High Volume DDOS Attack Mitigation on an SDN Software Defined Network

Presenting by:
Luis Eduardo Escobar Garcés
Advised by Dr. Jesús Arturo Pérez Díaz and MC Noe Marcelo Yungaicela Naula
February - May 2021

Abstract - In a world where there are more and more interconnected devices, the increases of DDOS-type attacks are a concern as it is one of the most common types of attacks. SDN is network architecture that decouples the data plane from the control plane in order to have a better control of the network. This work aims to use the results from a provided IDS to detect the potential malicious traffic and use this information to implement a mitigation technique for TCP or UDP flood-type DDOS attack in a SDN to deal with 3 main issues the overload of the controller, bandwidth overload and TCAM. As an IDS is provided it is not the scope of this work to analyze its performance.

I INTRODUCTION

DDOS attacks are one of the most frequent cyber-attacks today [1]. The architecture of the SDN allows to have a better control over the network, but being centralized, it becomes vulnerable to this type of attack, this causes that methods are sought to solve this problem. To begin, some concepts will be defined.

DDOS

CISCO defines DDOS attacks (Distributed Denial Of Service) as the bombardment of simultaneous data requests to a central server in which the attacker generates these requests from multiple compromised systems with the aim of depleting the bandwidth of target's internet and RAM to crash the target's system or disrupt their business [2].

SDN ARCHITECTURE

Software Defined Network (SDN) is a network architecture in which the control plane and the data plane are separated, this allows to have network visibility, centralized control, scheduling capability, software-based traffic analysis, easily implement security functions within the network, such as firewall, intrusion detection system (IDS), intrusion prevention system (IPS) among others [3].

The infrastructure layer (data plane) comprises network elements as switches, which expose their capabilities toward the control layer (controller plane) via interfaces southbound from the controller. The SDN applications exist in the application layer (plane) and communicate their network requirements toward the controller plane via northbound interfaces. In the middle the control layer, the SDN controller translates the application's requirements and exerts low-level control over the network elements, while providing important information about relevant information up to the SDN applications [20].

Northbound interfaces: These are the link between the applications and the SDN controller, the applications can tell the network what they need (data, storage, bandwidth, and so on).

Southbound interfaces: These facilitate control over the network and enable the SDN controller to dynamically make changes according to the real-time demands and needs.

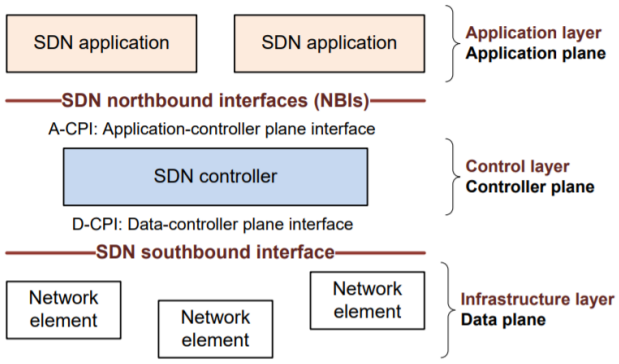


Fig 1. SDN architecture [20]

FLOOD ATTACK

A flood attack is a type of DDOS attack that seeks to fill with malicious traffic through botnets (networks of zombie computers that are remotely controlled by cybercriminals) to block the server service. Flood attacks can be carried out at layer 3 (network), 4 (transport) and 7 (application) of the OSI model [4].

TCP-SYN Flood attack [14]: As fig 2, SYN flood attack take advantage of the link connection protocol TCP, it consists in sending repeatedly high number of initial connection packet requests (SYN), the attacker can overload all available ports in the victim server, this could lead that the server responds the legitim traffic very slow

or even not responding at all. The process of the attack is the next one:

1 The attacker sends a high volume of SYN packets to the victim server, generally through a false IP.

2 The victim server responds to each one of the connection requests from the attacker and leave a port open ready to receive a response.

3 While the server waits for the final ACK packet, that never it will come, the attacker keeps sending more SYN packets. The arrive of each new packet do that the server keep a temporally open a new connection of port during a certain amount of time, once all the available ports have been used, the server cannot work with normality.

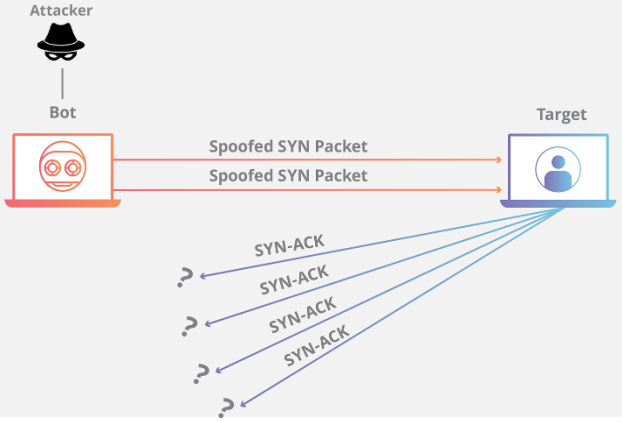


Fig. 2. TCP-SYN flood attack [14]

UDP Flood attack [15]: As fig 3, UDP flood attack take advantage of the link connection protocol UDP. Packets are sent to the victim server with the aim of overwhelming the ability to process and respond. During this attack, the attacker generally not use their own real IP address, instead they spoof the source IP address of the UDP packets, impeding the attacker's true location from being exposed and potentially saturated with the response packets from the targeted server. The process of the attack is the next one:

1 The server receives a UDP packet at a particular port.

2 The server checks to see if any programs are running which are presently listening for requests at the specific port.

3 If no programs are receiving packets at that port, the server responds with ICMP (ping) packet to inform the server that the destination was unreachable.

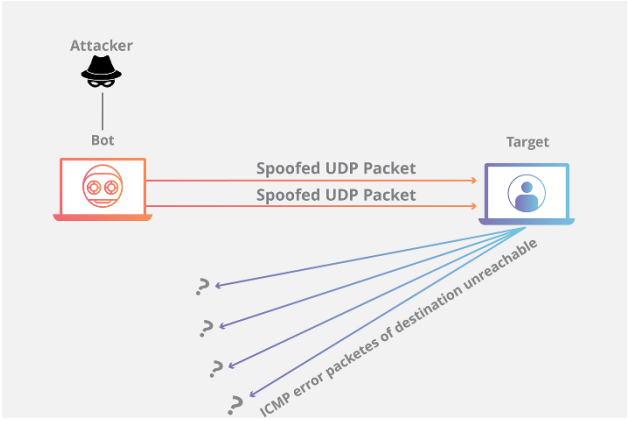


Fig. 3. UDP flood attack. [15]

DNS flood attack [16,17]: As fig 4 illustrates, DNS is the Domain Name Protocol that translates domain names into IP address to connect to identify the destiny server. DNS flood attack is a symmetrical DDOS attack. These attacks attempt to exhaust server-side assets (memory or CPU) with a flood of UDP requests.

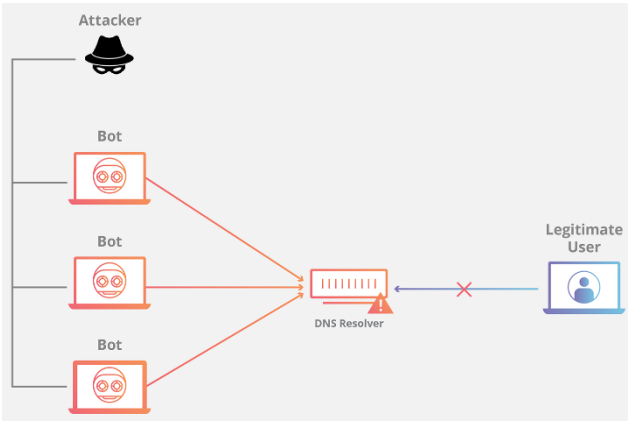


Fig. 4. DNS flood attack [16]

Packet-in message process:

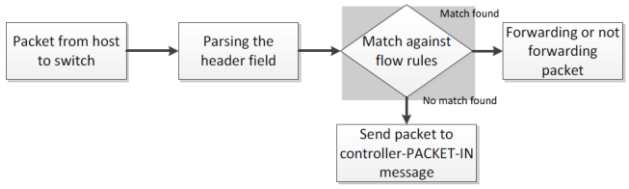


Fig. 5. Packet-in message [13]

As in Fig. 5 [13], shows the process of the packet-in message, first a packet is sent from the host to the switch and then the header field is analyzed, then it checks if it's a match against flow rules if there is no match the packet is sent to the controller to verify which type of flow rule need and in the other scenario if there is a match found against

flow rules, the packet is forwarded or not depending on the flow rule.

When the switch receives the packet through its input port, it checks its flow table for any entries that matches the fields in the incoming packet like IP address, MAC address, port number etc., if a match is found it simply performs the actions specified in the matching rule which includes forwarding the packet to a specified output port, forwarding packet on all outgoing ports, or dropping the packet. If no match is found, the packet is immediately forwarded to the controller [18].

Flow rule timeouts [18]: There are 2 types of timeouts for flow rules. IDLE timeout and hard timeout. IDLE timeout is used to remove flow table rule if there are no matching packets for the specified amount of time. Hard timeout is used to remove flow table rule after specified time, irrespective of whether matching packets arrive or not. When either of these expire, the switch removes the corresponding flow entry from its flow table.

II PROBLEMATIC AND JUSTIFICATION

DDOS attacks have been presented with higher frequency in recent years, At the end of 2020, the FBI recommended that schools implement security mechanisms against DDOS attacks as there are attacks large enough to take them out of service [5]. Kaspersky mentions that the average loss of companies being attacked by DDOS can be 20 thousand dollars per hour [6].

DDOS-type attacks may not greatly affect the profits of small or medium-sized businesses, but it can affect their status and the level of trust of their customers, in a survey of IT security professionals at Infosecurity, it stated that 42% of those surveyed, that the loss of customers trust was the worst effect of DDOS [7]. SDN-type architectures allow better control of the network, but by having network control centralized, SDNs can be the victim of attacks such as traffic flooding (flood) by botnets, which makes this architecture vulnerable and therefore it is necessary to find a way to prevent, identify and mitigate these attacks to avoid the economic loss and trust of customers due to lack of service.

SDN DDOS Problematic

SDN due to its centralized control architecture can lead to 3 main issues with DOS or DDOS attacks [12]:

Overloading the SDN controller: Packet-in messages will be stuck in the controller's queue and no more routing decisions can be taken for the new incoming flows. Flows with no forwarding entries will be stuck in the switches.

Exhausting the control plane bandwidth (switch to controller bandwidth): Switch to controller links get congested and therefore some packet-in messages might then be lost, which will delay the decision regarding the waiting flows.

Switch flow tables TCAM memory overflow: Large number of new flows could saturate the flow forwarding tables of the switches. When this happens, switches are forced to constantly add and remove flow entries and to send more packet-in messages toward the controller.

For this work it will be used an IDS using LSTM algorithm to help to identify potential attacks, the analysis of the LSTM algorithm is out of the scope of this work. Given the results of the IDS, the mitigation technique proposed is deployed to mitigate the malicious flows to warranty the service availability of the server.

This work will be more focused on the configuration of a SDN with Mininet and ONOS and implementing the mitigation strategy since there is already an intrusion detection system (IDS) provided by the advisers of this work.

III STATE OF THE ART

Due to the centralized control architecture of SDN, contributions have been made to identify and mitigate flood-type DDOS attacks.

In [3] Safaa et al, in January 2020 presented a proposal to mitigate DDOS flood attacks using a SDN architecture, they made a program with the P4 programming language, the program connects the controller with the switch to install it on the latter and checks the flow table where the TCP protocol monitors, then checks if the packet received by the client is SYN, if it is, it generates a SYN Cookie based on data from the received packet, then it is hashed to ensure the integrity of the packet and it sends it to the client, if the client responds with a TCP ACK and matches the local SYN, it is accepted and if not that traffic is rejected. They ensure that their strategy does not use much memory space since it is done directly on the switch and that 100% of the benign packets are accepted, while the bad ones are rejected.

In [8] by Nguyen et al, in February 2020 contributed in a way to identify and mitigate flood-type DDOS attacks in SDN architectures using machine learning with the Nearest-Neighbor (KNN) algorithm and XGBoost, which is an algorithm based on machine learning that belongs to the increasing gradient. The algorithm mitigates flood attacks for TCP-SYN and ICMP protocols, with an

efficiency to mitigate the attack of 98% with TCP-SYN and 99% with ICMP, benign traffic is not affected. The KNN algorithm was used in the entropy values after having calculated the ports or the number of ICMP packets. For their dataset they used the CAIDA 2007 dataset with a size of 21 GB [9].

In [9] Jesús et al, contributed with the detection and mitigation of flood-type DDOS attacks in an SDN. The algorithm works as follows, they retrieve the switch states, the entropy values are calculated for the source IP, destination IP, source port and destination port, when the entropy values fall and the algorithm detects it, depending on the size of the window, flow rules are added to the switches using OpenFlow protocol to mitigate attackers, and benign traffic remains intact. In 10 tests that were carried out, for 3 of them, the attack was not detected and mitigated, the problem that occurred was with the size of the window, which is 60 seconds, if the attack is launched just when the size of the window ends it causes the entropy values do not vary enough with the expected thresholds, and at the beginning of the next time window, no attack is detected because the entropy values did not exceed the predicted limits. The ratio of false positives detected depends on the value given to Θ of the window size, giving a percentage of 20% in false positives.

In [10] Marcos et al, contributed with a detection and mitigation technique for DDOS attacks in an SDN. For detection it is used a Gated Recurrent Units (GRU) a supervised deep learning method that have 2 sub-modules, 1st sub-module can go back to decide whether to use important historical data of the traffic flow for analysis and the 2nd sub-module that classifies them as normal or abnormal traffic. The mitigation module divided in 2 sub-modules, the 1st sub-module determines the optimal countermeasure against the detected anomaly, the 2nd sub-module sends the drop policy to the SDN controller for implementation. For detection, the proposal was tested against CNN, LSTM, DNN, SVM, LR, KNN and GD algorithms, considering accuracy, precision, recall and F-measure, showing that GRU had an attack identification success average of 97.09% of the previous variables with CICIDS2018 dataset and a 99.94% with CICDDOS2019 dataset. For mitigation, since the mitigation approach directly depends on the detection's method efficiency, the tests were considered with the number of legitimate flows dropped and the malicious flows not dropped, GRU achieved the most balanced outcome, guaranteeing both the detection of malicious flows and preserving legitimate ones, for malicious flows that were not dropped, GRU achieved the most balanced outcomes in this test scenario compared with the other algorithms tested.

In [11] Metheus et al, contributed with a detection and mitigation system in SDN architectures for DDOS and Portscan attacks. The proposal has three modules, the first module employs a Deep Learning algorithm called Long Short-term memory (LSTM) that allows to predict the normal behavior on the network learning long term dependencies, formed by 3 gates, forget, input and output gate in order to choose what previous data is necessary to remember and use, then it is calculated the Shannon's entropy of non-qualitative data like source ports and ip and destiny port to work with them, the next step is to define the threshold between the predicted traffic and the real traffic using Bienaymé-Chebyshev's inequality. The second module has 2 sub-modules the first one detects anomalous events using fuzzy logic theory to help with the decision taking for anomaly detection, the second sub-module uses membership function called Gaussian membership function and the anomaly score to determine rather the traffic is normal, portscan or DDOS. The third module establish an Event Condition – Action model (ECA) to choose a proper action, in this module it is implemented a Safe List that contains IP flow attributes of legit users for a determined time of 5 minutes if there is a case of a flash crowd anomaly of benign users, the IP's and ports are compared with the safe list to not drop their traffic flow. The technique proposed were tested in 2 scenarios, one with an emulated scenario, and the 2nd one with CICDDOS 2019 dataset, the precision, recall, and AUC were tested, in the 1st scenario LSTM-FUZZY had an average of more than 99% of success with these variables, in the 2nd scenario more than 99% of success and compared to the other methods in the literature, the proposal is successful in detecting and mitigating DDOS flood attacks.

Lobna et al, in [12] proposed a mitigation technique for DOS attacks called SDN-guard, the proposed solution aims to help to solve 3 issues when a TCP-SYN DOS attack is deployed, the overload of the SDN controller when packet-in messages overload the queue of the controller, exhausting the control plane bandwidth (switch to controller bandwidth) because packet-in messages might be lost and switch TCAM memory or known like flow tables might be saturated if a large number of new flows get there. The strategy of the SDN-guard is focused on the mitigation, for the detection they use the flow results of an IDS to identify the potential malicious flows, given this flow they have 3 modules, the flow management module that selects the shortest path for benign users and the least-used links for malicious flow and also assigns a higher hard timeout value for malicious traffic and a normal hard timeout value for normal flow, the rule aggregation module that aggregates flow entries of malicious traffic to reduce

the number of entries in TCAMS, and the monitoring module that collects statistics about flows, switches and links to know the link bandwidth usage, flow throughput, switch TCAM usage. The results of the proposal is that SDN-guard succeeds in reducing the throughput of the controller incoming throughput by up to 32%, the average switch table size test decreases the number of flow rules in the switch by up to 26% compared with the baseline, with test about the number of packets sent from the source and the destination there is a 40% with the baseline packet loss compared to 35% with SDN-guard, also the round trip time it reduced with 23% because hard timeouts are set for malicious flow.

Mutalifu et al, in [13] contributed with a mitigation technique for DOS attacks in SDN, their technique is called FlowSec it consists in preventing an attack on the control plane bandwidth by limiting the number of packets sent to the controller. To achieve it, they use the meter table that is directly connected with the flow entries so they can configurate to control the rates of the flows of those packets. The algorithm needs to calculate the bandwidth consumption of each path, if the bandwidth utilization is greater than 80% of total bandwidth, then the switch rate at that moment is divided by 2 and then it is returned the switch port speed. For the experiment, the attacker sent 100000 UDP packets with a data size of 512 bytes, the rate is about 1000 packets per second about 4 mbps. When the attack is deployed and the FlowSec starts, the flow table entries are reduced from 8 mbps to 4 mbps due to dropping rules installed into the Open-vSwitch.

In [18] Sanjeetha et al, proposed a mitigation technique for DDOS attacks, the technique consists in a module that observes the number of requests sent by each switch to the controller and calculates the standard deviation of the overall traffic. It then compares the flow table requests from each switch with this calculated value, if the requests are greater than the standard deviation, the corresponding switch is identified as compromised, and identifying the compromised switch the result is sent to the controller so the controller blocks the requests from compromised switches by not processing their flow table requests.

Reviewing the previous contributions, there are some factors that can be analyzed, Safaa et al, in [3] proposes to use the SYN cookie technique in which it works as the "authentication" of other systems, its high success rate is good and also ensures that this technique can be implemented in SDN or normal architectures, however the testing was done in a range of 0 to 1000 TCP-SYN packets per second, it would be interesting to do a test with millions of packets and verify that the bandwidth and success rate are not altered. With respect to the second work by Nguyen et al, in [8], the percentage of success in the identification and mitigation of 98% is high, in addition to using machine learning so that the size of the time window is modified depending on the traffic received to achieve a better mitigation efficiency. The third work [9] by Jesús et al, also uses the use of entropy for the identification and mitigation of attacks, however they present a problem with the size of the time window since it does not adapt with respect to the flow of traffic it receives, causing that when the cycle of the time window ends, if the attack is received in that period of time, the attack will not be identified, in addition to having a percentage of 20% of false positives, which in the first and second job they don't have. The fourth work in [10] Marcos et al, proposed a GRU supervised deep learning method Gated Recurrent Units that can go back to use important historical data for its analysis having a identification success average of 97.09% of the previous variables with CICIDS2018 dataset and a 99.94% with CICDDOS2019 dataset, nevertheless its mitigation method is 100% related with the identification method efficiency it would be interesting to evaluate the drop time window usage of the mitigation module to minimize the computational cost and improve the mitigation outcomes. In [11] Metheus et al. proposed LSTM-FUZZY that is a deep learning algorithm that can use historical data in order to learn previous patterns of the normal traffic flow and abnormalities in the network, for both scenarios the 1st with a simulated attack and 2nd with CICDDOS2019 dataset had an average of success in the recall, precision and AUC of more than 99% identifying and mitigating Portscan or DDOS attacks, nevertheless it would be interesting to test with more scenarios with other topologies and to incorporate mitigation policies to meet new demands that might emerge in SDN environments. In [12] Lobna et al proposed a technique for mitigating TCP-SYN DOS attacks setting a higher hard timeout value for malicious flow and sending them to the least utilized links in terms of bandwidth and setting a normal hard timeout for normal flows and forward the traffic to the shortest path, the proposal helped to reduce the overload of the SDN controller, avoid exhausting the control plane bandwidth usage and the TCAM memory, it would be interesting to prove the performance of the proposed solution in larger-scale environments and also to try a estimating flow threat probability to identify false positives and the impact of malicious flow accuracy on the SDN-Guard performance. In [13] Mutalifu et al, proposed a technique called FlowSec for mitigating DOS attacks, the proposal consists in depending on the bandwidth utilization of the controller if its utilization is more than 80% of total bandwidth then the switch rate is divided at half and the malicious flows are

mitigated, it would be interesting to have more experimental tests with other topologies, and also more statistics of the performance of the proposal taking into account the inherit inaccuracy and possible delay presented in topologies with large link latency. In [18] Sanjeetha et al proposed a mitigation technique for DDOS attacks, the technique consists in a module that observes the number of requests sent by each switch to the controller and calculates the standard deviation of the overall traffic, then it is compared the flow tables requests from each switch with the standard deviation previously calculated, and then the compromised switch is detected and then the controller is programmed to block the requests from compromised switches, one interesting observation is that all the switch requests are blocked as it could be very effective to delete all malicious traffic, also the benign users traffic could also be dropped.

The purpose of this investigation is to implement a mitigation technique for DDOS flood attacks for TCP or UDP flood attacks in an SDN with Mininet and the ONOS controller. The proposal of this work it is a variation of the SDN-GUARD in [12].

IV DEFINITION OF OBJECTIVES

GENERAL OBJECTIVE

The objective of this work is to implement a modification of the SDN-GUARD in [12], a mitigation technique for high volume DDOS attacks like flood TCP-SYN or UDP in a simulated environment with mininet and the ONOS controller.

SECUNDARIES OBJECTIVES

1 Review and identify different techniques of mitigation of complex DDOS attacks for high volume like, SYN flood, UDP flood and Domain name Server amplification attacks. Make an analysis of the mitigation's strategies with its advantages, requirements, and limitations.

2 Implement the proposed mitigation strategy in the SDN architecture using the ONOS controller.

3 Test the mitigation strategy proposed.

4 Finally report the results technically and scientifically as a shape of tesina.

V CONTRIBUTIONS AND EXPECTED PRODUCTS

At the end of this work, it is expected to deliver an implementation of mitigation technique for flood-type DDOS attacks for TCP or UDP protocols in a virtual machine with an SDN in a simulated environment with Mininet and the ONOS controller, with the trained IDS given by the advisers of this work, implementing a variation of the SDN-GUARD in [12] mitigation strategy that could improve the performance of the strategy helping with 3 main issues that SDN could lead if it is attacked by DDOS, like the overload of the SDN controller, bandwidth consumption and TCAM or flow tables availability.

Also, it is expected to deliver a tesina with the review of several techniques for mitigating DDOS high volume attacks like flood attack, and with the analysis, tests, evaluation of performance, response time and scalability of the proposed algorithm in code for mitigation of flood-type DDOS attacks.

VI METHODOLOGY AND ACTIVITY PLAN

In the fig 7, we can see that the project is divided into 4 phases. The first is the identification of different mitigation techniques in complex DDOS attacks, where mitigation strategies for large volumes of attacks such as SYN flood, UDP flood, and Domain name Server attack amplification will be investigated, analyze the mitigation strategies found, mention its advantages, requirements, and limitations, identify mechanisms for the validation of the mitigation of these attacks. The second phase is to implement the mitigation strategy, identify the IDS configuration that is based on Deep learning, recognize the available parameters that the IDS provides to properly define the mitigation mechanisms, improve an existing strategy to mitigate high-volume attacks. In the third phase, it is to evaluate the implemented solution, perform online tests of the proposed strategy using the existing testbed, and evaluate the evaluated parameters based on the previously reviewed literature. In the fourth phase, report all the technical and scientific discoveries of the project.

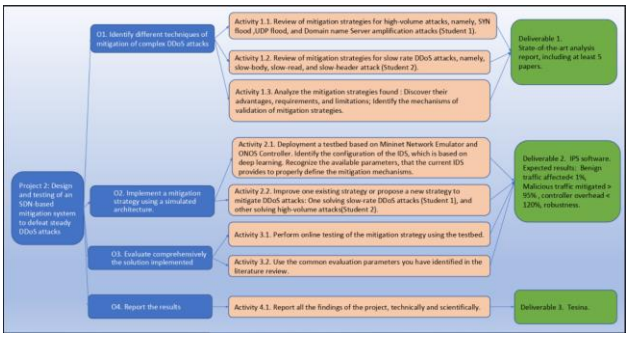
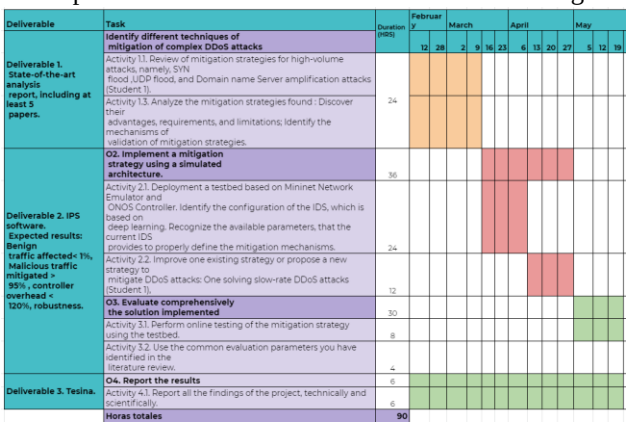


Fig. 7. General objectives

To complete the objectives of the project, a plan was developed that will be defined as in the fig 8.



Deliverable	Task	Duration (HRS)	February		March				April				May			
			12	28	2	9	16	23	6	13	20	27	5	12	19	
Deliverable 1. State-of-the-art analysis report, including at least 5 papers.	**Identify different techniques of mitigation of complex DDoS attacks**															
	Activity 1.1. Review of mitigation strategies for high-volume attacks, namely, SYN flood ,UDP flood, and Domain name Server amplification attacks (Student 1).	24														
	Activity 1.3. Analyze the mitigation strategies found : Discover their advantages, requirements, and limitations; Identify the mechanisms of validation of mitigation strategies.															
Deliverable 2. IPS software. Expected results: Benign traffic affected< 1%, Malicious traffic mitigated > 95% , controller overhead < 120%, robustness.	**02. Implement a mitigation strategy using a simulated architecture.**	36														
	Activity 2.1. Deployment a testbed based on Mininet Network Emulator and ONOS Controller. Identify the configuration of the IDS, which is based on deep learning. Recognize the available parameters, that the current IDS provides to properly define the mitigation mechanisms.	24														
	Activity 2.2. Improve one existing strategy or propose a new strategy to mitigate DDoS attacks: One solving slow-rate DDoS attacks (Student 1).	12														
	03. Evaluate comprehensively the solution implemented	30														
	Activity 3.1. Perform online testing of the mitigation strategy using the testbed.	8														
	Activity 3.2. Use the common evaluation parameters you have identified in the literature review.	4														
Deliverable 3. Tesina.	**04. Report the results**	6														
	Activity 4.1. Report all the findings of the project, technically and scientifically.	6														
	Horas totales	**90**														

VII PROPOSED SOLUTION

LSTM

For this proposal it is used a deep learning neural network called Long Short-Term Memory with an accuracy of 95% in detecting potential malicious flows, the results obtained by this Intrusion Detection System (IDS) are the flows with the following data:

IP source: The IP of whom started the communication.

IP destination: The destination IP of whom started the communication.

Source port: From which port comes whom started the communication.

Destination port: To which port arrives whom started the communication.

Protocol Id: What type of protocol is being used for the communication.

Timestamp: The date, hour, minute second in which the flow is detected.

Real label: If it is an attacker or normal flow. It is possible to know this because there are used specific hosts that do normal communication and there are used others specific hosts for the attack. By getting the IP of the attackers and normal users it is possible to know this information.

Predicted label (normal, syn, udp, dns, incomplete): This is the prediction of the IDS regarding the traffic behavior when the flows are received.

Total flows: The number of accumulated flows since the IDS started to detect.

PROPOSAL

The solution proposed in this work have some variations of Lobna's proposal in [12] but the most important are the next ones. The first variation consists in finding the shortest path in terms of bandwidth using the Dijkstra's algorithm to redirect the suspicious flow to those paths, Lobna's contribution do not search the shortest path, the aim of searching the shortest path with Dijkstra's algorithm is that in case of false positives, normal flows can reach their destination with less latency. ONOS have an application for packet forwarding, but that application uses the shortest hop-count as best path calculation and not considering the bandwidth [21]. The second variation consists in a temporally blacklist that stores the malicious flows one time, if the suspicious flow is detected again as malicious and it is in the blacklist then it is installed a drop rule to end with the communication for that traffic flow, the aim of doing this is to reduce false positives.

ALGORITHM

The steps of the mitigation proposal are shown in Fig 6.

1 The first step is when an incoming flow arrives whether being normal or malicious.

2 The flow arrives to the IDS. The IDS detects if the flow is malicious or not.

3 It is checked if the predicted tag result from the IDS is malicious or not.

4 If the flow is detected as a normal flow, then that flow it is forwarded as normal to the shortest hop-count destination switch.

5 If the flow is detected as malicious then it is verified if the flow is already in the temporally blacklist.

6 In case that the flow is not in the blacklist, then it is added to a temporally blacklist.

7 Then it is calculated the bandwidth of all the links in the topology.

8 Then it is calculated the shortest path using the Dijkstra's algorithm setting the costs considering the bandwidth of the links calculated previously. Then parallel go to step 11.

9 Finally the malicious flow it is forwarded to the least used links in terms of bandwidth.

10 In the case that the malicious flow it is already in the blacklist then it is installed a drop rule to end with the

communication between attacker and victim in the nearest switch of the detected flow as malicious.

11 It is started a timeout to delete the suspicious flow from the temporally blacklist when the times expire, the reason of this is that any flow can have access to the server again in case of false positives or in case that an attacker gets not corrupted one day. Also, to do not have in ONOS' controller memory for long time these flows. Then it goes to step 12, where until the timeout for the flow ends, the flow will remain in the temporally blacklist. Then when the timeout ends, in step 14 the flow is deleted from the temporally blacklist.

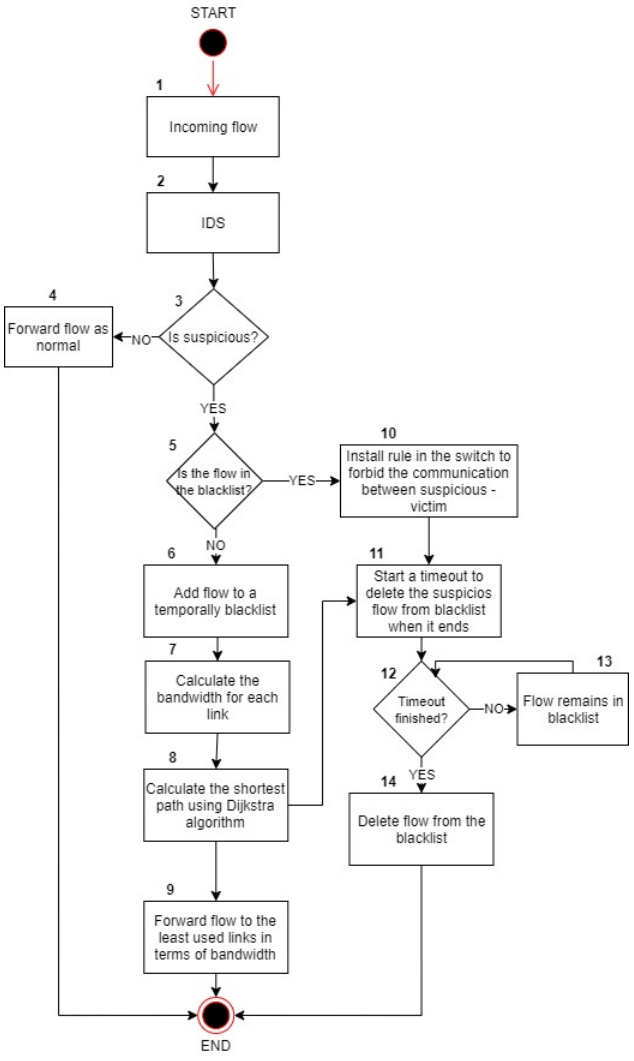


Fig. 6. Mitigation strategy diagram

ARCHITECTURE

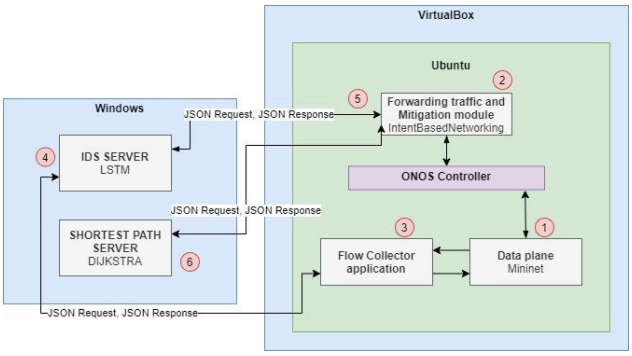


Fig. 7. Overall SDN Architecture

Figure 7 shows an overall architecture of the SDN that it is used for this mitigation strategy. The SDN architecture is divided in two parts, the first one is the intrusion detection system (IDS) and the second one is the intrusion prevention system (IPS). First the traffic is generated in the data plane (1), then (2) the Intent Based Networking application it is responsible for forwarding the traffic from origin to its destination using the shortest hop-count as best path calculation, then (3) the flow collector application collects all the flows from the data plane and then it passes these flows to the IDS server in step (4), the IDS server detects if the flows received are malicious or normal, then in (5) the intent based networking application requests to the IDS server its results, the IDS server in (4) responds with the results obtained, then in (5) the intent based networking application receives the prediction of the flows with the IP source, IP destination, predicted label and real label and then the mitigation strategy starts as it is shown in Fig 6. When the prediction from the IDS is received in the intent based networking application, then there are obtained the links of all the topology and the source and destination switches location of the corresponding hosts obtained from the IDS in order to send to the Dijkstra sever in step (6) the switches links and origin-destination switches from the source and destination host to calculate the shortest path with the Dijkstra's algorithm. Then the Dijkstra server responds with the result of the shortest path and finally the intent based networking application forwards the malicious traffic to those paths obtained.

IMPLEMENTATION

The implementation in the intent based networking application needs 4 modules. It is important to highlight that switch of:00000000000065 and the host in it, are just used as a mirroring switch in order to collect all the incoming flows and pass them to the IDS server, so no traffic from the mitigation strategy will be forwarded to that switch and also the bandwidth of the links connected

to this switch as well as the connection ports connected to this switch are not obtained.

The first module calculates the bandwidth of all the links in the topology using the port statistics service an ONOS library that obtains the egress load for the given port in terms of bytes per second, then the result is multiplied by 8 and divided by 1000 to convert it in kilobits, this is calculated for the origin and destination ports, then for obtaining the bidirectional bandwidth of the links, the source bandwidth link is summed to the destination bandwidth link and divided by 2.

$$srcBandWidthInMegabits = \frac{bandwidthInBytes * 8}{1000\ bits}$$

$$dstBandWidthInMegabits = \frac{bandwidthInBytes * 8}{1000\ bits}$$

$$bidirectionalBandwidth = \frac{srcBandWidthInKilobits + dstBandWidthInKilobits}{2}$$

Then it is obtained the switch location of the source host and the destination host to obtain from what switch to what switch is the Dijkstra algorithm is going to search the shortest path. For example as in fig 8, if the host 10.0.0.5 is detected as malicious and its destination is host 10.0.0.1, the switch origin for the host 10.0.0.5 is switch of:000000000000003, and the switch origin for the host 10.0.0.1 is switch of:00000000000002. So, to the Dijkstra server it will be sent:

- Switch origin of:00000000000003
- Switch destination of:000000000000002

And the links with its bandwidth of all the topology.

- of:00000000000001 of:00000000000003 BW 0
- of:00000000000003 of:00000000000004 BW 0
- of:00000000000001 of:00000000000004 BW 0
- of:00000000000001 of:00000000000005 BW 0
- of:00000000000005 of:00000000000001 BW 0
- of:00000000000004 of:00000000000001 BW 0
- of:00000000000003 of:00000000000001 BW 0
- of:00000000000003 of:00000000000002 BW 0
- of:00000000000002 of:00000000000001 BW 0
- of:00000000000005 of:00000000000004 BW 0
- of:00000000000002 of:00000000000005 BW 0
- of:00000000000004 of:00000000000005 BW 0
- of:00000000000005 of:00000000000002 BW 0
- of:00000000000004 of:00000000000003 BW 0
- of:00000000000001 of:00000000000002 BW 0
- of:00000000000002 of:00000000000003 BW 0

Then Dijkstra server will respond with the shortest path found in terms of bandwidth, in case that the bandwidth is 0 or equal for the links that could pass the host source and destination, then Dijkstra server will respond with the path with the shortest quantity of hops between switch origin and switch destination (example [of:00000000000003, of:00000000000002]). So the source attacker host will go first through the switch 3 and then to the switch 2.

Now the task that the second module has to do is to obtain the source ports in which the links between each switch are connected through which the source host has to pass until it reaches the destination host. To obtain these ports it is necessary to know the path through which the source host has to pass, to know between which switches the ports must be obtained, this path of switches is the result of module one. Example, if the source host 10.0.0.5 wants to communicate to host 10.0.0.1 and the path result from the Dijkstra server is [of:00000000000003, of:00000000000002], then it is checked the source por connection between [of:00000000000003, port 1, of:00000000000002, port 2], it is only needed to know the port 1 from the of:00000000000003 switch as only through port 1 will connect to switch of:00000000000002. Then for the final switch to reach the destination in this case switch of:00000000000002, it is obtained the port connection between the final destination host and the switch post. Example, switch port between host 10.0.0.1 and of:00000000000002 would be 3. It is important to obtain the ports of the switches and the port between the final host and the switch in order to install the forwarding rules in the respective switch.

The third module installs in each corresponding switch the rules to forward the malicious traffic to the least used links in terms of bandwidth, the source port and devices id of the switches and the host port between switch and final destination host are obtained in the previous module. It is installed a rule with a priority of 120 as we can see in figure 8 in order to ensure that this rule has more priority than others forwarding rules so the traffic will pass through the corresponding shortest path in terms of bandwidth previously obtained. The rules with a priority of 10 are the ones that forward the traffic to the path with the shortest hop-count.

Finally the fourth module is the mitigation module in which depending if the incoming flow is malicious then it is added into a temporally blacklist and forwarded to the least used links in terms of bandwidth, and in the case that is already in the blacklist, then it is installed a drop rule of type “NO ACTION” with a priority of 150 and a temporally timeout as shown in figure 9 to ensure that this rule has more importance than others and that it is installed in the ONOS controller, this type of rule forbid the traffic communication between attacker – victim for a temporally given of time, then the rule is dropped from the ONOS

controller, the temporally time is to ensure that in case or false positives or that host being non corrupted one day, can have access to the server service, also to not have in ONOS controller rules that could overload its memory.

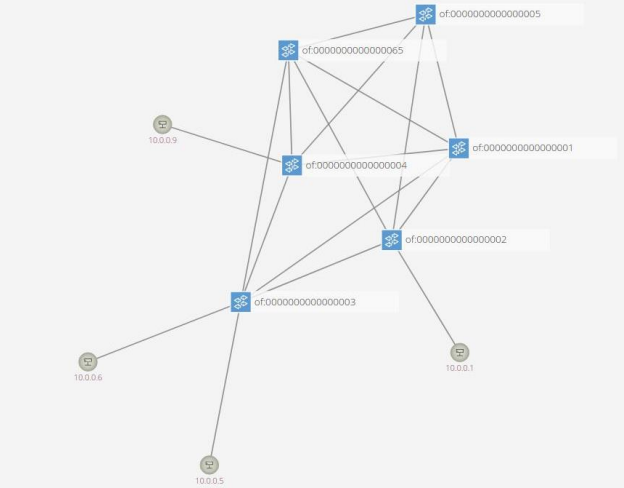


Fig. 8. Topology

FLOW PRIORITY	TABLE NAME	SELECTOR	TREATMENT
40000	0	ETH_TYPE:arp	imm[OUTPUT:CONTROLLER], cleared:true
5	0	ETH_TYPE:ipv4	imm[OUTPUT:CONTROLLER], cleared:true
120	0	ETH_DST:8A:62:D6:D4:C8:77, ETH_SRC:DA:12:B9:9C:2B:C8	imm[OUTPUT:6, OUTPUT:1], cleared:false
10	0	IN_PORT:6, ETH_DST:DA:12:B9:9C:2B:C8, ETH_SRC:8A:62:D6:D4:C8:77	imm[OUTPUT:3], cleared:false
120	0	ETH_DST:DA:12:B9:9C:2B:C8, ETH_SRC:8A:62:D6:D4:C8:77	imm[OUTPUT:4], cleared:false
10	0	IN_PORT:3, ETH_DST:8A:62:D6:D4:C8:77, ETH_SRC:DA:12:B9:9C:2B:C8	imm[OUTPUT:6, OUTPUT:1], cleared:false
150	0	ETH_DST:DA:12:B9:9C:2B:C8, ETH_SRC:8A:62:D6:D4:C8:77	imm[NOACTION], cleared:false
40000	0	ETH_TYPE:lldp	imm[OUTPUT:CONTROLLER], cleared:true
40000	0	ETH_TYPE:bddp	imm[OUTPUT:CONTROLLER], cleared:true
10	0	IN_PORT:3, ETH_DST:8A:DB:71:3B:67:D9, ETH_SRC:DA:12:B9:9C:2B:C8	imm[OUTPUT:5, OUTPUT:1], cleared:false
10	0	IN_PORT:5, ETH_DST:DA:12:B9:9C:2B:C8, ETH_SRC:8A:DB:71:3B:67:D9	imm[OUTPUT:3], cleared:false

Fig. 9. Installed Rules in switch of:00000000000003

RESULTS

SYSTEMS SPECIFICATIONS

As one of the restrictions is the quantity of memory that needs ONOS to work properly and for some experiments that could not be carried out due to this limitation, the systems specifications are the next ones:

Principal operative system:

- Windows 10 Home
- Intel core i7 6700 HQ 2.6 GHZ
- Ram memory: 12 GB
- 4 physical cores
- Storage: 1 TB
- Graphic memory: Nvidia 960 2gb

Memory dedicated to Ubuntu in VirtualBox:

- Ubuntu 18.04 LTS
- Dedicated cores 2
- Ram: 5184 MB
- Storage 30 GB

TOPOLOGY

For the experimental it is used the topology shown in figure 10. The role that each host has is the following:

Collecting flows:

Switch of:00000000000065: As already said previously, this switch and the host in it, are just used as a mirroring switch in order to collect all the incoming flows and pass them to the IDS server.

Attackers: These hosts are the ones that attack to the victim server 10.0.0.1, with a TCP-SYN high volume attack with 5000 of connections.

- Host 10.0.0.6:
- Host 10.0.0.9:

Normal host: These hosts are the ones that are establishing normal communication to the server 10.0.0.1.

- Host 10.0.0.5
- Host 10.0.0.10
- Host 10.0.0.13

Server: This is the server that is providing the TCP service so users can access to it.

- Server 10.0.0.1

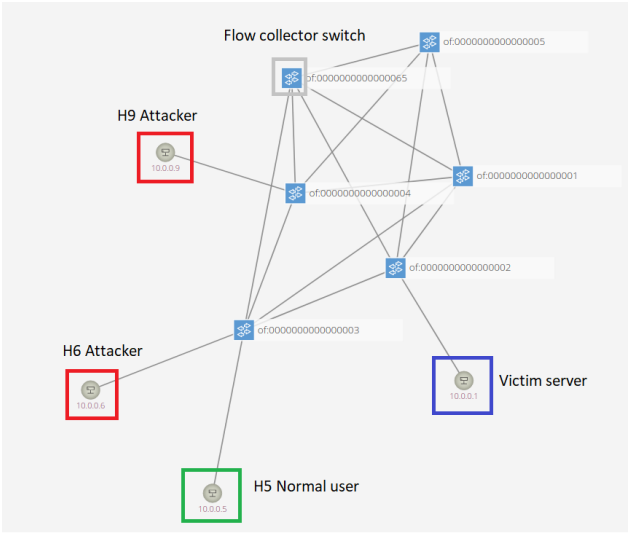


Fig. 10 Topology for experiments

Two situations were tested, with non-active mitigation strategy, and with the mitigation strategy activated.

Non-active mitigation strategy: For this test it was deployed the attack without activating the mitigation strategy, only data was collected on the behavior of the bandwidth in the whole topology depending on whether it was normal traffic or if the attack was present and the amount of time in which the network returned to normal.

Mitigation strategy activated: For this test there were tested 4 different attacks, but for the aim of study only 2 of them will be analyzed.

EXPERIMENTS

NON-ACTIVE MITIGATION STRATEGY

As we can see in Fig 11, it is shown the normal traffic flow and the attack deployed. The attack starts at second 435 and the packets per seconds trend increased and it remained until second 946 with a duration of 511 seconds, and then taking more than 40 seconds for the traffic trend to return to normal.

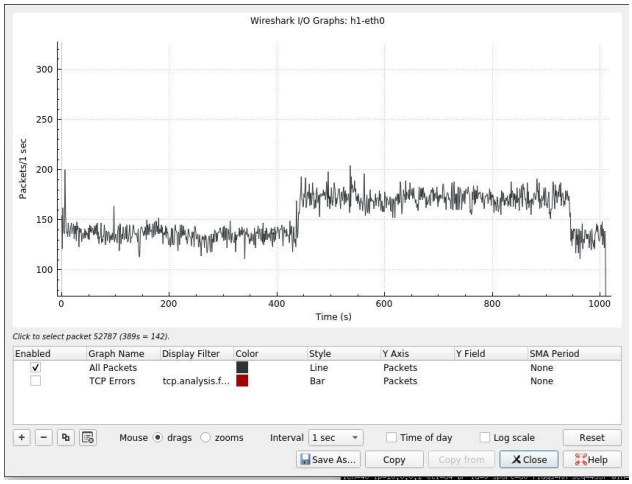


Fig. 11 Non-active mitigation strategy packets graph

As it is shown in fig 12 in which it is shown that a total of 344 flows had a prediction and were forwarded to the intent based networking application, 79 were normal flows and 265 malicious flows. As we can see any malicious flow were dropped.

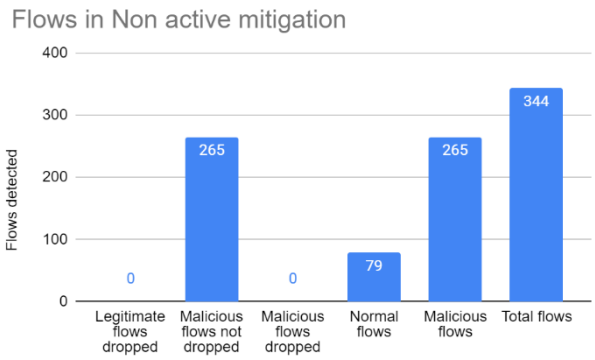


Fig. 12 Flows collected (non-active mitigation strategy)

MITIGATION STRATEGY ACTIVATED

In the first attack with the mitigation strategy activated as we can see in fig 13, it can be seen that the trend for the normal traffic remains before second 346, because in this second the attack starts and ends until second 364 with a duration of 18 seconds lesser than if the mitigation strategy would be deactivated. It took also 18 seconds for the graph to return in a normal traffic trend as before the attack.

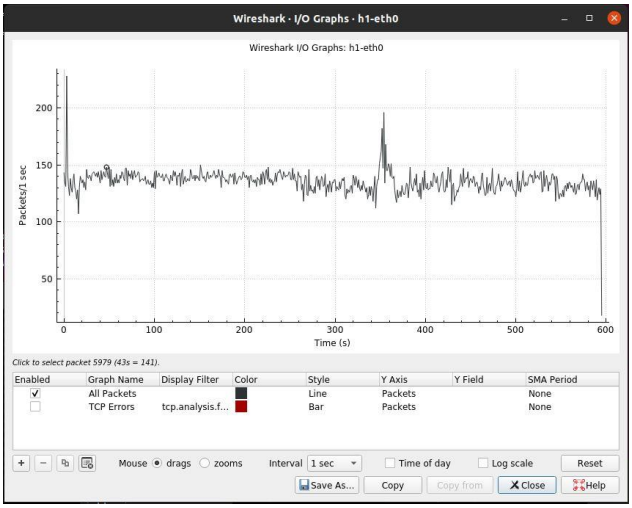


Fig. 13 Active mitigation strategy packets graph attack 1

As it is shown in fig 13 in which a total of 87 flows were detected, the application counted 8 malicious flows before it will install de drop rule for the malicious flows, so 2 malicious flows were dropped because there were 2 attackers host 10.0.0.6 and host 10.0.0.9. Also 0 legitimate flows were dropped thanks to the temporally blacklist.

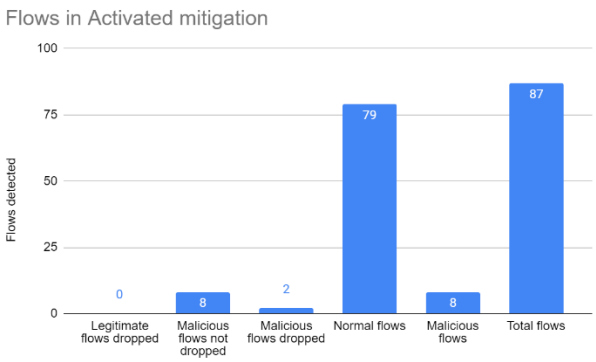


Fig. 14. Gantt diagram

CONCLUSIONS

As it can be concluded the mitigation strategy helps to distribute the traffic in a better way, as ONOS normal forwarding is the fastest hop-count, but with this proposal the use of Dijkstra's algorithm to search the shortest path or the cheapest help to maintain a better balance of the bandwidth along the network. Also, the temporally blacklist helps to ensure that false positives do not get dropped and only the malicious flows. For future work it would be interesting to recollect the bandwidth only by one direction and not bidirectional, also to search the shortest path not only for the outward journey but also for the return, as in this implementation it returns by the same path. Also it would be interesting to have an increasing timeout in case that a malicious flow continually returns over time to the server to avoid him attacking the server frequently.

REFERENCES

[1] Cisco. Cyber attack—What are common cyberthreats? (s. f.). Cisco. Retrieved 20 of February of 2021, from https://www.cisco.com/c/en/us/products/security/common-cyberattacks.html

[2] Cisco. What is a ddos attack? Distributed denial of service. (s. f.). Cisco. Retrieved 21 of February of 2021, from https://www.cisco.com/c/en/us/products/security/what-is-a-ddos-attack.html

[3] Mahrach, Safaa & Haqiq, Abdelkrim. (2020). DDoS Flooding Attack Mitigation in Software Defined Networks. International Journal of Advanced Computer Science and Applications. 11. 10.14569/IJACSA.2020.0110185.

[4] Weisman Steve. What is a DDoS attack? (2020). Retrieved 21 of February of 2021, from https://us.norton.com/internetsecurity-emerging-threats-what-is-a-ddos-attack-30sectech-by-norton.html

[5] Oleg Kupreev, Ekaterina Badovskaya, Gutnikov Alexander. Ataques DDoS en el cuarto trimestre de 2020. (2021). Retrieved 21 of February of 2021, from https://securelist.lat/ddos-attacks-in-q4-2020/92850/

[6] Karspesky. Distributed denial of service: Anatomy and impact of ddos attacks. (2021, January 13). Usa.Kaspersky.Com. https://usa.kaspersky.com/resource-center/preemptive-safety/how-does-ddos-attack-work

[7] Group, I. D. M. (2018, September 13). La pérdida de la confianza, principal consecuencia de los ataques DDoS | Security and Risk Management. Discover The New. https://discoverthenew.ituser.es/security-and-risk-management/2018/09/la-perdida-de-la-confianza-principal-consecuencia-de-los-ataques-ddos

[8] Tuan, N. N., Hung, P. H., Nghia, N. D., Tho, N. V., Phan, T. V., & Thanh, N. H. (2020). A DDoS Attack Mitigation Scheme in ISP Networks Using Machine Learning Based on SDN. Electronics, 9(3), 413. MDPI AG. Retrieved from http://dx.doi.org/10.3390/electronics9030413

[9] Galeano-Brajones, J., Carmona-Murillo, J., Valenzuela-Valdés, J. F., & Luna-Valero, F. (2020). Detection and Mitigation of DoS and DDoS Attacks in IoT-Based Stateful SDN: An Experimental Approach. Sensors, 20(3), 816. MDPI AG. Retrieved from http://dx.doi.org/10.3390/s20030816

[10] Marcos V.O. Assis, Luiz F. Carvalho, Jaime Lloret, Mario L. Proença. (2020). A GRU deep learning system against attacks in software defined networks, Journal of Network and Computer Applications, Volume 177, 2021, 102942, ISSN 1084-8045. Retrieved from: https://doi.org/10.1016/j.jnca.2020.102942

[11] M. P. Novaes, L. F. Carvalho, J. Lloret and M. L. Proença. (2020) "Long Short-Term Memory and Fuzzy Logic for Anomaly Detection and Mitigation in Software-Defined Network Environment," in IEEE Access, vol. 8, pp. 83765-83781, 2020, Retrieved from: 10.1109/ACCESS.2020.2992044.

[12] L. Dridi and M. F. Zhani, (2016) "SDN-Guard: DoS Attacks Mitigation in SDN Networks," 2016 5th IEEE International Conference on Cloud Networking (Cloudnet), 2016, pp. 212-217, Retrieved from: 10.1109/CloudNet.2016.9.

[13] M. Kuerban, Y. Tian, Q. Yang, Y. Jia, B. Huebert and D. Poss, (2016) "FlowSec: DOS Attack Mitigation Strategy on SDN Controller," 2016 IEEE International Conference on Networking, Architecture and Storage (NAS), 2016, pp. 1-2, Retrieved from: 10.1109/NAS.2016.7549402.

[14] Cloudflare. (2021). Ataque de inundación SYN. Retrieved 26 April 2021, from: https://www.cloudflare.com/es-es/learning/ddos/syn-flood-ddos-attack/

[15] Cloudflare. (2021). UDP Flood Attack. Retrieved 26 April 2021, from https://www.cloudflare.com/es-es/learning/ddos/udp-flood-ddos-attack/

[16] Cloudflare. (2021). ¿Qué es una inundación de DNS? | Ataque DDoS de inundación de DNS. Retrieved 26 April 2021, from: https://www.cloudflare.com/es-es/learning/ddos/dns-flood-ddos-attack/

[17] Imperva. (2021). What is DNS flood attack. Retrieved 27 April 2021, from: https://www.cloudflare.com/es-es/learning/ddos/dns-flood-ddos-attack/

[18] Raja, Sanjeetha & Srivastava, Shikhar & Pokharna, Rishab & Shafiq, Syed & Kanavalli, Dr. (2018). Mitigation of DDoS attack instigated by compromised switches on SDN controller by analyzing the flow rule request traffic. International Journal of Engineering & Technology. 7. 46. 10.14419/ijet.v7i2.6.10065.

[19] Abdulaziz, Abdul-hafiz & Adedokun, Emmanuel & Man-Yahya, Sani. (2017). Improved Extended Dijkstra's Algorithm for Software Defined Networks. International Journal of Applied Information Systems. 12. 22-26. 10.5120/ijais2017451714.

[20] Open Networking Foundation. (2014). SDN architecture. 27 April 2021, de Open Networking Foundation from: https://opennetworking.org/wp-content/uploads/2013/02/TR_SDN_ARCH_1.0_06062014.pdf

[21] T. Irfan, R. Hakimi, A. C. Risdianto and E. Mulyana, "ONOS Intent Path Forwarding using Dijkstra Algorithm," 2019 International Conference on Electrical Engineering and Informatics (ICEEI), 2019, pp. 549-554, doi: 10.1109/ICEEI47359.2019.8988853.