be deployed and automated by the controller based on the requirement and necessities of the network. This will allow SDN to maximize the features associated with each of the protocol.

Table 2. IPsec Protocols and Modes

Protocol / **Mode**	**AH**	**ESP**
Transport	AH Transport Mode Packet	ESP Transport Mode Packet
Tunnel	AH Tunnel Mode Packet	ESP Tunnel Mode Packet

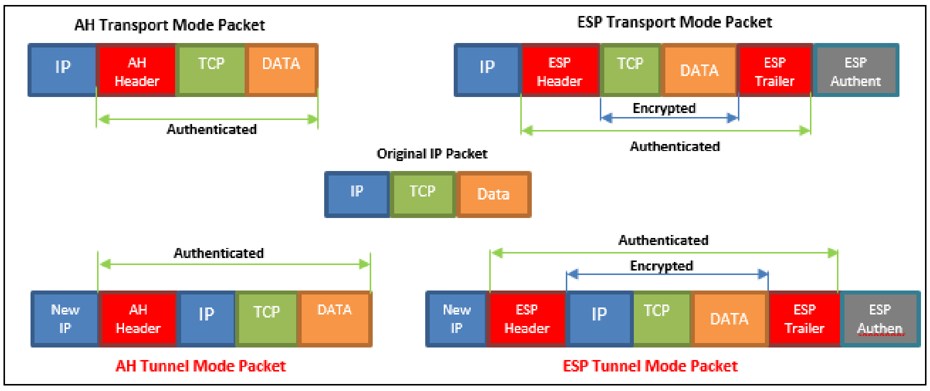


Figure 3. IPsec Data Payloads

A network administrator can define rules via the application layer in the SDN network according to the importance attached to the traffic and the type of the network (e.g., host to gateway or gateway to gateway). For instance, transport mode is good for host to gateway transmission while tunnel mode is more secured and appropriate for gateway to gateway transmission.

3.3 A Secured SDN Architecture

The proposed architecture employs the sFlow technology and IPsec protocol to improve the security of the SDN as seen in Figure 4. IPsec encapsulates the links between hosts and switches, and all packets in transit are protected from unauthorized users from tampering or modifying the data in transit. By doing this, the network is protected against MITM and replay attacks while sFlow detects and protects the network against DDoS attacks.

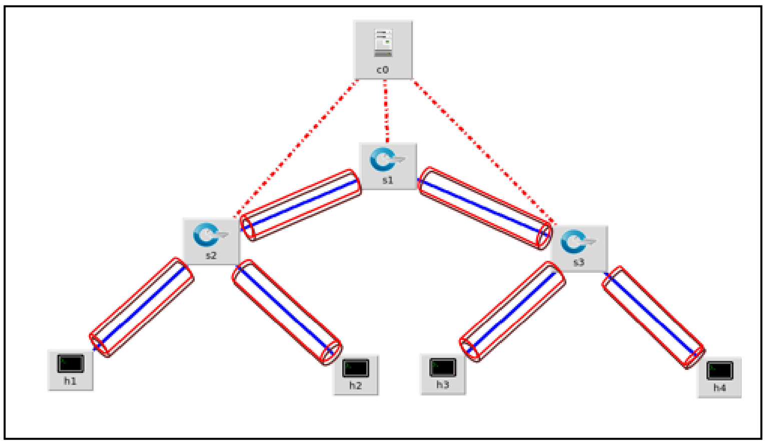


Figure 4. A Secured SDN Architecture

4. EXPERIMENTAL SETUP AND EVALUATION

In this section, SDN network is setup and some scenarios have been created to emulate possible security vulnerabilities on the network. The proposed security measure is later deployed to protect the network and its performance is evaluated in terms of its effectiveness. The SDN network is emulated on Mininet. Mininet is an open source emulator software which uses real devices that run real applications on a virtualized hardware (e.g., Oracle VM VirtualBox) on a single computer [14]. It has the capabilities that enable researchers or network programmers to create SDN prototype in a simple manner and to interact, customize and share the emulated network. It also provides a conducive environment for running the prototype on hardware in real life situations [15].

A Casper PC with Processor Intel (R) Core (TM) i5 CPU @ 2.50GHz, (RAM): 8.00 GB (7.5GB usable), Operating System: Windows 8.1, 64-bit, Virtual Machine: Oracle VM Virtual Box version 5.2.6, Guest Operating System: Mininet Emulator version 2.2 on Ubuntu 14.04.4 32bits, 1024MB of RAM, and a Floodlight controller is used for the experimental setup.

The SDN network topology is emulated on Mininet by creating virtual switches, hosts, links and a controller. The OpenVSwitch (OVS) is used as an OpenFlow switch [16,17] and floodlight [18,19] as the controller. The emulated network has a tree topology consisting of one controller, three OpenVSwitches, four hosts and six links. Figure 5 illustrates the emulated network topology.

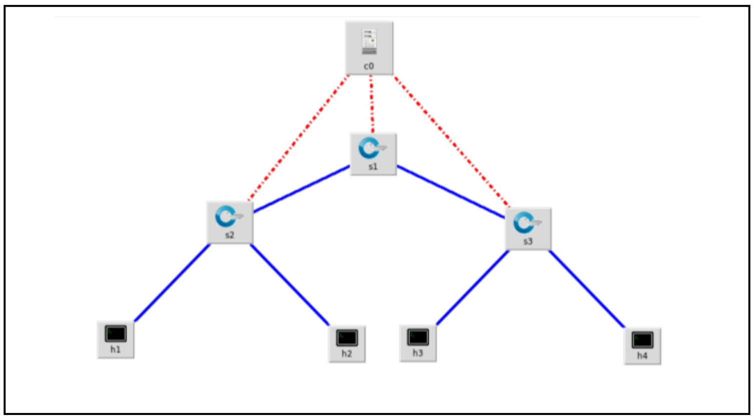


Figure 5. Emulated Network Topology

Mininet creates the network using command line interface (CLI) while the graphical user interface (GUI) is generated in MiniEdit. The CLI for creating the network is given below:

```
sudo mn --topo=tree,depth=2,fanout=2
```

The next CLI will install the sFlow agent on each of the switches and will collect samples that will be sent for analysis:

```
sudo ovs-vsctl -- --id=@sflow create sflow agent=eth0 target=\"127.0.0.1:6343\"
sampling=20 polling=30 -- -- set bridge s1 sflow=@sflow
```

With this command, at every interval of 30 seconds, one out of every 20 packets transiting the network will be collected for sample.

Multiple application layer programs can be run on the network and the performance of the network can be monitored from the dashboard through a browser. Series of tests are carried out on the network in different phases to ascertain the impact of security threats. Then, the proposed security method is implemented and analyzed. Host h1 is set as the victim while other hosts are set to compromise the network by establishing DDoS flood attacks on the network.

Phase 1: Host h1 was ping from other hosts on the network and it was proved reachable. Figure 6 shows the reachability of h1.

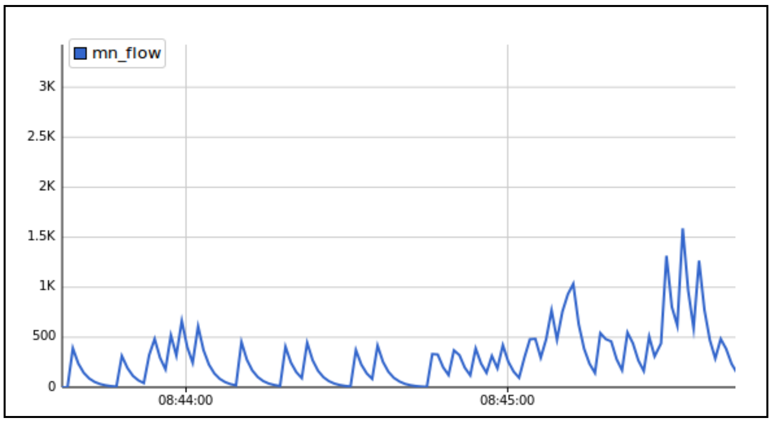
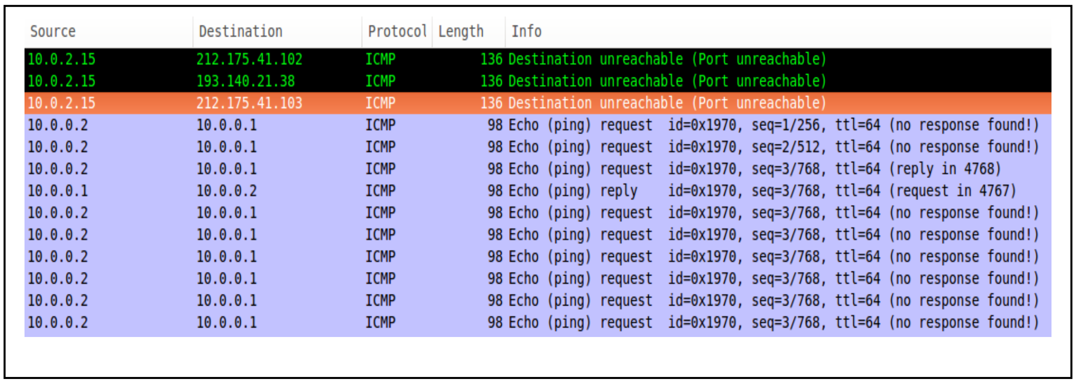


Figure 6. Testing for Reachability on Host h1

Phase 2: A TCP SYN flood was then generated using hping3 on host h3 and directed towards h1 with the aim of flooding the network resources and rendering h1 unreachable. To verify this, host h2 was used to ping host h1 but it was proved unreachable showing the presence of the SYN flood attack which is seen in Figure 7.



```
Source          Destination      Protocol Length  Info
10.0.2.15       212.175.41.102   ICMP        136 Destination unreachable (Port unreachable)
10.0.2.15       193.140.21.38    ICMP        136 Destination unreachable (Port unreachable)
10.0.2.15       212.175.41.103   ICMP        136 Destination unreachable (Port unreachable)
10.0.0.2        10.0.0.1         ICMP         98 Echo (ping) request  id=0x1970, seq=1/256, ttl=64 (no response found!)
10.0.0.2        10.0.0.1         ICMP         98 Echo (ping) request  id=0x1970, seq=2/512, ttl=64 (no response found!)
10.0.0.2        10.0.0.1         ICMP         98 Echo (ping) request  id=0x1970, seq=3/768, ttl=64 (reply in 4768)
10.0.0.1        10.0.0.2         ICMP         98 Echo (ping) reply    id=0x1970, seq=3/768, ttl=64 (request in 4767)
10.0.0.2        10.0.0.1         ICMP         98 Echo (ping) request  id=0x1970, seq=3/768, ttl=64 (no response found!)
10.0.0.2        10.0.0.1         ICMP         98 Echo (ping) request  id=0x1970, seq=3/768, ttl=64 (no response found!)
10.0.0.2        10.0.0.1         ICMP         98 Echo (ping) request  id=0x1970, seq=3/768, ttl=64 (no response found!)
10.0.0.2        10.0.0.1         ICMP         98 Echo (ping) request  id=0x1970, seq=3/768, ttl=64 (no response found!)
10.0.0.2        10.0.0.1         ICMP         98 Echo (ping) request  id=0x1970, seq=3/768, ttl=64 (no response found!)
10.0.0.2        10.0.0.1         ICMP         98 Echo (ping) request  id=0x1970, seq=3/768, ttl=64 (no response found!)
```

Figure 7. Unreachability of Host h1 after SYN Flood Attack

Phase 3: ICMP flood attacks were launched from other hosts on the network towards h1. This was done to generate a huge amount of traffic that could overwhelm the network resources. This is done by sending continuous requests to the controller for addition of missing packets in the flow tables. Shortly, the controller became so busy that it could not respond to requests from other hosts on the network and it starts dropping packets from eligible users or hosts. Figure 8 shows the ICMP flood attack on the network.

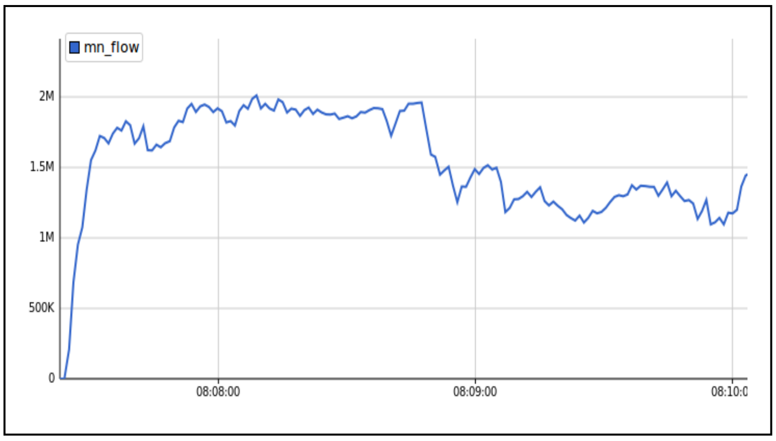


Figure 8. ICMP Flood Attack

Phase 4: Security measures are taken against the threats associated with the network by implementing the sFlow detection and mitigation technique by running the mitigation script on the application layer of the SDN network. A threshold of 700,000 pps was set on the network at a sampling rate *S* of 20 and a polling interval *I* of 30 seconds. It can be seen from Figure 9 that at the start of the attack, the packet flow soared to a peak of approximately 1,700,000 pps but since measures were taken to curb it, the packet drops below the threshold of 700,000 pps after about approximately 15 seconds of implementation.

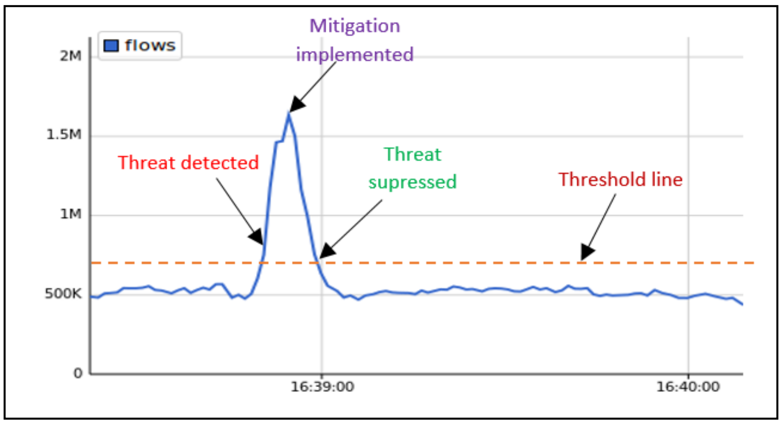


Figure 9. Detecting and Mitigating the Attack

5. CONCLUSIONS

This paper presents and demonstrates how the SDN network can be secured by implementing sFlow technology combined with IPsec protocol. The sFlow technology is a real-time analyzer that can be embedded with the controller and its packet overhead consumption is just 0.02% of a 10 Gb ethernet link yielding very flexible and a light solution. Its response detection time is about 5 secs and control time is approximately 15 secs which makes it a probable solution to mitigate the DDoS attack posed on the SDN network resources.

Comparing with the other methods discussed in Section 2 of this manuscript, our method outperforms them as it detects threats on the network in real-time and helps the controller to quickly decide on dropping packets from infected ports while ensuring that the network resources are not overwhelmed. Most of the methods in the literature have large packet overheads that consume notable fraction of the overall bandwidth of the controller. In contrast, our method only consumes 2 Mb in a 10 Gb ethernet link making it scalable and easy to deploy.

The implemented security measure is able to protect the network against threats and ensured availability of the network to authorized users. At the same time, it prevents both internal and/or external attacks and abuses such as DDoS attacks that could be generated by eligible or unauthorized users on the network. With the measure taken, confidentiality and integrity of the data, and availability of the network resources are achieved.

REFERENCES

[1] Martin Vizv´ary, Mitigation of DDoS Attacks in Software Defined Networks, Ph.D. thesis proposal, 2015.

[2] Yanbiao Li, Dafang Zhang, Javid Taheri, and Keqin Li. SDN components and OpenFlow. Taheri CH003.tex pg 53. January 8, 2018.

[3] Rowan Klöti, Vasileios Kotronis, Paul Smith, OpenFlow: A Security Analysis. Proceedings of the 21st IEEE International Conference on Network Protocols (ICNP) At: Göttingen, Germany. Oct, 2013.

[4] Ijaz Ahmad, Suneth Namal, Mika Ylianttila, Senior Member, IEEE, and Andrei Gurtov, Senior Member, IEEE, Security in Software Defined Networks: A Survey. IEEE Communication Surveys & Tutorials, vol. 17, no. 4, fourth quarter 2015.

[5] Vaughan-Nichols SJ. OpenFlow: The next generation of the network? Computer, vol. 44, no. 8, pp. 13–15, Aug. 2011.

[6] Karamjeet Kaur, Krishan Kumar, Japinder Singh, Navtej Singh Ghumman. Programmable Firewall Using Software Defined Networking. 2015 2nd International Conference on Computing for Sustainabile Global Development (INDIACom).

[7] Rajat Kandoi, Markku Antikainen. Denial-of-Service Attacks in OpenFlow SDN Networks. 2015.

[8] H. Wang, Zhang D and Shin KG. "Detecting SYN Flooding Attacks," Proceeding of Twenty-First Annual Joint Conference of the IEEE omputer and Communications Societies, pp. 1530-1539, 2002.

[9] Bing Wang, Yao Zheng, Wenjing Lou, Y. Thomas Hou. DDoS Attack Protection in The Era of Cloud Computing and Software Defined Networking. International Conference on Network Protocols (ICNP), IEEE, Raleigh, NC, 2015.

[10] Ogundile OO, Lawal BH and Osanaiye OA. A Secured Voice over Internet Protocol (VoIP) Setup Using MiniSipServer. International Journal of Scientific & Engineering Research, Volume 3, Issue 11, November-2012 ISSN 2229-5518.

[11] R. Marin-Lopez, G. Lopez-Millan, Software-Defined Networking (SDN)-based IPsec Flow Protection.

[12] Frankel S and Krishnan S. IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap, RFC 6071, DOI 10.17487/RFC6071, February 2011.

[13] Levent Ertaul, Krishnakumar Venkatachalam. Security of Software Defined Networks (SDN). Int'l Conf. Wireless Networks | ICWN'17 | [13] RFC 2406, 2017.

[14] Shie-Yuan Wang. Comparison of SDN OpenFlow Network Simulator and Emulators: EstiNet vs. Mininet.

[15] Faris Keti, Shavan Askar. Emulation of Software Defined Networks Using Mininet in Different Simulation Environments. In proceedings of the 6th International Conference on Intelligent Systems, Modelling and Simulation, 2015.

[16] Lantz B, Heller B and McKeown N. "A network in a laptop: rapidprototyping for software-defined networks," in Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks. ACM, 2010.

[17] Jad Naous, David Erickson, G. Adam Covington, Guido Appenzeller, Nick McKeown, Implementing an OpenFlow Switch on the NetFPGA platform. 2008.

[18] http://www.projectfloodlight.org/floodlight/, accessed date: 03/17/2018.

[19] Hakan Akcay, Derya Yiltas-Kaplan. Web-Based User Interface for the Floodlight SDN Controller. March 22, 2017.