Analysing different Distributed Denial of Service (DDoS) attacks and its solutions in SDN

Dylan Durand
Supervisor(s): Chhagan Lal, Mauro Conti
EEMCS, Delft University of Technology, The Netherlands
22-6-2022

A Dissertation Submitted to EEMCS faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

Analysing different Distributed Denial of Service (DDoS) attacks and its solutions in SDN

Dylan Durand
Supervisor(s): Chhagan Lal , Mauro Conti
EEMCS, Delft University of Technology, The Netherlands

Abstract

Software-Defined Networks are an exciting network paradigm that brings many advantages to its users. However, its architecture also makes it vulnerable to attacks. Distributed Denial of Service attacks are one of those attacks that can exploit the weaknesses of an SDN. This paper explains the weaknesses that can be exploited and the consequences of a DDoS attack. State-of-the-art machine learning and statistical solutions to this problem are presented and evaluated. The limitations of each solution are analysed, and a new system is proposed that eliminates the mentioned flaws.

1 Introduction

Software-Defined Networking (SDN) is a relatively new network architecture where the data and control plane are separated within the network to allow for a centralized control plane that can control all the routers and switches using software [1]. This separation makes the switches simple data forwarding devices that can be (re)configured in one centralized place, removing the redundant way of having to (re)configure each switch individually.

The SDN controller's primary protocol to communicate with the data plane is OpenFlow [2]. This protocol uses flow tables which are stored in the switches in the data plane. These flow tables contain flow rules that tell the switch to propagate the data. However, if an unknown packet arrives at the switch, the flow rule will not have been established for it yet. When this happens, a $packet - in$ message is sent from the switch to the controller containing partial information about the new packet. The controller will then compute a new routing path (flow rule) for that message and send it back to the data plane. All the flow tables in the data plane will be updated with this new flow rule. So now, whenever a similar packet arrives, the data plane will know what to do with it.

A Denial of Service attack aims to disrupt a network by flooding a network with a large amount of unwanted traffic from a single machine [3]. Nowadays, most networks can fend off traffic from a single machine. However, distributed Denial of Service floods a target network by sending traffic from multiple machines. An adversary usually uses a botnet for this [4]. Botnets are a group of machines that have been compromised by malware. Giving the attacker control of what traffic the machines send out. Usually, the owner of an infected machine does not even know that their device is being used for malicious intent. A DDoS flood would significantly slow down or potentially crash an entire network. This is because all the traffic depletes the resources of a network, such as CPU, bandwidth, or memory.

An SDN is vulnerable to security threats due to its architecture. Especially an adversary trying to plan a DDoS attack can easily find areas to exploit the network structure. For example, consider the centralized controller that controls where all the traffic goes. This gives an adversary an easy target to attack when trying to crash a server.

Fortunately, many solutions have been researched that reduce the effect that a DDoS attack can have on an SDN.

This paper presents a survey that will analyze the different types of DDoS attacks in an SDN and provide state-of-the-art solutions and propose a new design of a solution. Firstly, the works from other research will be acknowledged. Secondly, the different attacks and threats of DDoS in an SDN will be explained. Thirdly, the state-of-the-art solutions against DDoS proposed by other researchers are presented. Limitations of the solutions and a comparison will be presented fourthly. In the fifth section, a new solution will be discussed which overcomes the aforementioned limitation. The sixth section explains how the literature study of this survey was done ethically and responsibly. Lastly, the conclusion will summarize the paper and elaborate on future works.

2 Related works

In recent years, much research has been done regarding the general security of an SDN [1]. However, few surveys have been done regarding DDoS attacks specifically. Table 1 shows the different surveys focused on analyzing DDoS attacks within an SDN.

In [5] a survey of DDoS attacks within an SDN and cloud computing is presented. This gives an excellent overview of the current DDoS attacks that can be carried out. However, since it also focuses on cloud computing, it lacks information regarding solutions to DDoS attacks. Similarly, [6] has also shown how to use features from an SDN to mitigate a DDoS attack.

A very extensive survey shown in [7] presents many solutions to different threats. However, most of these solutions are now almost ten years old. This survey will try to provide similar information to more recent research.

The research presented in [8] provides a broad range of information regarding the DDoS attack and the threats they pose to an SDN. Included are also state-of-the-art solutions for those attacks. However, this research does not focus on any machine learning solutions.

This paper will explain the different forms of attacks and the impacts they can have on an SDN. It will also survey the different state-of-the-art solutions currently available. Finally, these solutions will be analyzed in terms of their performance, and limitations will be pointed out.

3 DDoS in SDN

This section will focus on the different threats an SDN faces, DDoS attacks that can be carried out within an SDN, which entities are impacted, and how the network's performance is affected [7] [5].

3.1 Threats

This subsection describes the different vulnerabilities an adversary could exploit in an SDN using DDoS.

Flow table overflow

When a switch's flow table is filled up with flow rules, it will automatically eliminate entries using the Least Recently Used principle. This elimination also happens if the number of active flows is greater than the number of entries in the flow table, resulting in active flows being kicked out and must be installed in the flow table again. Due to this mechanism, an adversary could easily overflow the flow table by sending numerous malicious packets to the switch, insinuating that the active flows will have to be eliminated. This means that useless flows from the attacker replace benign users' flow rules, leading legitimate flows to be handled barely, rendering the service for the users useless.

Controller resource saturation

When a new packet arrives in the network, the switch must set up a new flow rule. This is done by sending $packet - in$ messages from the switch to the controller. The controller will then have to calculate the routing path, encapsulating the new flow rules into the control message and sending that message to the data plane. The controller can usually do this process with no problem. However, if malicious packets flood the switches, simultaneously sending $packet - in$ messages to the controller, it will exhaust the controller's CPU and memory. This degrades the network's performance as legitimate requests will not be handled quickly.

Data-to-control plane saturation

The $packet - in$ messages sent from the switches only contain partial information about the original packet. This is done not to overcrowd the communication channels between the data and the control plane. The incoming packet is then buffered in the switch's flow table whilst it waits for the new flow rules from the controller. This buffer will fill up when the network gets flooded with malicious packets. Once that happens, the entire packet will be sent over the communication channel. These packets could then collide with outgoing messages from the controller and congest the entire channel. This channel can be seen in Figure 1. Benign users will then face the unavailability of the services.

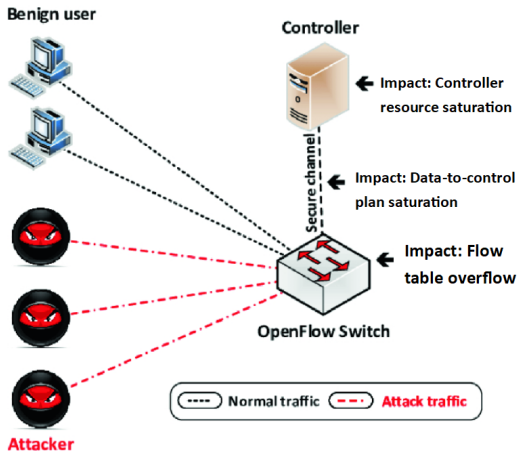

Figure 1: Shows the impact of DDoS attacks in an SDN

3.2 Attacks

This subsection describes how an adversary could launch a DDoS attack on an SDN and how the network is impacted.

UDP Flood attack

UDP flood exploits the fact that the targeted server will constantly check for a listening application after sending a UDP packet. As a result, the server will return an ICMP destination unreachable packet after a certain period. In an SDN, if this happens with many packets, the controller's resources will reduce, causing delay or even a complete server shutdown. In addition, the controller is affected as it will have to continuously keep many connections that are checking for any matching applications that do not exist, causing controller resource saturation.

DNS amplification

An adversary could use zombie hosts to send many DNS queries to a server using the victim's IP address as the destination. This could, for example, be an OpenFlow switch inside an SDN network. In this event, the switch would receive many DNS response messages that would fill up the buffer, which could cause data-to-control plane saturation.

HTTP Flood

When large amounts of HTTP GET and POST requests are sent to a targeted SDN network, the server tries to establish an HTTP connection. However, the initial connection will never be completed as spoofed IP addresses are used. These incomplete connections will saturate the controller, and eventually, benign users cannot make a connection. This controller saturation will delay the responses to the benign users and could, in the worst case, even shut down the server.

Title	Year
Research trends in security and DDoS in SDN [7]	2016
DDoS attack detection and mitigation using SDN: methods, practices, and solutions [9]	2017
A survey on distributed denial of service (DDoS) attacks in SDN and cloud computing environments [5]	2019
New-flow based DDoS attacks in SDN: Taxonomy, rationales, and research challenges [8]	2020
Detection and mitigation of DDoS attacks in SDN: A comprehensive review, research challenges and future directions [10]	2020
Towards DDoS detection mechanisms in software-defined networking [11]	2021

Table 1: Surveys of DDoS in SDN

TCP SYN attack
This attack exploits the connection protocol of the internet. An adversary client sends an SYN request to the server. SYN + ACK pair is sent back to the client, and the server now awaits an SYN packet from the client. However, if the client does not send an ACK packet back deliberately, the server will wait for some time. If this happens within an SDN, many half-open connections on the controller will be maintained for a while, depleting its resources and denying benign users to establish a connection.

4 Solutions

This section will elaborate on the state-of-the-art solutions that have been surveyed, as shown in Table 2. These have been divided into two categories; Machine learning and statistical-based solutions.

4.1 Machine learning

Machine learning techniques are used to classify network flow as malicious or benign by utilizing different features from the traffic.

SVM
Using a Support Vector Machine to classify the incoming network packets has been used in [12]. This defence framework is divided into three parts, the traffic collection module, the DDoS attack identification module, and the flow table delivery module. The traffic collection module is only called when the IP entropy detection result constitutes a DDoS attack. Then the module will extract eight information features using a sniffer and a monitor for network traffic information. These features are *count, srv_count, same_srv_rate, dst_host_count, dst_host_srv_count, dst_host_same _src_port_rate, dst_host_serror_rate, dst_host_rerror_rate*. From this, much information can be gathered regarding host- and time-based network traffic. A feature vector is created from this and used as input for the SVM. The classifier will output a -1 if it is a DDoS packet. Otherwise, it is considered a normal packet. The flow table delivery module will forward the traffic unless it has been classified as a DDoS packet; then, it will be dropped.

The KDD99 [19] data set was used to train the classifier. The data set contains entries with five major categories under which normal and DoS are included. These two labels are then selected, and the data set is divided into training (75%) and test (25%) sets. The evaluation was done by measuring the accuracy (as seen below) of the DDoS attack classifier.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Based on this evaluation, the classifier had an accuracy of 0.998.

K-means
K-means, another machine learning classifier, was suggested to be used as a DDoS mitigation system in [13]. The system is composed of two parts. The first is used for detection and location of the attack, using the K-means clustering algorithm. In addition, a traffic dictionary is used to provide information about each OpenFlow switch and locate at which switches the attack is happening. The second part is the defence mechanism, filtering $packet - in$ messages. The system's structure is shown in Figure 2.

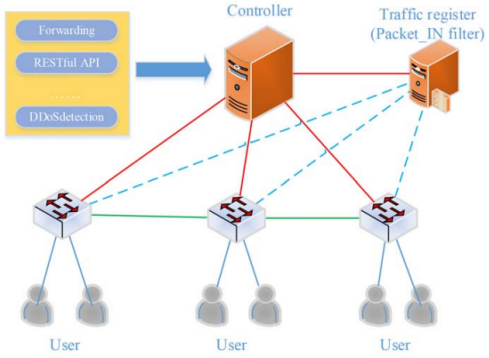

Figure 2: K-means system architecture [13]

The detection module is designed to detect low-traffic flow attacks. These attacks are numerous but relatively small in size. The attacker utilizes this as many of these small flows from the switches will become a giant attack flow against the controller. The K-means clustering algorithm can distinguish low-traffic flows by creating a traffic distribution. The following statistics are acquired from the OpenFlow switches; existing time per flow, $Duration$; packets per flow for the existing time, $PacketCount(PC)$; packets per flow during the recent time, $RecentPacketCount(RPC)$. The $HitRate(HR)$ is calculated by:

$$HitRate = \frac{PacketCount}{Duration}$$

The feature vector ($PacketCount$, $RecentPacketCount$, $HitRate$) is used as input for the K-means algorithm. The algorithm will provide the traffic distribution of input flows. Two clusters are created to be able to distinguish low-traffic

Title	Category	Methodology
SVM [12]	Machine learning	Uses SVM classifier to detect DDoS packets
K-means [13]	Machine learning	Uses the K-means clustering algorithm on the incoming packets to detect low-traffic flows
vSwitchGuard [14]	Machine learning	Different classifiers were tested. K-NN and variational autoencoder performed best
DETPro [15]	Machine learning	A modified decision tree is used to detect DDoS packets.
SAFETY [16]	Statistical	Entropy based method using a dynamic threshold
WIsdomSDN [17]	Statistical	One-to-one mapping of DNS requests and entropy is used.
DAISY [18]	Statistical	Uses a threshold to identify a DDoS attack

Table 2: Surveyed solutions

flow more easily. Each feature vector is individually labelled. This helps identify which switch produces the most low-traffic flows. Once the attack has been detected, the filter module will be activated. The traffic register filters the $packet - in$ messages from the switch that was targeted by the attack. $Packet - in$ messages that have not frequently appeared will not be sent to the controller.

The DARPA 1999 Intrusion Detection data set [20] was used to evaluate this defence mechanism. This data set consists of two-week training data and one-week testing data. The scales of attacks used for evaluation vary between 100-500 packets per second. The detection rate improves for attacks with a higher packet per second rate (PPS). At 100 PPS, $\approx 70\%$, at 500 PPS, $\approx 96\%$. The CPU utilization increased to about 13% during an attack and decreased back to normal after about 5 seconds.

vSwitchGuard

In [14] three supervised and four semi-supervised machine learning classifiers were evaluated against different saturation attacks to see which would work most effectively. The proposed vSwitchGuard consists of four modules; network topology manager, feature extractor, victim switch detection, and countermeasure. The network topology manager extracts the SDN topology and can extract the network's topology based on the connected OpenFlow switches. The feature extractor provides the classifier with four message header features (Packet-in, Packet-out, Packet-Mod, TCP-ACK), two message payload features (entropy of source IPv4 addresses, Table-Miss Packet Rate), and the combination of them (all six features). The researched supervised classifiers were K-Nearest Neighbour, SVM, and Naive Bayes. The semi-supervised classifiers include one-class SVM, isolation-forest, basic autoencoder, and variational autoencoder. The reason semi-supervised classifiers were researched is that supervised classifiers require the training data set to include all the types of saturation attacks possible. If a new saturation attack presents itself, the new pattern would be classified as benign by the classifiers. Therefore, semi-supervised classifiers can be trained without labelling all the samples. Among the four semi-supervised classifiers, the variational autoencoder achieved the best performance. Once the victim switches have been identified, the countermeasure module will be activated. Firstly, the Packet-in Deep Inspection Filter will be applied. The module will inspect the data for all $packet - in$ messages from the victim switch and extract the source and destination IPv4 address, MAC address, and port numbers. These features will be compared to the network topology to see if they are malicious packets. Any packets that are labelled malicious will trigger the blocking-rule manager. A blocking-flow rule will be installed on the identified victim switch from which the malicious packets are coming. Based on the offline evaluation, the K-NN and variational autoencoder were the best supervised and semi-supervised classifiers, respectively.

The online evaluation was done using a combination of the K-NN and variational autoencoder classifiers. Thirty-one case studies were performed, each including a period of regular traffic followed by an attack. From these 31 studies, the countermeasure model took about 2 7 seconds to recover the victim switches. In addition, one case study was used to show how quickly the CPU utilization recovers after an attack. During an attack, the CPU usage was around 95%, and after 2.2 seconds (including detection and countermeasure), the CPU usage was at 45%.

DETPro

The DETPro system [15] uses a modified decision tree to identify malicious traffic. Initially, the information about the traffic has to be collected. The data is gathered using the OpenFlow information collection module. However, when a DDoS attack is ongoing, the OpenFlow data path will become congested, making it almost impossible to gather data during an attack. That is why the mentioned module is combined with a sFlow-based information collection module [21]. This collector can still sample packets, even during a DDoS attack. The collected information is then passed through the trained modified decision tree. If a DDoS packet is detected, the IP address will is checked to see if it is on the white list. This list keeps track of all the IP addresses that are benign. If the address is not on this list, a new flow rule will be made that drops all the flow with this source IP address. Therefore, the modified decision tree can correctly detect an attack 98% of the time. Besides this, it only has a detection delay of about 2 seconds.

4.2 Statistical approach

SAFETY

This solution proposed by [16] incorporates entropy to try and detect DDoS traffic. Entropy is used to measure randomness in a variable. The higher the entropy, the more random it is and vice-versa. This calculation is done on three parameters of a network packet, namely, destination IP address, destination port, and TCP flags. With regular traffic, there is a large variety in the number of IP addresses and ports. However, more traffic with the same IP address and ports will

come through the network if an adversary is at play. This decreases the randomness and so increases the entropy significantly. The traffic will be labelled malicious if this entropy value is above a certain threshold for a specific period. The threshold must consider the dynamism of network traffic. An adaptive threshold is implemented to overcome this. All the entropy values above the initially set threshold will be considered benign and stored in a buffer. When the buffer gets full, the mean and standard deviation of these collected entropy values will be calculated. Then, a new threshold will be established to consider the current network traffic flow. The switch where all the malicious traffic comes through is known as the Attachment Point (AP). The mitigation module ensures that an SYN flood will no longer happen from the AP but still allow benign TCP connections. Only some concurrent half-connections (SYN requests) will be allowed. A new SYN request will only be allowed when one half-connections receives an ACK.

Two scenarios were used to evaluate this system. From the first scenario, it can be seen that the response time for benign users remains minimal using the SAFETY system. It is around 3 seconds, whilst it is around 0.5 seconds without any attack. Furthermore, it is shown that CPU usage only increases to about 12% in this scenario, while if there were no security, it would have gone up to at most 45%. In the second scenario, similar results are seen as the response time for benign users only goes up to around 4 seconds. CPU usage, again, only goes up to around 12% whilst without security, it goes up to 28% at its peak. Therefore, in both scenarios, a 100% True Positive Rate applies and a False Positive Rate of around 27%.

WisdomSDN

A mitigation technique is proposed for DNS amplification attacks known as WisdomSDN [17]. This system consists of three parts; Proactive And Stateful scheme (PAS) that maps DNS requests and responses one-to-one, a detection module, and a mitigation scheme. In PAS, a DNS response is considered benign if it has the same reversed field values as a previously sent DNS request. Otherwise, it is considered illegitimate, and it will be eliminated. The detection module aims to protect the TCAM of an OpenFlow switch. Firstly, network traffic features are collected using sFlow [21]. Then, flow sampling is used, which makes extracting features more scalable and efficient. The extracted features are; the source and destination MAC address, source and destination IP address, the UDP source and destination port of the packet, and the transport protocol used. These features are used to compute the entropy values of the network traffic. Then based on the calculated entropy values, a binary classifier will classify illegitimate DNS requests. Finally, when malicious requests have been identified, the mitigation scheme will install new flow rules into the switches, dropping the malicious flow.

Attacks were run on a simulated SDN environment to evaluate this defence mechanism. These attacks varied from 100 to 500 Mbps. It is seen that the victim's bandwidth does not decrease during the attack when WisdomSDN is deployed. Whereas without security, the bandwidth drastically decreases, all the way to 0. the time it took to mitigate the attack after it was initiated was less than 13 seconds. Two scenarios were run to calculate the TPR and FPR of WisdomSDN. For both, a 100% TPR was established and a 23% and 21% FPR.

DAISY

DAISY [18] is a detection and mitigation scheme that blocks malicious traffic rather than the whole port. This mechanism has two major components: parameters and functions, where the parameters are fixed before start-up. $Iterationinterval$ is the time between consecutive iterations of the DAISY mechanism. This time interval can be from 2 to 10s depending on how quickly the attack should be detected and on the controller's processing power. $FlowRequestLimit(FRL)$ is the maximum number of $packet - in$ messages a benign host can send during a DAISY time interval before it is flagged as suspicious. $WarningThreshold$ represents the limit on how many times a host crossed the FRL during a period. If this threshold is exceeded, a flow rule that stops corresponding traffic for a short time (around 2 to 6s) will be installed. $BlockingThreshold$ also limits how many times a host exceeds the FRL. This threshold is usually set as double the value of the warning threshold. If this threshold is surpassed, the corresponding traffic is blocked for longer. $IdleandHardTimeouts$: Idle timeout is the time after which an inactive flow rule is removed. Hard timeout is when a flow rule gets eliminated, whether active or not.

DAISY consists of four functions: data collection, threat detection, attack prevention, and threat value reduction. Data collection retrieves information from each arriving packet-in message. The collected data consists of switch-ID, in-Port, Ethernet type, source MAC address, destination MAC address, Protocol, source IP address, destination IP address, source port, and destination port. The threat detection checks to see if a specific combination of the collected data exceeds the FRL. If so, then the attack prevention is called, and the content of the combination is passed to it. Once the attack prevention function is called, it creates a threat entry. These are stored in a table called Malicious Flows. It will create a new entry for each new flow combination. If the entry already exists, its threat value increases. If this threat value exceeds the warning threshold, then a flow rule is installed to drop traffic for a short time. If the blocking threshold is exceeded, the traffic will be dropped for longer. The threat value reduction function aims to gradually decrease the threat values and the duration of the blocking flow rules.

The evaluation was done using three different scenarios, each consisting of a different number of attackers and targets. It is shown that during an attack, the CPU utilization is reduced by 3%, 40% and 12% in their respective scenarios. The higher reduction percentages are due to the larger attack volume, which significantly increases CPU usage.

5 Limitations and comparison of solutions

This section discusses the limitations of the solutions mentioned earlier. Along with this, a comparison is made among the solutions.

5.1 Limitations

The advantages of each solution also accompany the limitations provided in this subsection.

SVM

Benefits: Works very well on the data set that was used to evaluate its performance (0.998 accuracy).

Limitations: No network features such as CPU or bandwidth were used to evaluate the system. Thus, it does not provide insight into how well attack mitigation would work in practice. Also, no time delay was mentioned, which is an essential factor as the implementation of a mechanism should not significantly hinder the network's performance.

K-means

Benefits: provides a high detection rate on the provided data. Along with this, multiple network metrics were also tested, including CPU, bandwidth, time delay and packet loss rate.

Limitations: The mechanism only focuses on low traffic DDoS attacks. Such type is hard to detect and is usually undetected by other implemented solutions. As the focus is only on these specific attacks, it already assumes that there is a mechanism in place that can handle larger saturation attacks.

vSwitchGuard

Benefits: As multiple supervised and semi-supervised classifiers were tested, the best combination of them could be deployed during evaluation leading to a high recognition rate.

Limitations: No unsupervised classifiers were tested which could have been very effective as well (as seen in [13]). Also, more classifiers could have been evaluated, which could have improved performance even more. The evaluation does not consider how the benign users would be inconvenienced (connection delay) if their solution is implemented.

DETPro

Benefits: A PEP strategy is implemented to reduce model complexity and the running time. It has a high detection rate in its evaluation.

Limitations: The modified decision tree (the detection module) is only slightly better than a regular decision tree when there is a low attack rate. This suggests that this detection module might not be the optimal solution for attacks with a much higher attack rate.

SAFETY

Benefits: detects TCP SYN Flood attacks with 100% accuracy in the provided case studies. CPU usage and response time are also evaluated and can be seen to both remain close to normal when the mechanism is deployed.

Limitations: The proposed solution, however, assumes that there is only one victim destination node which would not be the case in a real-life scenario.

WisdomSDN

Benefits: WisdomSDN provides a solution that achieves a 100% detection rate in its evaluation.

Limitations: The DNS mitigation scheme has a fixed idle and hard timeouts for flow rules. This could lead to legitimate responses being dropped.

DAISY

Benefits: There is almost no additional delay as only some information of the $packet - in$ message is copied. It can respond fast to an attack and install the appropriate flow rules quickly.

Limitations: Due to its architecture, it does not detect any flow conflicts which could lead to forwarding loops. It also does not detect any garbage data transferred between two hosts.

5.2 Comparison

Each researched solution evaluated their mechanism differently. This makes comparing them to one another more challenging. Some solutions did happen to use similar evaluation metrics, which are presented in table 3.

Mechanism	Mitigation time (seconds)	Delay (seconds)	Detection rate
K-Means [13]	5	0.035	0.65 - 0.95
SAFETY [16]	1.2	0.145	1
vSwitchguard [14]	2.2	-	1

Table 3: Comparison table

These metrics were used for comparison as they represent important features of a solution. The mitigation time tells how quickly the mechanism can detect and dissolve the attack. The delay represents the extra time it takes benign hosts to connect with the server whilst the defence is deployed. This shows how inconvenienced a user might become. The detection rate represents how often an attack is correctly classified.

As can be seen, the K-means system has the longest mitigation time. This is due to its complex architecture and the computation time of the clusters. On the other hand, SAFETY exhibits the fastest mitigation time, which is most likely due to it only using an entropy-based method rather than a machine learning one, like K-means and vSwitchGuard.

Among the experiments used to evaluate the K-means, an average delay of 0.035 seconds (to establish a TCP link) was measured whilst the packet filter was active. This is more than four times as long if no filter was implemented (0.008s). However, once a TCP link is made, the following packets can be forwarded immediately to their destination. Thus, the quality of service is merely affected because of the filter. Furthermore, with SAFETY implemented, it also takes longer for benign hosts to make a connection, 0.145s, compared to no security, 0.0345. vSwitchGuard did not include a similar metric for connection delay as it was tested in different topologies, which means that a standard connection time could not be established.

The detection rate for K-means appears to be the lowest of the three. This rate was evaluated using different volumes of attacks, from 100 to 500 packets per second. When the attack contained 100 packets per second, the detection rate was the lowest due to the attack flow not being the majority in the flow table. This suggests that this mechanism would not suit very low-rate attacks. SAFETY and vSwitchguard both achieved a 100% detection rate in their evaluations. However, SAFETY's adversary model only includes a single victim destination node. This provides quite limited evidence that this solution would perform just as well in a more life-like scenario. vSwitchGuard was tested with different network topologies, attack types, number of victim switches and number of zombie hosts. Here, more confidence is given that the proposed solution would work well in practice.

From the machine learning-based solutions, it is possible that K-means and DETPro mechanisms will not be very robust for high traffic attacks. DETPro showed in their evaluation that it might not detect an attack as adequate as a regular decision tree for high-rate attacks. K-means is only made to detect low-rate traffic, which assumes that big floods should already get detected elsewhere. It can also be derived that machine learning solutions require a slightly longer mitigation time, as seen in Table 3.

Due to their statistical nature, DAISY and SAFETY introduce almost no time delay in their solution. This benefit could be favoured when the time delay is a crucial factor. However, SAFETY has only been tested with one victim node, which does not provide enough insights into whether it would work similarly with multiple victim nodes. Furthermore, DAISY cannot detect any flow conflicts leading to forwarding loops which could have detrimental effects on the server's resources.

6 Discussion and Future Work

This section will propose a new solution that overcomes the limitations mentioned in the previous section. Future directions for this paper are also included.

6.1 Proposed solution

Many solutions handle specific DDoS attacks (SYN flood, DNS amplification). There are currently many types of attacks, so it is hard to construct a solution that will work in all the cases. Hence, it is essential to understand which attacks are most familiar to make the proposed solution as effective as possible. TCP-SYN floods are currently one of the most common DDoS attacks [22]. As SAFETY [16] provides a workable solution to the TCP-SYN flood problem, some ideas are taken from that mechanism. First, the IP entropy method would be used to detect a potential DDoS attack. This method measures the randomness of the IP addresses of the incoming packets. The less random the variable is, the lower the entropy value. If the entropy falls below a dynamic threshold, the packets from the destination IP address will be labelled as a DDoS attack. The entropy method is used to prevent all the incoming packets from going through the machine learning classifier as this could cause significant overhead and slow down the entire network.

Once a DDoS attack has been potentially identified using the entropy method, the victim detection and countermeasure proposed in [14] will be used. The feature extractor will be called if the entropy falls below the dynamic threshold. This will provide the K-NN and variational autoencoder classifier with four message header features (Packet-in, Packet-out, Packet-Mod, TCP-ACK), two message payload features (entropy of source IPv4 addresses, Table-Miss Packet Rate), and the combination of them (all six features). The flow will now be classified, and the victim switch will also be identified if it is a DDoS attack. After this, the mitigation module will drop the DDoS flow from the switch.

6.2 Future direction

- More solutions should be studied and explained, providing more perspectives on different defence mechanisms for DDoS attacks. This, in turn, would provide a broader overview of current limitations in current state-of-the-art solutions and allow for more inspiration in proposing a new solution.
- From the research done into the different solutions, a lot of machine learning came up. However, many defence mechanisms only use one machine learning model. An interesting research opportunity would be to try and combine different (un/semi) supervised models to try and get an improved evaluation.
- The evaluation for some of the existing machine learning-based solutions only uses a data set divided into training and test data. This form of testing could lead to potentially over-fitting the data, giving a good result in their experiment, but when implemented in an existing SDN network, it might not nearly work as well. Thus, more solutions should be evaluated thoroughly to show that it performs as well as expected.

7 Responsible Research

This literacy study required the research to be conducted ethically and not provide false information. Therefore, the first subsection will explain how reliable the sources' information is. The second subsection shows the impact that this study could have.

7.1 Sources

All the information presented in this report had to be reliable and actual. The sources used were either provided by the supervisor or found in IEEE and ACM, both well-respected computer societies. Literature was taken from these libraries as the published material is constantly peer-reviewed and proven to produce reliable information. No papers published before 2017 were considered to ensure that the solutions provided are state-of-the-art.

7.2 Use of paper

This paper provides an insight into how a DDoS attack works in a Software-Defined Network. A reader with malicious intent could be inclined to use the information provided in this paper for ideas on how to carry out an attack themselves. However, this paper presents DDoS attacks that are already

well-known and existing solutions to combat these. Thus, any reader can learn new insights into this topic but necessarily use it for malicious purposes.

8 Conclusion

This paper presents the vulnerabilities of a Software Defined Network and how a Distributed Denial of Service attack could exploit these to disrupt the target server. A survey including state-of-the-art solutions is included to show different perspectives on how to combat DDoS attacks and how effective they are. The two subcategories, machine learning and statistical, exhibit different prominent features that assist in mitigating a DDoS attack. Furthermore, limitations are elucidated to show the imperfections of each surveyed solution. Finally, a new theoretical solution is proposed to combat the limitations. This includes using an IP entropy method to activate the attack detection and locating module. The K-NN and variational autoencoder classifiers are combined to decide if a DDoS attack exists. If so, a filter module is called that will drop the malicious flow from the victim switch.

References

[1] D. Kreutz, F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2014.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM computer communication review*, vol. 38, no. 2, pp. 69–74, 2008.

[3] R. Swami, M. Dave, and V. Ranga, "Software-defined networking-based ddos defense mechanisms," *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, pp. 1–36, 2019.

[4] S. S. Silva, R. M. Silva, R. C. Pinto, and R. M. Salles, "Botnets: A survey," *Computer Networks*, vol. 57, no. 2, pp. 378–403, 2013.

[5] S. Dong, K. Abbas, and R. Jain, "A survey on distributed denial of service (ddos) attacks in sdn and cloud computing environments," *IEEE Access*, vol. 7, pp. 80813–80828, 2019.

[6] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-defined networking (sdn) and distributed denial of service (ddos) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE communications surveys & tutorials*, vol. 18, no. 1, pp. 602–622, 2015.

[7] N. Dayal, P. Maity, S. Srivastava, and R. Khondoker, "Research trends in security and ddos in sdn," *Security and Communication Networks*, vol. 9, no. 18, pp. 6386–6411, 2016.

[8] M. P. Singh and A. Bhandari, "New-flow based ddos attacks in sdn: Taxonomy, rationales, and research challenges," *Computer Communications*, vol. 154, pp. 509–527, 2020.

[9] N. Z. Bawany, J. A. Shamsi, and K. Salah, "Ddos attack detection and mitigation using sdn: methods, practices, and solutions," *Arabian Journal for Science and Engineering*, vol. 42, no. 2, pp. 425–441, 2017.

[10] J. Singh and S. Behal, "Detection and mitigation of ddos attacks in sdn: A comprehensive review, research challenges and future directions," *Computer Science Review*, vol. 37, p. 100279, 2020.

[11] Y. Cui, Q. Qian, C. Guo, G. Shen, Y. Tian, H. Xing, and L. Yan, "Towards ddos detection mechanisms in software-defined networking," *Journal of Network and Computer Applications*, vol. 190, p. 103156, 2021.

[12] L. Yang and H. Zhao, "Ddos attack identification and defense using sdn based on machine learning method," in *2018 15th international symposium on pervasive systems, algorithms and networks (I-SPAN)*, pp. 174–178, IEEE, 2018.

[13] J. Cui, J. Zhang, J. He, H. Zhong, and Y. Lu, "Ddos detection and defense mechanism for sdn controllers with k-means," in *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*, pp. 394–401, IEEE, 2020.

[14] S. Khamaiseh, E. Serra, and D. Xu, "vswitchguard: Defending openflow switches against saturation attacks," in *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, pp. 851–860, IEEE, 2020.

[15] Y. Chen, J. Pei, and D. Li, "Detpro: A high-efficiency and low-latency system against ddos attacks in sdn based on decision tree," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2019.

[16] P. Kumar, M. Tripathi, A. Nehra, M. Conti, and C. Lal, "Safety: Early detection and mitigation of tcp syn flood utilizing entropy in sdn," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1545–1559, 2018.

[17] Z. Abou El Houda, L. Khoukhi, and A. S. Hafid, "Bringing intelligence to software defined networks: mitigating ddos attacks," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2523–2535, 2020.

[18] M. Imran, M. H. Durad, F. A. Khan, and H. Abbas, "Daisy: A detection and mitigation system against denial-of-service attacks in software-defined networks," *IEEE Systems Journal*, vol. 14, no. 2, pp. 1933–1944, 2019.

[19] G. Meena and R. R. Choudhary, "A review paper on ids classification using kdd 99 and nsl kdd dataset in weka," in *2017 International Conference on Computer, Communications and Electronics (Comptelix)*, pp. 553–558, IEEE, 2017.

[20] MIT, "Intrusion detection data set," 1999.

[21] InMon, "sflow-rt," 2001.

[22] M. Nooribakhsh and M. Mollamotalebi, "A review on statistical approaches for anomaly detection in ddos attacks," *Information Security Journal: A Global Perspective*, vol. 29, no. 3, pp. 118–133, 2020.