FUZZY PROTECTION METHOD FOR FLOOD ATTACKS IN SOFTWARE DEFINED NETWORKING (SDN)

Mohammad Hadi Zahedi*
Department of Computer Engineering
K.N. Toosi University of Technology
Tehran
Iran
mhadi_zahedi@yahoo.com

Abbas Ali Rezaee
Department of Computer Engineering
Payame-Noor University
Tehran
Iran

Zeinab Dehghan
Department of Computer Engineering
Payame-Noor University
Tehran
Iran

Abstract. Flood attacks (FA) are a type of distributed denial of service (DDoS) attacks. In FA, an attacker sends massive floods of packets to consume all resources. Hierarchical architecture and numerous weaknesses in the structure of communication protocols in conventional networks lead to the fact that firewalls are incapable to provide an integrated and effective mechanism against these attacks. With the emergence of Software Defined Networking (SDN), there are new prospects for solving structural and security problems in conventional networks. This study investigates some ideas for protecting against distributed FA using SDN. Later, by analyzing the strengths and weaknesses of these ideas, a heterogeneous method is proposed based on a combination of conventional service provider and the Software Defined controller. In the proposed method, the attack detection and the fuzzy decision modules are located in the service provider and the controller (i.e. SDN), respectively. In order to simulate the heterogeneous method the MiniNet emulator is applied in combination with the Pox controller. Afterwards, the simulated model is evaluated. Results show that besides protecting against attacks in conventional networks, the proposed method provides other benefits including the extent of computational load and the response time in comparision to other Software Defined methods.

Keywords: fuzzy system, software defined networks, DDoS attacks, flood attacks, OpenFlow.

∗. Corresponding author

1. Introduction

One of the most important measures of information security is the accessibility of resources, which means that we have to make sure of the correct functioning of information processing devices and the communication channels. In other words, authorized people should have access to resources whenever needed. Denials of service (DoS) attacks are amongst the threats which seriously challenge access to information. Using these attacks, attackers try to hog a major portion of the accessible resources in order to interrupt service [1-2]. FA is amongst the most dangerous denial of service attacks, which bombards the network due to weaknesses in communication protocols. For instance, SynFlood attacks which occur in transfer plane account for more than 90 percent of DoS attacks [3]. The hierarchical structure, weakness in communication protocols, and the lack of a centralized command center in current networks causes serious challenges to protect against these attacks [4, 5].

Transfer control protocol (stocktickerTCP) weakneses lead to challenges listed above. One is the weakness of three-stage handshaking operation. This problem is due to the lack of authentication for the communication initiator by the service provider. Under normal conditions, after receiving the SYN packet, the service provider allocates some memory to follow the communication session and waits for the communication to start. Utilizing this weakness, attackers start sending floods of invalid packets with the sign SYN=1 and fill up the service provider's memory with invalid requests. When the memory is clogged, the service provider has to abandon new requests [6] and so it cannot provide anymore service.

User datagram protocol (UDP) also has its own probems. This protocol starts a communication without connection, which means that the connection is a one-way communication in one direction from source to destination without considering the status of the receiver [7]. In UDP attacks, packets are sent from random ports to the victim machine to saturate the bandwidth of the target machine and prevent service provisioning for the users. In this state Firewalls are not able to alleviate these attacks [8].

With the emergence of Software Defined networks (SDN's), new prospects for solving the structural problems of conventional networks have appeared [9]. In this paper, a novel method is proposed based on the cooperation between the service provider and the SDN controller for protecting against flood attacks in conventional networks.

The rest of this paper is organized as follows: Section 2 provides a background to better understand the concepts of the study. Section 3 investigates the related works and research. Section 4 introduces the proposed heterogeneous defense method and Section 5 evaluates the performance of the proposed method. Section 6 discusses the conclusion.

2. Software Defined Networks (SDN): state of the art

Defense against FA attacks has been studied for several decades. Some studies [10, 11] have investigated a number of methods for protecting against these attacks. They use firewalls or attack filtering devices. Based on previous studies, firewalls and attack filtering devices do not provide effective approaches to protect against attacks where weaknesses in protocols are the basis for attack [12]. Because of management complexities and weakness in communication protocols, the architecture of conventional networks cannot provide effective and integrated approaches for protecting against FA.

In the architecture of SDN, data and the control section have been separated. The architecture of the horizontal network is more intelligent and controllable. One of the big ideas in SDN's is the fact that a device, called a controller, is directly linked to all the devices in a domain; it is aware of the network architecture and it plans the network from a central point. An SDN controller changes the planning model of the network from distributed mode to the centralized mode [13]. The centralized planning of the network is a valuable characteristic which can be used for different types of security policy. Using SDN's, there is no need for technical managers to control and manage all the devices on the network through terminals. The controller acts as a middleware which separate components of the network's physical layer, such as routers, switches, firewall, and load balancer from the software layer. In fact, the controller monitors the network from a central point using both information obtained and flexible instruction. In Fig. 1, the structure of SDN's is illustrated. It consists of three major components including the controller, virtual switches, and overlapping networks. Moreover, SDN's are based on the OpenFlow protocol [14].

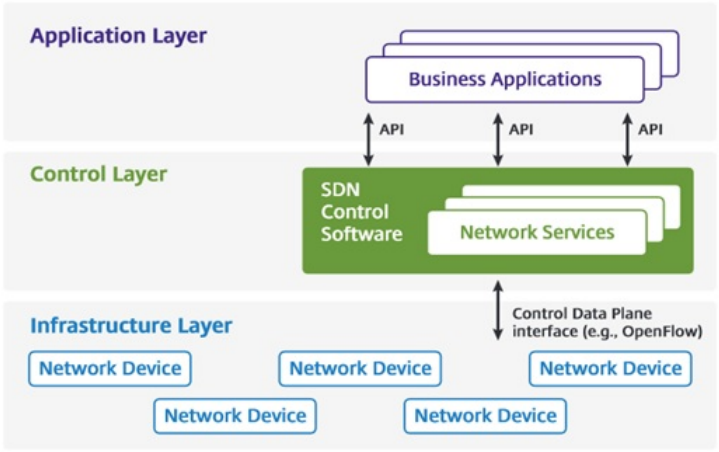

Figure 1: Structure of a SDN [14]

2.1 OpenFlow protocol

OpenFlow protocol is the first standard communication interface which is defined between the control and the data transfer [3]. Using this protocol, high-level programming languages can be used for creating low-level instructions and transferring them to routers and switches. As shown in Fig. 2, this protocol will first define the central controller. Then, with a secure connection to the controller, it enables the control of the network. In SDN's, the central controller maintains all the network rules and issues instruction through the OpenFlow protocol as needed [15].

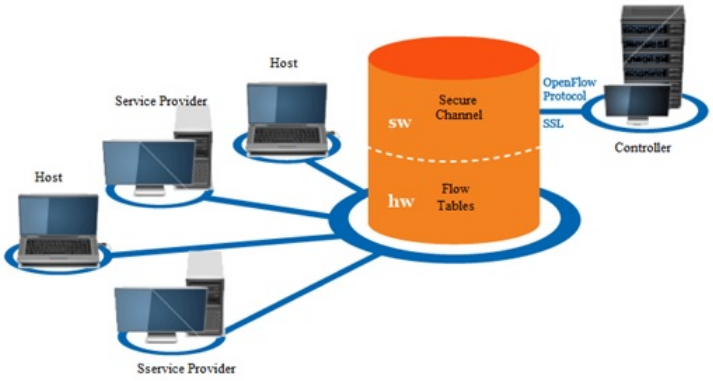

Figure 2: Schematic Representation of OpenFlow Protocol [16]

One of the issues in OpenFlow protocol is the flow-based nature of the exchange policies. A flow table is a list of all the flow entries which includes matching fields, counter, and instruction. After being received, the arriving packets are compared to the matching fields and if there is a match, the packet will enter the exchange process. Otherwise, it is referred to the controller, which decides what will happen. Nowadays, in large-scale datacenters there is a huge interest in OpenFlow. Huge companies design switches based on this protocol and many of them support this protocol. Google has used this protocol in its datacenters [17].

3. Related works

In the literature, there are various methods that protect against distributed denial of service (DDoS) attacks [18-19]. There are some studies regarding the methods to protect against DDoS attacks using SDN [20-21]. In [20] the random hopping plan of the service provider using OpenFlow in SDN's has been proposed. In this plan, using static IPs for service providers in the network has been proposed as a challenge and it is argued that using active static IPs in a domain or network, will lead to multiple scanning and denial of service attacks.

In order to resolve these issues, the authors proposed a method where real IPs are replaced with virtual IPs and the domain name is retrievable through the system while the real IPs are only visible to authorized individuals. This method prevents the identification of active sources on the network. In [21], a strategy is presented for security in the controller core in order to detect the security issues of flows in switches. This strategy is called FortNox and it has been developed based on Nox controller [5]. In this method, the security core is added to the controller, which controls the passing flows and monitors the conflicts and problems in flows. In this method, digital signature is used for authenticating programs. It has a 7-second overhead for checking the flows against the Nox controller.

In [22], a method is introduced for validating the source address based on Nox controller. In this method, each packet located outside the network that intends to enter the network is directed towards this controller. The process of validating the source of the packet is carried out by the controller. If the source is reliable, it permits the traffic to move along. Otherwise, the packet is stopped. This method can prevent IP spoofing operations in FA. In [23], a method based on Nox controller is suggested for detecting DDoS attacks. In this method, an algorithm is proposed for detecting attacks. It uses unsupervised neural networks and the characteristics of arriving traffic to detect DoS attacks. In this work, there is no clear discussion on attacks responds method.

In [12], the method of protecting webservers from SynFlood attacks based on Pox controller is presented. In this method, a controller called OPERETTA is introduced which validates each Syn packet and after validation, it permits the connection. In this technique, two scenarios for responding to FA has been discussed which include a central and a local scenario. In [24], a method for protecting against distributed traffic attacks in cloud environments based on the Floodlight controller is introduced. This controller consists of two modules of detection and defense. The detection module identifies attacks in the cloud and informs the defense module. Therfore the necessary measures are taken. This method is the first method proposed for protecting against attacks in cloud environment using SDN's.

In studies [12 , 23,24], detection and defense module is located centrally in the controller. One of the main problems with these methods is the increased load exerted on the controller during the FA. When the attacks start, a huge number of packets with spoofed sources flood the service provider. The controller must analyze all the packets and take necessary measures. In such cases, an additional computational load is exerted on the controller and it is possible that due to this extra load, the controller fails to manage and control the network requests [25]. The second problem with the above-mentioned methods occurs during the transfer of normal flows of the network to the controller. Authenticating the packets by the controller can create a delay in responding to authorized flows. Since the majority theories of SDN's are flow-based, the

additional delay in setting up an authorized flow is one of the main concerns .In Table 1; various methods proposed by collegues are summarized.

Method	Proposed Year	Controller	Establishment
SOM	2010	NOX	Centralized controller
FortNox	2012	NOX	Centralized controller
DaMask	2015	Floodlight	Centralized controller
OPERETTA	2015	POX	Centralized controller

Table 1. Published methods

4. Proposed method

The proposed method is a heterogeneous defense method based on the cooperation of conventional network service provider and the Software Defined controller to protect against FA. In the presented method, the detection and defense components are separated in order to decrease the computational load and prevent delay in authorized flows sent to the controller. The detection and defense modules with numerous analyses are seperated. During these analyses, it is noticed that attack detection in the targeted service provider which are more accurate and presents less false-positives [26]. In Fig. 3, the structure of the proposed heterogeneous method is demonstrated.

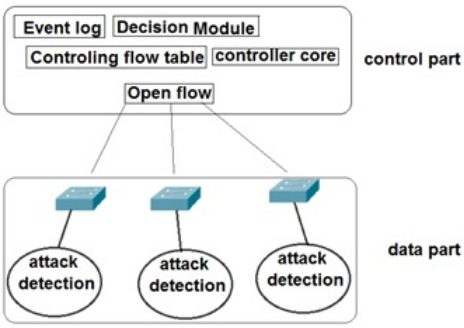

Figure 3: The structure of the proposed heterogeneous method

In heterogeneous method, the attack detection module is located in service provider and the decision module is located in the controller. It applied fuzzy methods to make decisions [27].

In the introduced method, Open vSwitch is used in order to integrate and connect the conventional service provider to the Software Defined controller. Open vSwitch is an open source software switch, licensed under Apache License 2.0. Besides supporting SFlow and NetFlow protocols, it also supports every version of OpenFlow protocol (1.0 to 1.5) [28]. In Fig. 4, the proposed algorithm

is shown in three phases. The closed-movement phase in the proposed method includes the following phases: 1- Connection request, 2- Event analysis and report, 3. Decision and response.

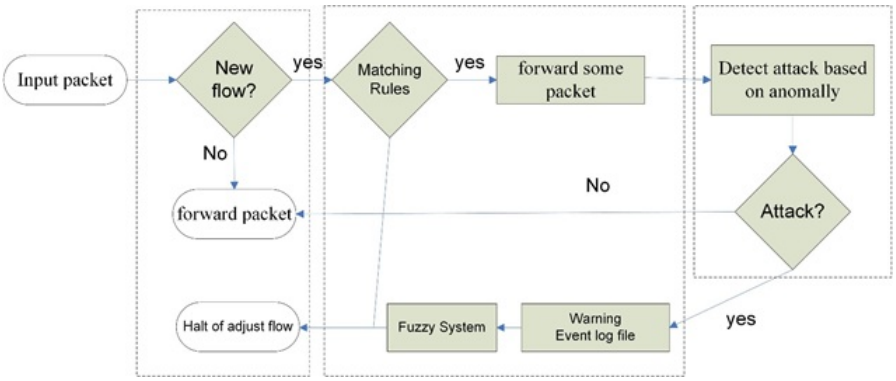

Figure 4: The architecture of proposed system

4.1 Connection request phase

In this phase, the packet enters the OpenFlow switch. Then, the switch checks the flow tables. If the arriving packet is present in the flow tables, the permission for the flow to move towards the destination is issued. Otherwise (if arriving packet belongs to a new flow), it is sent to the controller so that the necessary actions can be determined Fig. 5.

```
OpenFlow Protocol
▽ Header
    Version: 0x01
    Type: Packet In (AM) (10)  //Packet Arrival
    Length: 72
    Transaction ID: 0
▽ Packet In
    Buffer ID: 4294967295
    Frame Total Length: 54
    Frame Recv Port: 4
    Reason Sent: No matching flow (0)  //The Received Packets is a New and it is Not Switches Flow Tables
  ▽ Frame Data: 000000000004000000000006080045000284b2d40001906...
    ▷ Ethernet II, Src: 00:00:00_00:00:06 (00:00:00:00:00:06), Dst: 00:00:00_00:00:04 (00:00:00:00:00:04)
    ▷ Internet Protocol Version 4, Src: 10.0.0.4 (10.0.0.4), Dst: 10.0.0.1 (10.0.0.1)
    ▽ Transmission Control Protocol, Src Port: ftp-data (20), Dst Port: 30 (30), Seq: 13456, Len: 0
        Source port: ftp-data (20)
        Destination port: 30 (30)
        [Stream index: 2]
        Sequence number: 13456    (relative sequence number)
        Header length: 20 bytes
      ▽ Flags: 0x002 (SYN)
```

Figure 5: An Example of a Packet Sent to the Controller

Once the packet reaches the controller, preliminary investigations such as identifying the destination address and type of protocol are carried out. In case the packet is determined to be based on the current rules of the network, the

controller creates a temporary flow table and permits some packets to continue towards the service provider Fig. 6.

```
                // Creating Event For Listening To Arriving Packet
def launch ():
    core.openflow.addListenerByName("PacketIn", _handle_PacketIn)

                // Parsing Packet
def _handle_PacketIn (event):
    packet = event.parsed
    dst_port = table.get(packet.dst)

                // The Expiration Time For Temporary Flow Tables
    msg = of.ofp_get_n_packet_flow_mod()
    msg.match.dl_src = packet.src
    msg.match.dl_dst = packet.dst
    msg.actions.append(of.ofp_action_output(port = dst_port))
    event.connection.send(msg)
```

Figure 6: The Preliminary Analysis of the Packet and the Permission for Multiple Packets in the Flow

4.2 Event analysis and report phase

In this phase, packets are forwarded towards the service provider based on the temporary flow tables. Once the packets enter, the attack detection module starts there analysis. Different methods have been proposed for detecting DDoS attacks. Pprevious studies have shown that anomaly-based methods provide better performance compared to signature-based methods in order to detect attacks [24]. On the other hand, in anomaly based detecting methods, the presence of a few packets is enough for detecting attacks, while in signature-based methods all the packets related to that flow are required [24]. Therefore, considering the benefits of the anomaly-based method and the current limitations (the low number of packets arriving at the detection module) this detection technique is a suitable approach to be used in the heterogeneous method. The parameters in [29], such as packet type entropy, packet rate, and the number of packets for attack detection are applied.

4.3 Decision phase

After receiving the event report by the controller, the decision module comes online and updates the event report file to send the necessary decision to the defense component. The decision determines the type of response to the attack and can be carried out by the fuzzy controller. Under this condition the parameters of trust in the service provider and the sensitivity of the provided service

are considred. This phase makes a decision firstly and then responses to the attack subsequently.

4.3.1 Decision making step

Considering the fact that the parameters of trust and sensitivity can be represented in a fuzzy manner, these two parameters are used for fuzzy decision making and reporting the extent of the damage.

- **The parameter of trust in the service provider**

As shown in Fig. 4, after detecting the attack by the service provider, the event report is sent to the controller where it will be placed in the attack event file. By reading this file, the number of attacks on a service provider is calculated. Whenever an attack occurs on the service provider, it will be added to the event log file and the controller's trust score for the service provider is lowered. The trust score can be expressed using lingual variables including high, medium, and low trust.

- **The parameter of the service sensitivity**

The parameter for the service sensitivity can vary from a network to another. This parameter should be given to the controller by the network administrator. For instance, an attack on the web service provider, email service provider, or special ports such as 53, 80, and 443 will be more sensitive than non-reserved ports. The service sensitivity score can also be expressed using lingual variables including high, medium, and low sensitivity.

In the decision making step, using the different values of trust and sensitivity, the damage can be calculated and reported to the defense component. The inflicted damage can invoke different responses to the attack. For example, if the damage estimation is high, in response, the controller must choose a method which not only alleviates the attack but also does not lead to false positives in the network. In order to simulate the proposed fuzzy model, the fuzzy environment of the MATLAB emulator, triangular membership function, and center of gravity defuzzification are used. Using trust inputs, sensitivity, and "if... then" rules entered into the fuzzy system, the fuzzy inference system (FIS) is constructed. The set of utilized rules can be seen in Fig. 7.

The output of the fuzzy inference system can be turned into FCL language by MATLAB and using frameworks such as PyFuzzy, it can be implemented into Python and integrated into the controller. Fig. 8 shows the fuzzy decision behavior model.

4.3.2 Response to attack step

After detection of the damage, decisions will be sent to the response to attack component. A simple method is to prevent the construction of flow tables in the

Inputs		Confidence		
		low	middle	high
Sensitivity	low	middle	high	high
	middle	low	low	high
	high	low	low	middle

Figure 7: Fuzzy rules

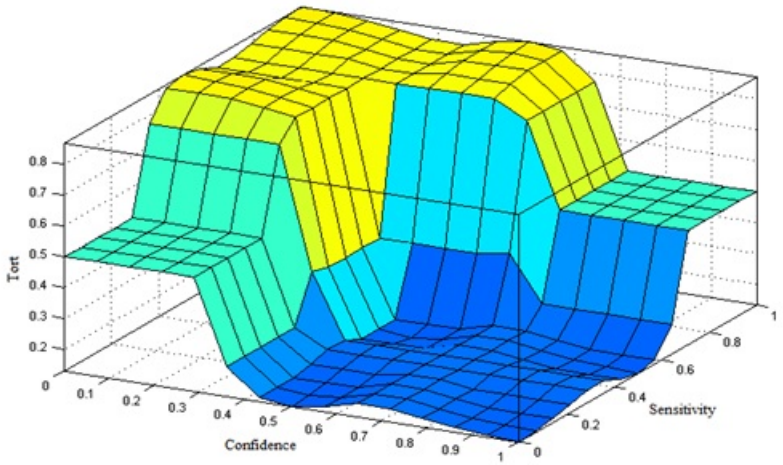

Figure 8: Fuzzy decision behavior model

attack source. In other words, after receiving the report, the Packet-In event related to the attack packets must be halted and the formation of flow tables for similar packets is prevented. Fig. 9 shows the code fragment related to halting the attack event. After the event report in this code fragment, controller's instruction is sent to the switch connected to the attacker and the flow passing through is halted.

5. Performance evaluation of the proposed algorithm

The proposed method is analyzed and evaluated in the MiniNet simulator which is one of the most popular Software Defined simulators by applying POX controller [11, 20, 27] on a Linux OS. MiniNet uses architecture of several virtual hosts that can construct virtual links, which can be connected to OpenvSwitch. In the MiniNet environment, every virtual host has a separate OS kernel and each is able to run its programs and commands independently.

```
def block_handler (event):

        packet = event.parsed   // Detecting The Connection Rule
        tcpp = event.parsed.find('tcp')
        udpp = event.parsed.find('udp')

        if tcpp:
        if tcpp.srcport in block_ports or tcpp.dstport in block_ports:
            core.getLogger ("SYN Receive")
            core.getLogger("DDos Blocker Start...").info("Blocked TCP-Attack Sourece
            event.halt = True     // Halting The Event Related To TCP
        elif udpp:
                if udpp.srcport in block_ports or udpp.dstport in block_ports:
            core.getLogger("DDos Blocker Start...").info("Blocked UDP-Attack Sourece
            event.halt = True     // Halting The Event Related To UDP
```

Figure 9: Halting the Attack Flow

In order to set up the Mininet emulator, Ubuntu 12.10 32bit OS with Quad Processor Q6600 CPU using 4 GB of memory is used. For running the Pox controller a laptop computer with an Intel 2Core i684-1.4 GHz CPU is used. In order to simulate the FA, Hping3 application is used. This tool is one of the most powerful programs for sending customized packets towards a determined destination. This tool is generally used for penetration tests. Hping3 provides the possibility to send a flood of packets by spoofing the source's IP towards the destination. In order to analyze and measure the rate of in-transit packets, WireShark tool is used. For measuring the processes which consumed resources the HTop tool [30] is used. Also for creating legitimate requests for receiving information iPerf [31] is utilized.

The network architecture consists of a number of Linux systems, which are connected to seven OpenFlow switches. In this architecture, a web server and the users are working. Also, there are a number of infected systems which try to attack and interrupt the normal flow of services in the network as shown in Fig. 10.

Since the heterogeneous defense method is a technique based on the cooperation of a service provider in conventional networks and a Software Defined controller, it should be able to not only pass the attacks in conventional networks but also provide a number of benefits over the Software Defined centralized defense method. Therefore, in order to evaluate the efficiency of the heterogeneous defense method, two comparisons are made. In the first comparison in Section 5.1, the superiority of the heterogeneous defense method over traditional methods is evaluated. Then, in Section 5.2, the heterogeneous defense method with defense methods concentrated in the controller is compared.

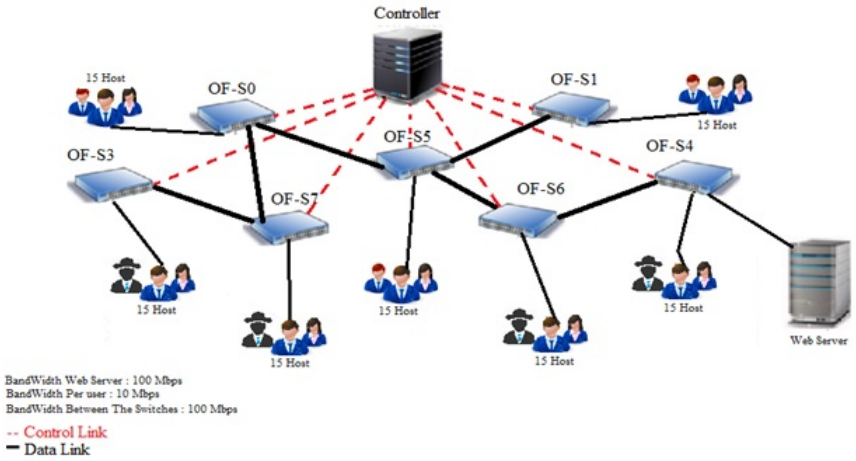

Figure 10: Network Architecture and Controller for Evaluating the Heterogeneous Defense Method

5.1 Comparing the proposed method with traditional methods

In this section, comparison of the heterogeneous defense method with traditional methods of protecting against attacks (SynCookies and the firewall present in the service provider) is done. In order to do the evaluation, criteria such as response time for authorized users and the processing load exerted on the service provider is used.

- **Service provider's response time**

In order to measure the response time of the service provider for authorized users in the traditional method and the heterogeneous method an experiment with a number of different estimated rates of SynFlood packets on the web service provider is carried out. In the first step after the attack, traditional methods are used (SynCookies and the firewall present in the service provider) to pass the attacks and results are recorded. Then, by activating the heterogeneous defense system defens against the attacks are tried and the results were also recorded.

As obtained from Fig. 11, in traditional method, by increasing the rate of attacks, a significant amount of time is spent for responding to unauthorized requests and the response time for authorized users is increased. In the heterogeneous defense method, after detecting the attack, the controller is informed of the situation. Then, the controller identifies the source and destination of the attacks and sends decisions to the switch connected to the attacker to cut off or adjust the flow. Afterwards, the attacks are halted and the response

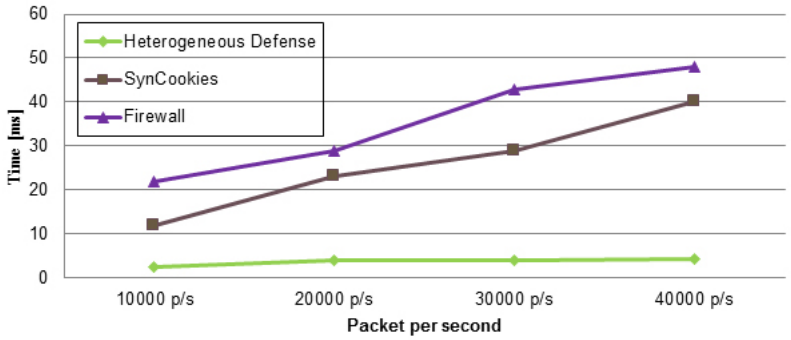

Figure 11: Average Response Time of the Service Provider for Authorized Requests over Packet/second

time is revived. In the proposed method, the controller does not allow attack flow towards connection links and the resources of the service provider from the attacker switch.

- **The usage of central processor**

In Fig. 12, the usage level of the central processor during the protection against attacks using the traditional and the heterogeneous methods is provided. As can be seen, in traditional defense methods, a high amount of processing load is exerted on the central processor of the service provider during the defense against the attacks. By increasing the rate of the attacks, the usage of the processor is also increased. Meanwhile in the heterogeneous method, due to the separation of the defense system, the only load exerted on the service provider due to calculations of the attack detection module, which can be seen in Fig. 12.

The results of experiments show that traditional method in comparision with the heterogeneous method provides protection of resources, access for the authorized users and prevents interference with network resources.

5.2 Comparing the proposed method with methods concentrated in the controller in thwarting flood attacks

For evaluation, criteria such as the computational load on the controller and the delay in responding to authorized flows are used. As previously mentioned in Section 3 , [12] , [23], and [24] provide methods for detecting and thwarting denial of service attacks based on Software Defined networks. In all proposed schemes, the detection and defense modules are concentrated on the controller. However, in the heterogeneous defense method, the detection module is in the service provider and the decision module is in the controller. The strategy proposed in

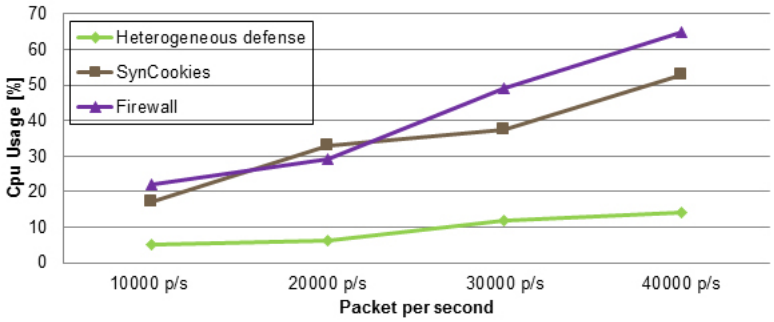

Figure 12: Average Processing Load on the Service Provider in Traditional and Heterogeneous Defense Methods

[24] is designed for cloud environments and [23] only discusses attack detection methods. Therefore [12] , provides a method for thwarting attacks in traditional networks.

In this scheme, the OPERETTA controller is proposed. The OPERETTA is an evolved version of the Pox controller for thwarting SynFlood attacks in conventional networks. In this scheme, each TCP packet which intends to reach the service provider is validated by the controller. If it passes the validation process, it is allowed to go to the service provider. Among the important problems in the OPERETTA controller, the high delay in delivering validated packets to the service provider can be mentioned. Here, the proposed heterogeneous defense method is compared with the centralized method of detection and defense based on Pox and OPERETTA controller. Table.2 shows the specifications of the testing environment for each controller.

Detection and Defense Method	Detection and Defense Centralized in Pox Controller	Detection and Defense Centralized in OPERETTA controller	Proposed Deterogeneous Dethod
Detectionand defense system	Integrated	Integrated	Based on cooperation
Simulator	MiniNet	MiniNet	MiniNet
Controller core type	Pox	OPERETTA	Pox
Controller hardware	Intel i3-2365m-1.4Ghz-Ram 4GB	Intel i3-2365m-1.4Ghz-Ram 4GB	Intel 2core i686 1.4 Ghz Ram 4GB
Experimental OS	Kubuntu 13.10 64bit	Kubuntu 13.10 64bit	Ubuntu 12.10 32bit
Ability to thwart attacks	SynFlood	SynFlood	SynFlood-UdpFlood

Figure 13: Specifications of the Testing Environment

Evaluating the processing load in controller

In order to evaluate the processing load on the controller in the heterogeneous method, a flood attack consisting of Syn packets with a number of different rates is carried out on the web service provider as shown in Fig. 10. Then, using the Htop tool, the results for the usage of the central processor on the controller process were measured. The obtained results were compared to OPERETTA centralized controller and the standard centralized method in Pox, extracted from [12]. The results of this comparison can be seen in Fig. 13.

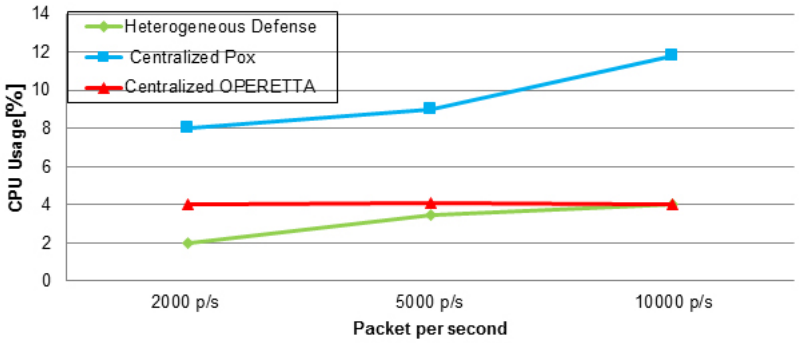

Figure 14: Average Usage of Central Processor over Packet per second

As the results show, the heterogeneous defense scheme provides a lower processing rate in the range of 2000 and 5000 packets. Compared to other methods in the range of 10000 packets, it is almost equal to the OPERETTA controller. However, as mentioned in Table. 2, the processor of the heterogeneous controller has a lower specification compared to the processor of OPERETTA controller and Pox.

6. Conclusions

Denial of service flood attacks is amongst the most common and powerful attacks which abuse the computational resources and the bandwidth of the network. The simplicity of creating denial of service tools for carrying out flood attacks and the problems of thwarting them due to the weaknesses in the structure of communication protocols have turned these attacks into the most common and most significant attacks ever. In this study, traditional networks, SDN's and the strategies of these networks for thwarting flood attacks are discussed. Moreover, their strengths and weaknesses of all methods are discussed. Then, a heterogeneous approach based on the cooperation of the service provider is presented. Also Software Defined controller, which is able to thwart flood attacks in the network, is introduced. Results show that the proposed method has a lower computational load and response time compared to other methods centralized in the controller. There will never be an ultimate defense method which can

protect all the devices against these attacks. For the future work, concentration should be on mechanism based on voting to detect attacks in a heterogeneous manner. Each service provider will have a vote and based on the votes, the controller will implement security measures in the network.

References

[1] Larson, Dave, *Distributed denial of service attacks–holding back the flood*, Network Security, 3 (2016), 5-7.

[2] *Alert (TA14-017A) UDP-based Amplification Attacks*, US-CERT, 8.

[3] Moore, D., G. Voelker and S. Savage, *Inferring Internet Denial-of-Service Activity In 10th USENIX Security Symposium*, W ashington DC, 200 (2001).

[4] Goransson, Paul and Chuck Black, *Software Defined Networks: A Comprehensive Approach*, Elsevier, 2014.

[5] McCauley, Murphy, *About pox*, URL: http://www. noxrepo. org/pox/about-pox/. Online (2013).

[6] C. Meadows, *A formal framework and evaluation method for network denial of service,* in Computer Security Foundations Workshop, 1999, Proceedings of the 12th IEEE, 1999, 4-13.

[7] U. D. Protocol, *RFC 768 J. Postel ISI 28 August 1980,* Isi, 1980.

[8] UDP flood attack, https://en.wikipedia.org/wiki/UDP_flood_attack.

[9] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, et al., *OpenFlow: enabling innovation in campus networks,* ACM SIGCOMM Computer Communication Review, 38 (2008), 69-74.

[10] T. Peng, C. Leckie and K. Ramamohanarao, *Survey of network-based defense mechanisms countering the DoS and DDoS problems,* ACM Computing Surveys (CSUR), 39 (2007), 2007.

[11] S. T. Zargar, J. Joshi and D. Tipper, *A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks," Communications Surveys & Tutorials*, IEEE, 15 (2013), 2046-2069.

[12] S. Fichera, L. Galluccio, S. C. Grancagnolo, G. Morabito and S. Palazzo, *Operetta: An OPEnflow-based REmedy to mitigate TCP SYNFLOOD Attacks against web servers*, Computer Networks, 92 (2015), 89-100.

[13] E. Borcoci, *Software Defined Networking and Architectures,* in Fifth International Conference on Advances in Future Internet (AFIN 2013), 2013.

[14] A. Hakiri, A. Gokhale, P. Berthou, D. C. Schmidt and T. Gayraud, *Software Defined networking: challenges and research opportunities for future internet,* Computer Networks, 75 (2014), 453-471.

[15] T. N. Subedi, K. K. Nguyen and M. Cheriet, *OpenFlow-based in-network Layer-2 adaptive multipath aggregation in data centers,* Computer Communications, 61 (2015), 58-69.

[16] A. Tavakoli, *Exploring a centralized/distributed hybrid routing protocol for low power wireless networks and large-scale datacenters,* University of California, Berkeley, 2009.

[17] S. Sondur, *Software Defined Networking for Beginners.*

[18] Gulisano, Vincenzo, Mar Callau-Zori, Zhang Fu, Ricardo Jiménez-Peris, Marina Papatriantafilou and Marta Patiño-Martínez, *STONE: A streaming DDoS defense framework*, Expert Systems with Applications 42, no. 24 (2015): 9620-9633.

[19] Yan, Qiao and F. Richard Yu, *Distributed denial of service attacks in software-defined networking with cloud computing*, IEEE Communications Magazine 53, 4 (2015), 52-59.

[20] J. H. Jafarian, E. Al-Shaer and Q. Duan, *Openflow random host mutation: transparent moving target defense using Software Defined networking,* in Proceedings of the first workshop on Hot topics in Software Defined networks, 2012, 127-132.

[21] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson and G. Gu, *A security enforcement kernel for OpenFlow networks,* in Proceedings of the first workshop on Hot topics in Software Defined networks, 2012, 121-126.

[22] G. Yao, J. Bi and P. Xiao, *Source address validation solution with OpenFlow/NOX architecture,* in Network Protocols (ICNP), 2011 19th IEEE International Conference on, 2011, 7-12.

[23] R. Braga, E. Mota and A. Passito, *Lightweight DDoS flooding attack detection using NOX/OpenFlow,* in Local Computer Networks (LCN), 2010 IEEE 35th Conference on, 2010, 408-415.

[24] B. Wang, Y. Zheng, W. Lou and Y. T. Hou, *DDoS attack protection in the era of cloud computing and Software Defined Networking,* Computer Networks, 81 (2015), 308-319.

[25] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado and R. Sherwood, *On controller performance in Software Defined networks,* in USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE), 2012.

[26] U. Tariq, Y. Malik, B. Abdulrazak and M. Hong, *Collaborative peer to peer defense mechanism for ddos attacks,* Procedia Computer Science, 5 (2011), 157-164.

[27] L. Feinstein, D. Schnackenberg, R. Balupari and D. Kindred, *Statistical approaches to DDoS attack detection and response,* in DARPA Information Survivability Conference and Exposition, 2003, Proceedings, 2003, 303-314.

[28] Production Quality, Multilayer Open Virtual Switch http://openvswitch.org.

[29] K. Lee, J. Kim, K. H. Kwon, Y. Han and S. Kim, *DDoS attack detection method using cluster analysis,* Expert Systems with Applications, 34 (2008), 1659-1665.

[30] *Htop Documentation http://hisham.hm/htop/*

[31] *Iperf Documentation, http://iperf.fr/*

[32] S. H. Yeganeh, A. Tootoonchian and Y. Ganjali, *On scalability of Software Defined networking,* Communications Magazine, IEEE, 51 (2013), 136-141.