National College of Ireland

Implementing Information Theory techniques for detecting multi-vector DDoS attacks in SDN

MSc Research Project
CyberSecurity

Vishnu Poovathoor Arunkumar
Student ID: 22224785

School of Computing
National College of Ireland

Supervisor: Raza Ul Mustafa

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Vishnu Poovathoor Arunkumar
Student ID:	22224785
Programme:	Masters in CyberSecurity **Year:** 2023-2024
Module:	MSc Research Project
Supervisor:	Raza Ul Mustafa
Submission Due Date:	12 August 2024
Project Title:	Implementing Information theory techniques for detecting multi-vector DDoS attacks in SDN

Word Count: 9727 **Page Count:** 20

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Vishnu Arun

Date: 11 August 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Implementing Information theory techniques for detecting multi-vector DDoS attacks in SDN

Vishnu Arun – 22224785

Abstract

Networking architecture has seen a shift in solutions since the introduction of Software Defined Networking (SDN). This update in technology has introduced opportunities to create solutions for problems that existed in the traditional realm, while bringing in new avenues of issues that didn't exist prior to its adoption. The proposed research is done on the DDoS attack detection solutions that are implemented in SDN networks in the form of programmable SDN controllers. An attention-grabbing solution presented for tackling this utilizes Information Theory for real-time detection of DDoS attacks on an SDN network. The study is conducted on this subset of solutions, compared to other widely researched detection techniques such as machine learning and signature-based detection solutions and it offers an interesting yet effective take on protection against complex DDoS attacks. The study proposes a novel system utilizing information theory techniques to detect combinations of different complex DDoS attack patterns created by volume generation manipulation. The results obtained from testing the suggested attacks on the simulated network show the detection capabilities of the system, which are backed up by measuring consistent threshold deviations in values based on entropy that are used in the mechanism, leading to successful detection in network changes, coupled with detection times for different attack patterns carried out in the study.

1) Introduction

Software Defined Networking is a form of networking architecture that is systematically gaining traction as more and more service domains are adopting it due to the undeniable benefits that it brings to the table. SDN is seen in action when the hardware nodes in a network which is known as the data plane in SDN terminology, are separated from the software that controls it, which means that the data plane is only responsible for data forwarding activities. The component known as the controller communicates with the application plane, which runs different networking programs that are now abstracted and designed in high level programming languages to suit any desired networking use cases. SDN is seen in huge effect mainly in data centers and is a chosen design for connecting on-premise infrastructure to the cloud because of the ease of access that it provides in setting up the hardware irrespective of its vendors. This has resulted in multiple cloud service providers offering their own SDN software as part of their services.

Traditional network security has relied on signature-based techniques such as Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS), which was in effect way before the introduction of SDN concept. These systems are used at present in both traditional network and SDN contexts, but are limited in providing protection against advanced persistent threats in the SDN space, as they work by detecting known attacks instead of newer more complicated attack patterns that are yet to be identified (Shirsath et al., 2024). In a SDN environment, attacks are therefore detected using different detection mechanisms that are deployed on the controller, which are programs that utilize unique software-based solutions other than signature-based techniques. Since the architecture is relatively new when

compared to the traditional networks, the entire structure has a multitude of new security vulnerabilities. The documented research on the security issues of SDN is abundant, yet incomplete due to the complexity of the attacks that presently exist.

A DDoS attack is used to deplete the resources of a target system, making its services unavailable. The attack is carried out by an attacker by using botnets, which is a collection of compromised devices. These devices can be anything, from computers to IoT devices that can transmit data, and the attacker can program them to send useless traffic to a target, which depletes its resources such as bandwidth or processing power. DDoS attacks are classified mainly as volumetric, protocol or application based. Volumetric attacks are carried out by mass forwarding of traffic via the aforementioned botnets, protocol-based attacks are carried out by exploiting inherent vulnerabilities that are present in different network protocols that are in use to transfer data between systems, and application-based attacks leverage the vulnerabilities that are present in the application layer of a web service which can be used to spam requests. In context of SDN networks, DDoS attacks can be distinguished on the basis of the planes of the architecture, since the components in the network are structured differently.

1.1) DDoS attacks based on SDN architecture

As mentioned earlier, a SDN network structure has three planes; Application, Control and Data planes. This is the key representative characteristic of the architecture, as it refers to the migration of all of the network data control and intelligence to a centralized software-based control, which functions on the over-arching applications running network functions that can be designed using high level programming languages (Camilo et al., 2020). The components in each plane communicate with each other using different SDN based protocols that are used in the interfaces between them. The protocols that are being used for implementing SDN networks in the industry are varied at present as different vendors come up with their own protocols integrated with their hardware products. A lot of the research done in SDN networks is implemented on OpenFlow protocol for communication between the nodes and the controller. (Singh and Behal, 2020) distinguishes the security problems within the SDN structure according to the planes that have the vulnerabilities within them. They go in depth on the types of DDoS attacks that are implemented based on the target plane and what vulnerability within it is exploited. The same method of dividing the attacks is used for understanding the topic focused on in this thesis (Figure 1: Security Issues in SDN

Application plane: The application layer is the part of the SDN architecture that is used for deploying customized SDN applications which can be developed as per requirement. (Jimenez et al., 2021) discussed a variety of security issues on the entire SDN architecture in the survey, in which persistent issues on the application layer are determined to be mainly malicious apps or apps which have pre-existing vulnerabilities in them. Applications are implemented on an SDN network by using different APIs or programming languages, that communicates via the northbound interface. Most of these apps lack authentication and access control mechanisms, and can be maliciously accessed and used to implement DDoS attacks on the controllers, amongst other attacks.

Control plane: The SDN controller can be thought of as a “brain” for the network, and is most sought-after target for different types of attacks as it can be a single point of failure for the entire network if the attacks succeed. In the survey paper focusing specifically on the

control plane and interfaces that surround it, (Zaheed et al., 2023) discusses on the multitude of vulnerabilities that pose on the SDN controller. The authors give an exhaustive list of different attacks that target the northbound, southbound and the east-west controller interfaces, which all connect to the controller itself, and are beyond the scope of this study. However, the main takeaway from the paper is that DDoS attacks are the biggest concern when looking at the attack scenarios on the control plane. The attack taxonomy depicted in their paper shows DDoS attacks are mainly targeting the controller itself and the southbound interface. On the controller, DDoS attacks are executed by methods such as using packet-in-flooding, saturating the controller with application requests sent by exploiting the vulnerable or malicious apps as discussed in the application plane section, and flow table flooding.

Data plane: The data plane comprises of the hosts, and data forwarding elements that can perform SDN functions. The nodes in the data plane are just regular endpoint devices that are connected to each other via the SDN enabled forwarding devices that form the network. On the southbound API that is used for communication between SDN enabled devices and the controller, packet-in-flooding the switches and congestion of the API itself is done to execute DDoS attacks that target the network. DDoS attacks are also carried out by spoofing the SDN switches, or by flooding the switch buffer to saturate its memory.

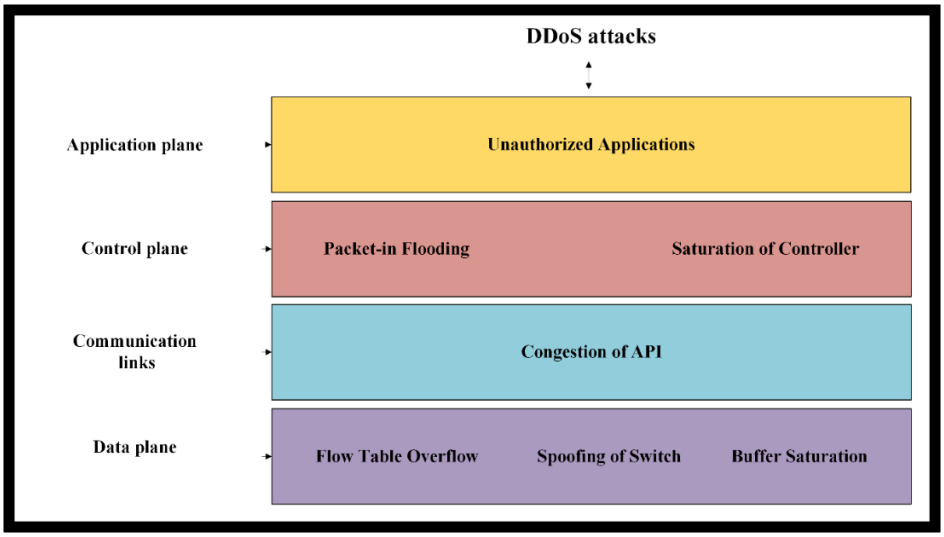


Figure 1: Security Issues in SDN architecture

Research on DDoS attacks in SDN based network designs is a very vast field as SDN offers the opportunity to program the software-based controller with a variety of detection mechanisms which are effective instantly once deployed. Multiple vendors have their own SDN solutions with their own security features available, however much of the research done on implementing different experimental solutions, and the DDoS attacks themselves of the SDN data plane network is conducted on simulations that use the OpenFlow protocol, which is one of the first southbound interface network protocols. It is used in OpenFlow enabled switches and works based on the instructions obtained from an OpenFlow enabled controller (Patel et al., 2023). The labs for simulating SDN networks and testing of DDoS attacks that are utilized for conducting research use these OpenFlow enabled simulators and controllers.

1.2) Motivation for the study

DDoS detection solutions in SDN that hve been implemented until the time of creation of this study are documented in the survey (Wang and Li, 2024) and is mainly divided into three categories; Information theory metrics, machine learning and deep learning techniques. Machine learning and deep learning techniques for detection are extensively researched upon at the moment and is not the focus of this project. Information theory techniques, mainly entropy-based detection solutions, work in real time by calculating statistical values based on anomalous behavior of different features of the OpenFlow flow table variables and detect said anomaly by comparing it to the value that is obtained during regular traffic simulation. Presently there are multiple variations of entropy-based techniques that work either in isolation or as part of a hybrid mechanism with machine or deep learning techniques. The original motivation of this study is to implement Shannon and Renyi entropy, two important information theory techniques, in order to detect a multi-vector DDoS attack scenario that targets a host on the SDN network. The novelty of the study originates from implementing both the entropy techniques together as well as testing of the proposed model with complex DDoS attack patterns that gives insight on the effectiveness of the mechanism.

1.3) Research Objective

It can be seen from the introduction of this thesis and the following literature review section that information theory techniques are being extensively looked into; particularly for detecting attacks in the data plane in a SDN setup. It can also be seen that certain aspects are overlooked within the present research which accredits the objectives that are the focus of this study that are listed below:

DDoS attack complexity based on packet rate manipulation.

Modifying existing Entropy/s techniques for threshold detections for overlooked complexity of DDoS attacks.

1.4) Research Question

Can an SDN controller be programmed to detect both high-rate and low-rate DDoS attacks by using different Entropy methods for a case of multi-vector DDoS attacks? How does the controller behave when the specified DDoS attacks are implemented additionally with burst attacks as in a multi-vector attack format that further resemble real time attacks?

Can the selected entropy-based detection mechanism perform its duty during different combination of these attacks during the trials?

2) Related works

The section gives a description of the papers studied for getting an understanding of the current research that has been conducted on entropy-based mechanisms for DDoS attacks, specifically targeting the control and data plane. The related work section is intended to clarify the basis of the decisions chosen for designing the trials conducted in this project as mentioned in the research objective, based on the prior work done for similar topics. The initial proposal of this project was inspired from DDoS detection solutions for a SDN

network, which was a very broad area of research. The survey papers that give an overview on security vulnerabilities and research done for SDN networks go in depth on the DDoS attacks of the different planes, which has been discussed upon in the introduction section for this study. The survey papers have listed multiple research papers on information theory detection solutions for different types of DDoS attacks on the SDN network, including Shannon and Renyi entropy. The basis of this project is derived from critically analyzing the work done in these research papers.

2.1) Focus on testing attack scenarios

(Sumantra and Gandhi, 2020) simulate a SDN network and execute three types of DDoS attacks, a TCP-SYN flood attack, a UDP flood attack and a slow HTTP attack. A TCP-SYN flood and UDP flood attack target the victim host in the network by saturating the data plane network link that the victim shares with the OpenFlow switch. The third attack they chose to simulate was a slow HTTP attack on one of their victim hosts that was simulated to function as a web server, which led to the server's CPU exhaustion. The detection solution proposed was based on Shannon Entropy, and the resulting mitigation times were given for each attack. Anomalies were detected based on the network features that change during the attacks, which are source IP addresses, TCP flags, Number of packets and Number of requests. The attack mitigation time measures the controller's time requirement for processing the entropy values and running their mitigation solution upon detecting an attack. The study did not implement a multi-vector attack scenario, which meant coupling of DDoS attacks together targeting multiple systems, which leaves room for experimentation.

A suitable outline for the experimentation for this research to be conducted is obtained from (Singh and Behal, 2021), where three entropy calculation techniques; Shannon, Renyi and Jensen-Renyi Divergence are used, and compares them for a high-rate and low-rate DDoS attack. However, no specifications were given on the type of DDoS attack, and there was no explanation on which features were particularly seen to have changes in its entropy values which led to the detection of the attacks. The attacks were also conducted separately, without the trials for a multi-stage attack scenario and comparisons were done based on a detection rating system of their own design, where it was compared to a dataset as well as a synthetic traffic dataset for analysis. The experiments concluded that JRD performed better in detecting both low and high-rate attacks, with a better detection and precision rate. Their study did not involve utilizing the entropy techniques in tandem to study on how a combined detection solution would perform when the attacks are executed simultaneously, and leaves room for future work.

The testing attack scenarios were explained with a breakdown of the detection mechanism used by authors (Asgari and Akbari, 2022) for their implementation of Shannon entropy to detect UDP, ICMP and TCP-SYN Flooding attacks, which gives an insight into the experimentation methodology that is used in most of the papers that conduct research on this topic. The detection mechanism they proposed involves two modules, a learning and a detection phase. The regular entropy values get determined during the learning phase of the simulation by calculating a threshold value under the assumption that the topology does not get affected. The detection phase compares the attack entropy values to the pre-calculated threshold. The mitigation is implemented by enforcing a hard or idle timeout on the switches, since the attacks used spoofed source IP addresses. Each of the attacks is carried out at

different intensity levels, and results declare the accuracy and the False Positive rates for each of them. The paper leaves room for further work in the form of extending the algorithm for detecting DDoS attack categories themselves, but overlooks the possibilities of multiple attacks targeting the network simultaneously.

In the papers by (Zhang et al., 2023) and (Shirsath et al., 2024), both discuss Renyi entropy to detect a DDoS attack of varying traffic levels in a simulated network. The papers discuss the α value that is used in Renyi entropy as a coefficient, and gives an understanding of its use in detecting more sensitive changes to the network variable values which is otherwise tougher to detect. (Zhang et al.) calculate the entropy values by only considering the source and destination IP addresses in order to detect a high-volume flooding DDoS attack which was not explained in depth. The attack was also conducted in different levels, at 25 and 75 percent, and confidence interval was calculated which was used as a tool for determining the false positive rate, which was found to be 0. (Shirsath et al.) goes into depth on the relation between Shannon and Renyi entropy, and proposes a multi-stage detection system coined as SYNTROPY, for TCP-SYN flooding attacks. A dynamic threshold entropy value mechanism is used for updating the values for a fine-tuned time window. The experimentation was carried out by deploying different Datasets in the Ryu controller and results were compared to that of another detection mechanism called SAFETY, another TCP-SYN flood detection mechanism based on Shannon entropy from (Kumar et al., 2018) to which they concluded with their proposed solution outperforming the former in metrics such as CPU load, Detection time and Confusion metrics.

2.2) Focus on comparing Entropy techniques

(Tseng et al., 2018) proposed a solution that can secure application deployment in SDN, where the controller was divided into two parts for performing both data plane and application plane functions. The application plane controller part is designed with an updated policy engine that runs on a sandbox outside the controller to provide a secured communication to defend against resource exhaustion attacks which is a type of DoS attack on the controller of the SDN network. The authors implemented this attack along with malicious command injections or API abuse attacks. DoS attack from a malicious app is not to be confused with application layer attacks in a general DDoS context as SDN applications use different APIs for running their apps.

(Swami et al., 2021) experiments with a statistical analysis method called a Z-score in order to detect spoofed TCP-SYN flood attacks on a SDN network emulation. The statistical measure is used to detect an attack and is used with two different tables of values, one for calculating the score by taking in the feature variables of the network and detecting spoofed IPs and other is used for storing the flagged IPs. The the authors have explained a unique mitigation scheme and the details on its steps. The paper proposes a model where both the detection and mitigation are compiled into a single component that is run on a controller, and results are measured based on the bandwidth usage, the CPU usage and response time after the blacklist is used to mitigate the attack by blocking the stored IPs.

(Valizadeh and Niar, 2022) implement a larger SDN network for their testing and uses Shannon entropy, along with another determining method that they proposed called PWI, in order to detect DDoS attacks that are carried out in multiple configurations. The attack is performed in repetition with differing attack scenarios, which are configured based on

increasing numbers of attacking hosts and victim hosts, and shows that the entropy detection solution fails to deliver when multiple victims are under attack, however their proposed solution were claimed to deliver results. Their solution is implemented in a separate controller, which receives the network traffic information and their proposed addition to the initial entropy detection finds success in detection when there are increasing number of victims. The authors have compared their technique only with detection based on Shannon entropy, along with only performing a single type of DDoS attack.

(Van et al., 2022) applies a threshold method that is dynamically updated during the traffic run in the simulation. The value is calculated by taking Shannon entropy values for a time window, then adding the average entropy and the standard deviation of the entropy for the given time window. The DDoS attacks are carried out in different rates, different volume of attack traffic, and the results were established according to the response times that were different for different levels of attack rates as well as coefficient values that were used for the standard deviation. The study offers an alternative approach to implementing Entropy along with statistical measurements, and focuses on the topic of early detection solutions for network attacks. The simulation is carried out with different attack rates at 25, 50 and 75 percent attack strengths, essentially comparing the threshold values at static assigning and their proposed dynamic assigning mechanism. The entropy detection solution does not consider for scenarios of multiple attack rates, and specifically low-rates of attack traffic. The dynamic threshold mechanism is not an entropy-based technique, but it is used for detecting low-rate attacks successfully.

A detection technique based on φ-entropy is discussed in the paper by (Li and Wu, 2020), which is another alternative proposed to enhance the sensitivity of the entropy value for the intensity of the DDoS attack simulated. The φ-entropy works by utilizing the hyperbolic sine function to adjust the entropy sensitivity, and is explained in depth in the paper. The comparison of the values obtained from this entropy function is compared to the Shannon entropy for a DDoS attack simulated at different levels of intensities, collaborated with different values of the tuning parameter of the function. The testing proved that higher intensities of attacks lead to stronger fluctuations in the entropy values. The idea of comparing different entropy values can be observed in this study, however the yielding values were not tested against different types of attacks, thereby limiting its use cases. This leaves room for implementing the entropy along with other information theory techniques; however the particular entropy method discussed in the paper is not the focus of this study.

2.3) Literature Review Summary

The different entropy-based detection solutions proposed in the reviewed studies conduct their testing with some level of packet-rate variations for displaying the metrics of their detection models and other factors, however it can be noticed that packet-rate is not considered as a definitive method for inducing complexities in attack scenarios, unlike the ones seen in abundance in real networks. The detection techniques used are varied in the studies where entropy techniques are utilized with different approaches in their proposed solutions. The solution proposed in this study tests the Shannon and Renyi entropy values in a novel mechanism for the relevant overlooked complexity of packet-rate based attack patterns.

The detection solution implemented in this study is tested on an extensive level of attack patterns based on this oversight with the intention to gather more insights on how the newly designed detection mechanism built on the basics of entropy behaves to this added challenge.

3) Research Methodology

Based on the previous section, it can be established further that information theory techniques are being sought after for developing logical solutions with reasonable computational requirements for defense against DDoS attacks in an SDN environment. It is also evident that a lack of attention is given to the complexities of DDoS attacks that can be simulated while testing out the proposed solutions. The research methodology is established based on the core procedure that were used for conducting tests in multiple papers that were studied.

The section provides an in-depth description of the components that will be used throughout the study, that is setting up the tools used in creating an OpenFlow enabled network topology, and the controller based upon the research papers reviewed as a test-bed for running SDN enabled applications in a controlled environment. The focus of the research problem is worked on using an SDN controller that will be connected to the network via the OpenFlow southbound API, which will be programmed to run the proposed detection logic. Once the detection component is implemented on the controller which can then be used to carry out multiple testing scenarios, the network can be populated with regular internet traffic in order to establish the entropy values that can be considered as threshold values used for the purpose of setting a limit. The simulated network will then be utilized for performing DDoS attacks, which will be used for testing the deployed detection mechanism, in increasing complexities that are seen in multi-vector DDoS attacks.

3.1) SDN network simulation

The network simulation for this project is done on Mininet, which is a free python-based software tool that allows the creation of virtual networks using virtual network links and virtualized server creation technology available on Linux platforms. A considerable number of the research papers utilize Mininet for testing out their experiments. This virtualization feature is called a network namespace which is used for providing a process that runs on the Linux kernel with a unique network interface and other network components such as routing or ARP tables, and are then connected using virtual ethernet pairs. The software is almost entirely developed on Python, with a little bit of C, and comes with a GUI tool called MiniEdit that can be used to visually design complex network topologies and export it as a Python script file.

3.2) SDN controller

SDN enabled controllers are programmable frameworks that can be used to develop virtual network functions in the form of “components” for different requirements. Mininet is pre-installed with a default OpenFlow controller but it does not provide sufficient functionality. Multiple open-source controllers such as OpenDaylight, ONOS, Ryu, Floodlight and POX are extensively used in enterprise network services and research. The most commonly used controllers based on the papers seemed to be either POX or Ryu. Excluding commercial focused controllers and focusing more on controllers that are used for conducting research, and by taking into consideration of the controller chosen by multiple research paper authors

that were referenced, I decided to use the POX controller for carrying out this study. In a study by (Sinha et al., 2023) four different open-source controllers were studied upon for the purpose of its behavior in response to DDoS attacks. A comparative testing of the open-source controllers listed above excluding ONOS, were subjected to DDoS attacks of multiple types were carried out to study the impact on their performance without implementing any detection mechanism. The studies concluded that the POX controller was negatively impacted by DDoS attacks that used spoofed source IP, MAC Addresses and destination IP addresses, which led to the selection of the type of attack to be simulated for effective testing of the proposed detection solution in this study. The controller behavior is gauged by the authors through the utilization of its CPU and memory bandwidth resources during the instance of a simulated attack.

The POX controller is a lightweight open-source framework that is entirely designed in Python, and has a good baseline collection of virtualized components which is intended to be used as starting grounds for developing custom solutions, making it ideal for the purpose of research. It is modified with the custom designed detection logic based on two entropy calculation techniques, Shannon and Renyi entropy, in order to detect fluctuations in the network feature variables.

3.3) Simulation of normal traffic

Once the lab is created for testing, the calculation of the different entropy values is to be carried out for regular simulated network traffic by using a custom python script that generates internet packets from random source IP addresses. The script is designed for simulating traffic within the Mininet network akin to an idle network with randomized packets being sent without any patterns.

3.4) Simulation of DDoS attacks and testing

The Mininet simulation is subjected to a high-rate UDP flood DDoS attack that targets hosts in the network. Further testing is then carried out by simulating another UDP flood attack, but as a low-rate attack which would be harder to detect, then creating multi-vector scenarios. The experimentation result is to be then recorded and analyzed in the later sections of this study, with the objective of achieving the goal of understanding the detection capabilities of the entropy techniques during a multi vector DDoS attack, thereby providing clarity in the research gaps discussed. The testing rounds are to be analyzed with a clear explanation of the results at the end of this study.

3.5) References to previous works

The papers studied for this project include a structure for the steps carried out for testing out each of their proposed solutions. The research methodology for this study can be explained using the traditional waterfall method for complete visualization.

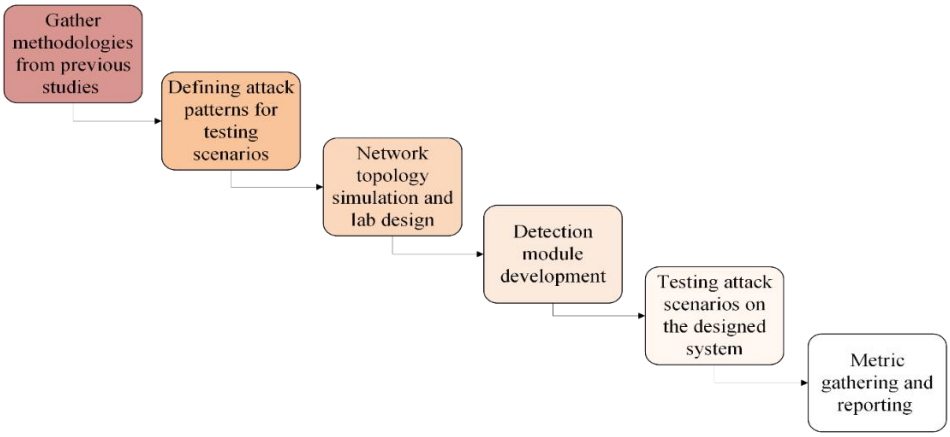


Figure 2: Waterfall Methodology

4) Design Specification

4.1) Designing the network topology

The network topology is designed using MiniEdit, and the script is submitted with the artefacts. The network topology can be connected to the POX controller by specifying the controller's network details in the topology script. This enables the controller to be implemented remotely on another system as well. However, it has been deployed in the same virtual machine for ease of use in the study. As seen (Figure 3: Network Topology), the controller is connected with two Open vSwitches that are connected to each other. Open vSwitches are functioning on the Linux VM kernel since the switch class is set to "OVSKernelSwitch" and utilizes its resources for better performance. There are 12 hosts connected in the network, 6 hosts to each switch. The network links are customized by utilizing "TCLink" (Traffic Control Link) a class in Mininet, to add some extent of realistic network features such as bandwidth, packet loss and delay values between host to host, host to switch and switch to switch. The simulated topology is usually ideal for research setups for testing and is commonly found in real life in small businesses and educational networks. The topology was chosen because of feasibility reasons of running the simulation on the system without compromising the resources, as it would be needed for simulating DDoS attacks.

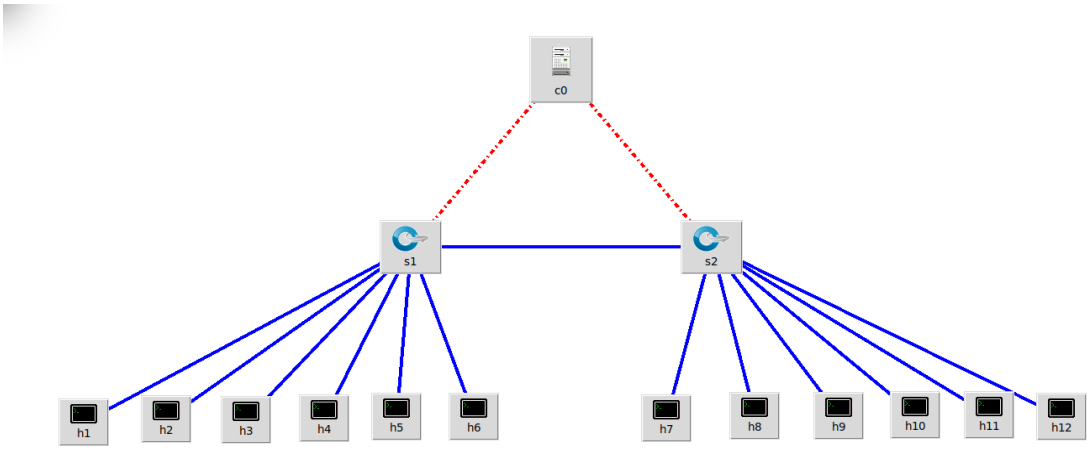


Figure 3: Network Topology

4.2) Detection mechanisms

a) Shannon Entropy

Entropy by its definition, is the measurement of uncertainty within a system. In information theory, entropy is utilized to understand the randomness within any variable, and is measured in bits which is binary. In the context of the detection mechanism, entropy is utilized for detecting changes in the probability distribution of the packets received by victim IP addresses during a DDoS attack, since it will fluctuate significantly. Two mathematically related information theory techniques are considered for the detection solution, Shannon and Renyi entropy. Shannon entropy was introduced in 1948 by Claude Shannon (Shannon, 1948), and defined the formula for calculating Entropy(H) for a discrete random variable(X) with outcomes for it (x_1, x_2, x_3, …x_n) and its probability distribution P(X) as-

$$H(X) = -\sum_{x=1}^{n} p(x_i) log_2 p(x_i)$$

In the context for detection, the random variable would be the destination IP addresses, and the probability distribution is calculated for the packets received for each of them. Higher packet counts for any IP would be recorded in the distribution, which leads to a change in the Entropy. Shannon entropy has been extensively proven to detect high volume DDoS attacks in a number of previous studies.

b) Renyi Entropy

Renyi entropy is a generalization of Shannon entropy, created by Alfred Renyi in 1961 (Renyi, 1961). The entropy measure introduces an additional parameter alpha(α), which is used to obtain a spectrum of entropy values, that give useful insights to different aspects of the probability distribution. The Shannon Entropy is included in the list of Renyi entropy values for alpha as one of the special cases, where α equals 1. The entropy is calculated by the formula for the same destination IP address distribution P(X) as-

$$H_\alpha(X) = \frac{1}{(1-\alpha)} log_2(\sum_{x=1}^{n} p(x_i)^{\propto})$$, where α is a finite positive real number.

The entropy values at different values of the order α gives us an insight into the properties of the distribution. This is possible as the parameter is used to define the entropy calculation's focus on different parts of the distribution itself. Therefore, Renyi entropy is used to find out traits about P(X) by tuning α. There are a few special cases of α which gives us distinctive properties of the distribution. For the detection module's context, this can be utilized to find out information about anomalous patterns of the packet distribution.

When $\alpha = 0$, the entropy is the number of the non-zero probabilities (Hartley entropy).

When $0 < \alpha < 1$, the entropy can be used to detect variables with low probability distributions in the collection. This is the main area of tuning that can may prove useful in detecting low-rate attacks, especially when there are multiple targets.

When $\alpha = 1$, the entropy converges to Shannon entropy due to approaching indeterminate fraction. After processing the limit of the equation as α reaches 1, we get the equation for Shannon entropy.

For $\alpha > 1$, Renyi entropy values are more sensitive towards the IP addresses with very high probability distributions.

When $\alpha = 2$, it magnifies the values of higher occurring probabilities as it squares the probabilities. In the study's context, this provides inadequate relevance.

4.3) Combined custom detection using Shannon entropy, Renyi Entropies

Entropy values are found to be less reliable when packet-rate decreases or number of targets increase. Renyi entropy at $\alpha = 1.5$ is more sensitive and deviates further when attack is targeted towards one or a few victims, and always deviates more than Shannon entropy. The first detection threshold is obtained as:

Ratio1 = Renyi entropy (1.5)/Shannon entropy

Renyi entropy at $\alpha = 0.5$ deviates more for low probability events, such as lower packet distributions to IP addresses with low activity. This can be utilized to detect attacks targeted at systems which don't receive as much traffic compared to the rest of the network, as well as attacks targeting multiple targets. The ratio can be defined as:

Ratio2 = Shannon entropy/Renyi entropy (0.5)

These ratios can be utilized to compare the deviation of the Entropy values for different attack patterns, and can be used to determine threshold values that would help in its detection. The ratios are calculated in this way to keep them between the range of 0 to 1, which makes it easier for detecting anomalies. The average of the ratios is calculated after collecting them three times for a current(c) estimation and any deviations in the entropy ratio is compared with the previous(p) calculated value, which gives us a difference value. This is done using the **deque** datatype in Python. The difference value is calculated, and then utilized as a flag.

The equations are:

Average ratio = $(\text{Entropy ratio})_1 + (\text{Entropy ratio})_2 + (\text{Entropy ratio})_3 / 3$

Difference = $(\text{Average ratio})_c - (\text{Average ratio})_p$ … if $c < p$

4.4) forwarding.l3_learning

The POX controller is available with a decent number of components to get started. The controller consists of a script called Forwarding.l3_learning, which is used for carrying out Layer 3 learning switch functions. The script is used for prototyping as it has the necessary key features of a Layer 3 network device, upon which additional custom functions can be tested. The switches in the topology send every new packet that they don't have instructions of, to the controller for processing by utilizing the "packet_in" function of the controller, which contains the detection mechanism written into the script itself. The learning phase of the algorithm can be visualized with the help of the flow-chart given (Figure 4)

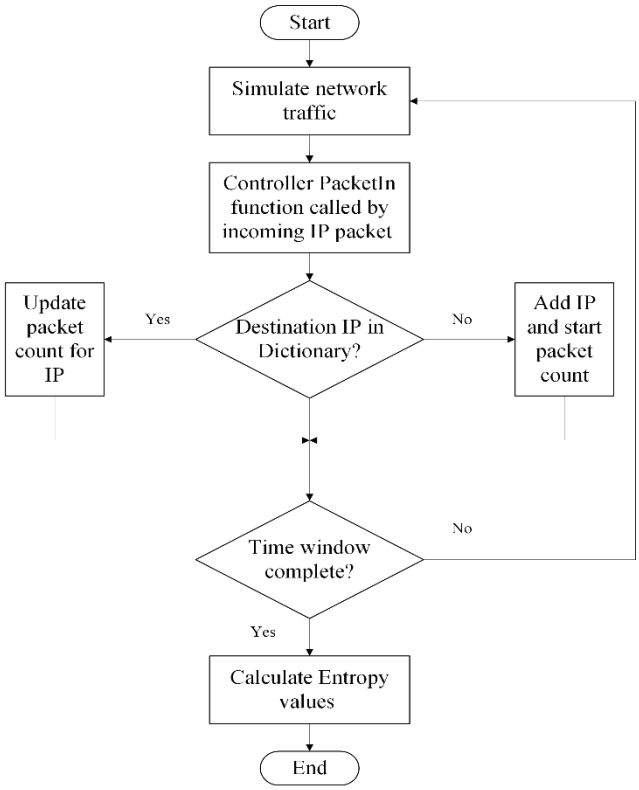


Figure 4: Learning phase flow-chart

A **defaultdict** variable is utilized by the PacketIn function of the POX controller to record incoming packets that are created from the simulated network traffic. This will help in keeping count of the number of packets arriving at each distinctive destination IP address. The learning process comprises the collection of this network information and executing the entropy calculation mechanism, which is the main part of the detection process. This is active during the entire duration of incoming network traffic. The entropy calculation is started at a periodic time interval during this learning phase, set at 20 seconds. The defaultdict variable is also updated with packets using OpenFlow stats request functionality, however this is used at a longer interval due to it's computational load on the controller.

4.5) Network traffic simulation tools and scripts

a) Normal Traffic

The SDN network is populated with normal UDP packets generated using a script that assigns randomized source and destination IP addresses within the designed hosts. This is done by utilizing Scapy and randrange. The normal traffic is active throughout the testing phase in the study, as it provides a base network entropy value to which malicious traffic is tested. The different attack patterns are then conducted alongside this traffic, which would test the detection mechanism.

b) Attack Traffic

A spoofed UDP flooding attack is conducted for attacks with a custom script that uses Scapy as well. This script is used to generate spoofed source IP DDoS attack traffic UDP packets at different packet rates and is also used for the purpose for multiple-victim targeting.

c) Additional attack

An additional attack is conducted along with spoofed UDP flooding using **burst traffic** in order to test the proposed multi-vector attack scenarios. The UDP flooding attack script is used to generate burst packets as well, creating rapid segmented attacks along with the flooding attacks carried out.

The detection phase of the algorithm is described in the following flow-chart (Figure 5). Once the time window is met, the entropy calculation is done on the data in the defaultdict variable. The probability distribution is calculated for the IP address and their respective packet counts. Three consecutive entropy values are then stored based on this probability distribution. The high-rate and low-rate averages are calculated, and their respective differences to their previous values are recorded as well. The thresholds are determined and compared to during the testing phase of this study.

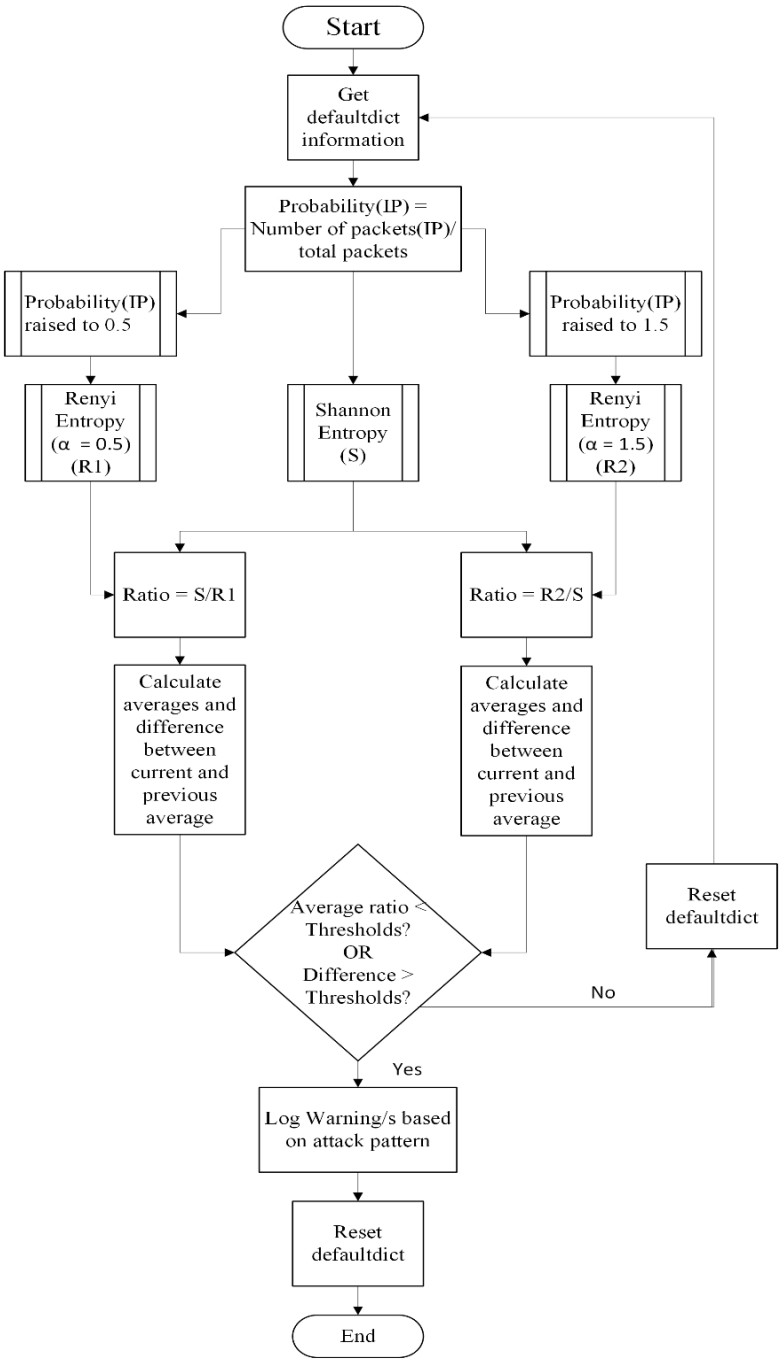


Figure 5: Entropy calculation flow-chart

The Implementation cycle used in the next section can be visualized with the flow-diagram:

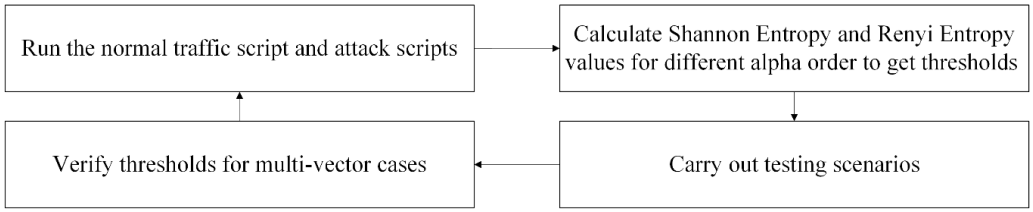


Figure 6: Implementation cycle

5) Implementation

The testing trials are carried out on a VirtualBox virtual machine (VM) running a Linux Ubuntu 20.04.6 LTS operating system. It has been assigned the following key resources:

CPU Cores: 2, RAM: 2 GB, VRAM: 16 MB, Dedicated Hard Disk Memory: 25 GB

The host machine is a Windows 11 Laptop which has the following key specifications:

CPU Cores: 10, RAM: 16 GB, VRAM: 7 GB (shared), Hard Disk Memory: 400 GB (SSD)

The VM is installed with Mininet 2.3.1b4, which comes prepacked with the POX SDN controller. The POX controller version was changed to 0.8.0 (halosaur), the latest one and to ensure Python 3 compatibility. The Mininet network topology, the custom POX controller script modified upon the component forwarding.l3_learning and the required network traffic scripts are all written in Python and shared in the artefacts subsection. Based on the network described in Figure 3: Network Topology, different attack scenarios are set up to test the proposed solution. Hosts from h1 to h4 are assumed to be infected and will be used to execute the attacks, with hosts from h7 to h12 as the targets. The topology script is started in the VM, and the xterm windows are used to open individual command line terminals for the relevant host machines in the network.

The testing environment offers flexibility for conducting the trials but the limitations of the resources were an unavoidable factor. The traffic generation involved in the testing induces overall lag in the host machine when kept running over time for generating the values, which highlights the clear distinction between the results obtained within the lab and in real SDN networks. The attacks were to be conducted therefore in moderated times and multiple trial-runs. Another notable difference is the complexities between generation of the base network traffic in comparison to real world networks, as real-time network traffic is much more random.

The normal network traffic is simulated within the network, and begins to generate traffic comprising of TCP, UDP and ICMP packets at equal probabilities of 1/3, to random destination IPs at a fixed packet traffic rate of 10 packets per second (pps). This is used to establish the base values of entropy which would be the highest values, as the randomness of the destination IP addresses, the types of packets and the packet rate are all chosen to establish maximum network traffic randomness. Real-time SDN networks have much more random traffic moving around and packet rates would be unpredictable, however the method executed for the study provides a decent starting point for satisfying the need of randomized traffic.

The attack scenarios are designed based on two key aspects; and are exercised to create distinct multi-vector attack scenarios as proposed for the study:

Packet rate: The high-rate flood is carried out at 100 pps, which is significantly higher compared to the regular traffic as stated earlier. This means that the low-rate flooding is carried out at 5 pps, lower than the defined regular traffic.

Number of targets: The attacks are tested for a minimum target of 1 host to a maximum of 4 hosts. This also plays a role in the calculation of entropy, and leaves room for testing the proposed detection mechanism.

The results are obtained from the calculated entropy values for different types of attacks, and their detection times. The breakdown of the metrics recorded is included in this section.

Three **entropy values** are calculated for a simulation of regular real-time traffic. Shannon entropy and Renyi entropy values at α 0.5 and 1.5 during this phase are at their highest due to the lack of probabilistic patterns that DDoS attacks would introduce to the distribution. The three entropy values averages at **3.3** to **3.4** during this phase.

The **entropy ratios** are based on the postulate of the relation between the collection of entropy values as theorized by Renyi, as stated in the earlier section. The ratios are used to stabilize the noise of the actual entropy values, and enable the threshold values for the different attack patterns be calculated within the range from 0 to 1. The averages of these ratios are calculated in order to detect the flooding attack type, to get information on their changes with each time window. The average values during normal simulation are recorded at **0.99** for both the values, as expected.

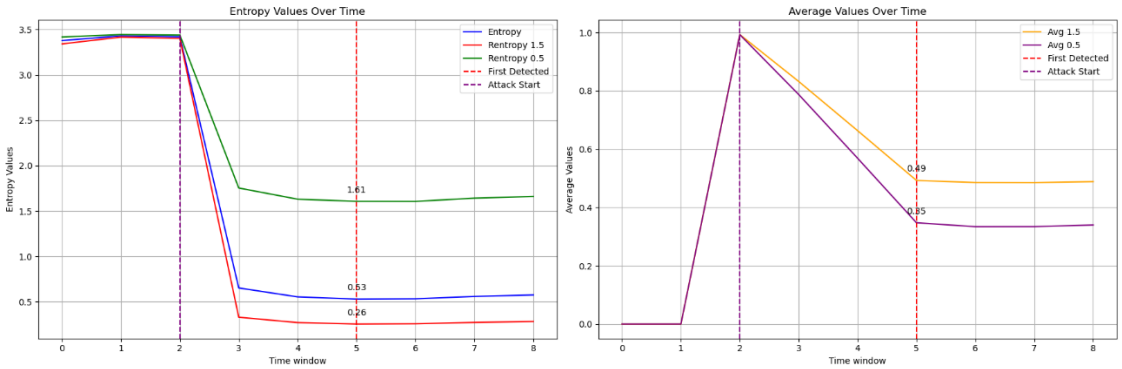
The **detection time** is calculated for each attack pattern, and gives insight on the effectiveness of the proposed entropy-based technique and the controller, in detecting the tested attack patterns and the speed of detection. The detection time was calculated by logging the attack script's execution start time and the time of detection by the controller, and subtracting the times. The results are visualized using Python in the form of graphs of the values to the time window for each attack pattern. One unit of the x-axis parameter on the graph represents the time window (TW) for the calculation, that is 20 seconds.

6) Results and Evaluation

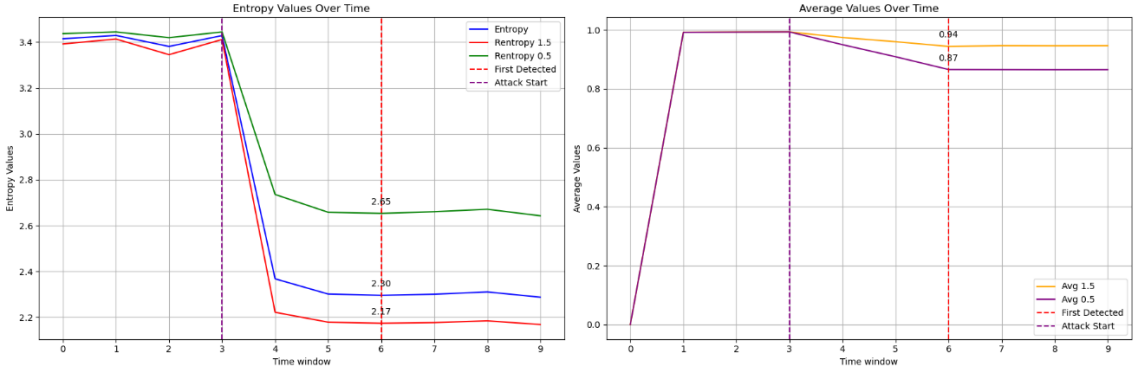
The detection times for each of the tested scenarios are given below:

Attack pattern (Test Scenarios)	Detection time	thresholds at detection(ratio1)	thresholds at detection(ratio2)
High rate/1 victim	59 seconds/ 3 TW	0.48	0.33
High rate/4 victims	56 seconds/ 2.8 TW	0.94	0.86
Low rate/1 victim	55 seconds/ 2.75 TW	0.66	0.64
Low rate/4 victims	50 seconds/2.5 TW	0.93	0.91
Multi-vector flooding at increasing levels	19s/22s/53s	0.91/0.86/0.91	0.90/0.79/0.86
Multi-vector flooding with burst attack	19s/20s/20s/54s	0.95/0.91/0.87/0.88	0.93/0.87/0.80/0.82

6.1) DDoS high-rate flood

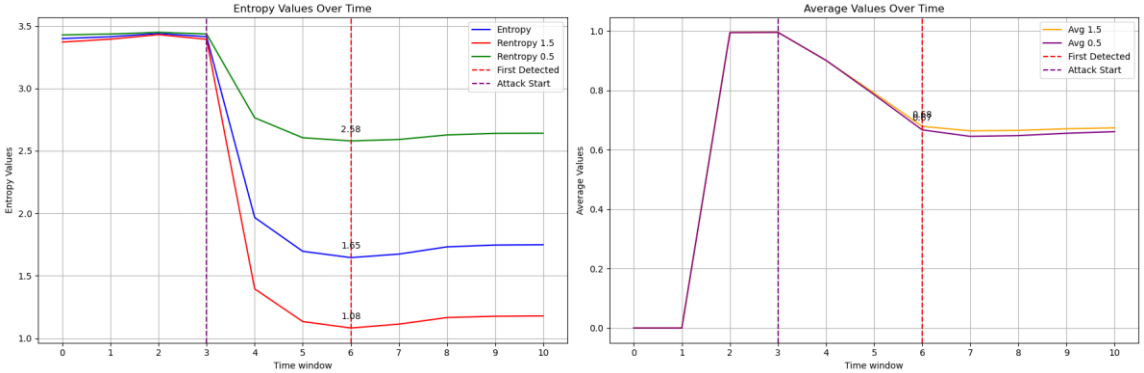


For one victim host: The trials are started by conducting high-rate flooding attacks targeting one victim. The entropy values are then recorded and used as thresholds for further testing. The UDP flooding is conducted from hosts h1, h2, h3 and h4 to one victim h7. The entropy values and the averages are both significantly lower, since the probability distribution skews due to flooded packets to the single destination IP of host h7. The threshold is set between the range (0.45, 0.50) for Ratio1 average and (0.30, 0.35) for Ratio2. The detection time for this attack was calculated to be at 59 seconds that is close to three-time windows.

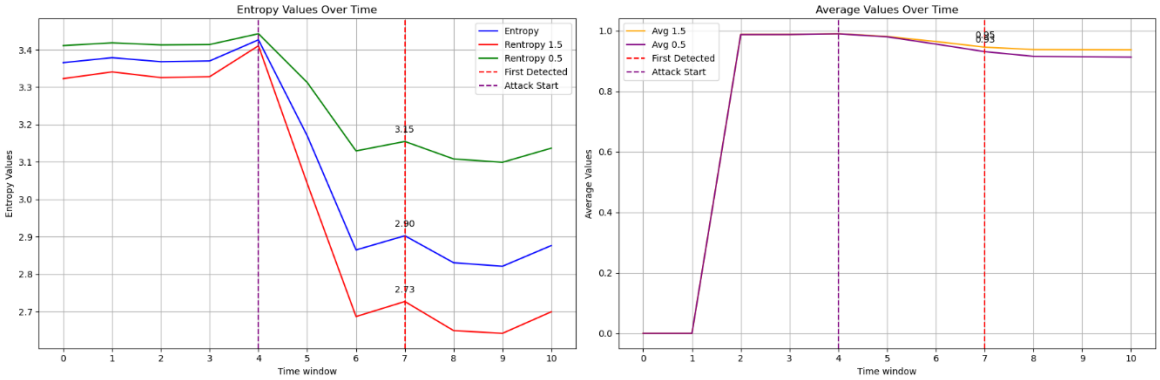


For multiple victims: The second scenario involves the same 4 attackers, this time targeting 4 different victims h7, h8, h9 and h10. The entropy values deviate less since the probability distribution is spread to more IP addresses. Ratio1 is set to (0.90, 0.95) and Ratio2 is set to (0.82, 0.87) for threshold. The detection time is calculated at 56 seconds.

6.2) DDoS low-rate flood

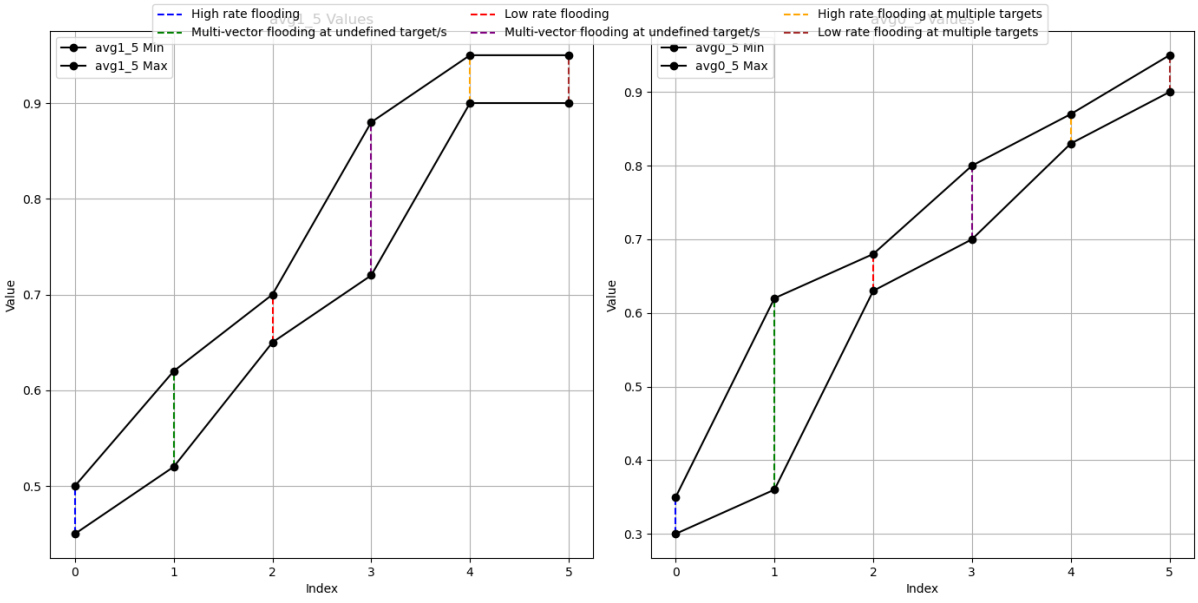


For one victim host: The attack is done similar to the high-rate flooding pattern to a single victim. The entropy values start to become varied due to lower packet counts. The Renyi entropy at α of 1.5 deviates more than the others as the single IP address is receiving all the low-rate attack packets. The averages are set at (0.65, 0.70) for Ratio1 and (0.63, 0.68) for Ratio2 as thresholds. The detection time for this pattern is calculated at 55 seconds.



For multiple victims: Similar to the high-rate flooding attack, low packet rates are transmitted this time to 4 hosts h7, h8, h9 and h10. The attack creates heavy fluctuation in the entropy values, with Renyi at α of 1.5 still deviating the most. However, the proposed averages are assigned values of (0.90, 0.95) to both Ratio1 and Ratio2 for this pattern. This attack is detected at 50 seconds.

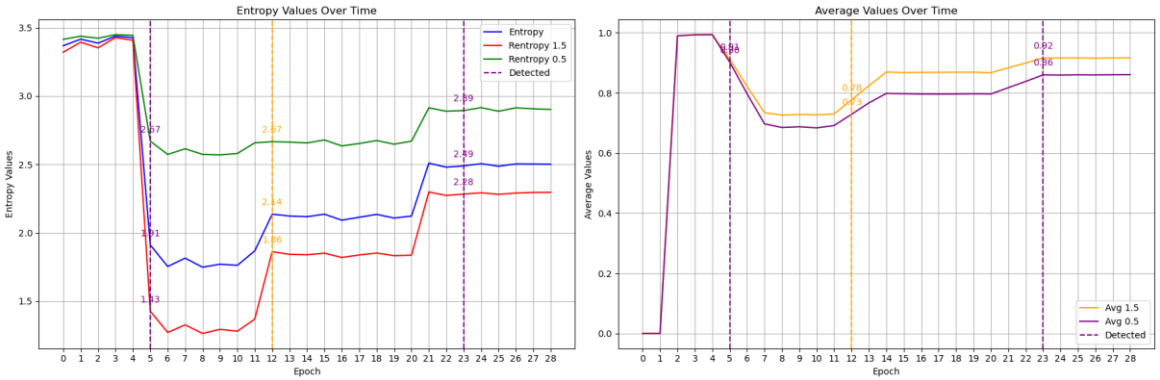
Based on the tested scenarios, different threshold ranges are assigned for the averages calculated. This information is now utilized to assume threshold ranges for the proposed multi-vector attacks, a combination of both the tested attack patterns. The threshold ranges for detecting this combined pattern can be visualized using the graph as shown.



As seen, high-rate and low-rate average values form an ascending order of values for both cases of single and multiple targets at different parts of the scale between 0 to 1. For the proposed scenarios in this study, the threshold ranges are set between the values recorded. The Ratio1 is set at (0.52, 0.62) and (0.72, 0.90) and Ratio2 is set to (0.36, 0.62) and (0.70, 0.90), which exist between the recorded thresholds.

6.3) Low and high-rate attacks

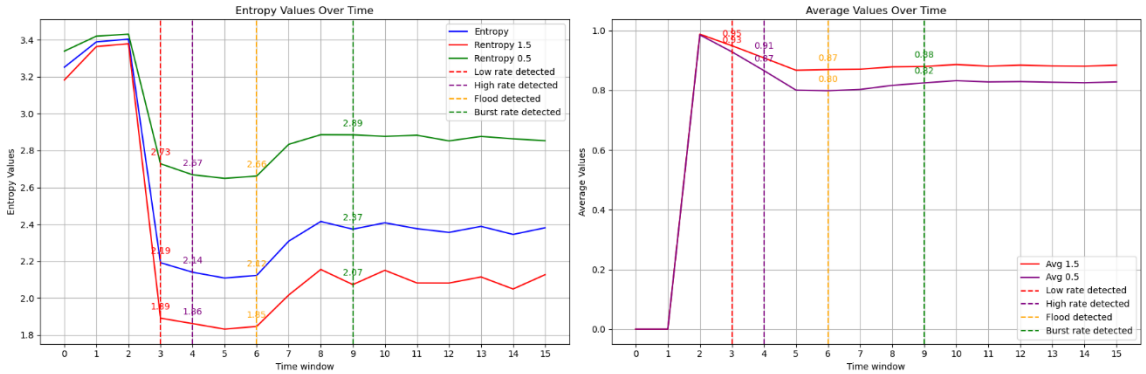
For testing out the threshold values, an incremental, combined pair of low and high-rate attacks are carried out on individual targets at gradual stages. The attack starts from hosts h1 and h2 initiating a low and high-rate attack respectively towards hosts h7 and h8. The next attack follows the same pattern, starting from h3 and h4 towards h9 and h10, followed by h5 and h6 targeting h11 and h12.



The first set of attacks triggered the low-rate thresholds between (0.90 and 0.95) for both ratios, however the averages were trending downwards as seen in the graph. The second attack was detected by the estimated multi-vector thresholds between (0.72, 0.88) for ratio1 and (0.70, 0.80) for ratio2. The third layer of the attack is detected as high-rate flooding by the thresholds assigned earlier, at (0.90, 0.95) for ratio1 and (0.83, 0.87) for ratio2. The detection time for the attacks were at 19 seconds due to the premature detection of the low-rate threshold at the start, followed by 22 seconds for the second layer, and the third layer of the attacks were detected at 53 seconds by the high-rate threshold.

6.4) Burst traffic attacks

The difference values calculated are used as flags for detecting whether the multi-vector attacks are involving burst traffic as well. The difference between the averages calculated over the time window will generate a difference if there are random packets arriving to the target IP during the flooding attack, irrespective of the time window. Since the addition of burst traffic can be considered in the multi-vector category, the same thresholds are applied, however with this added check for flag conditions in action. The differences calculated are used as a simple flag, if the values are zero, while the thresholds are met, the attack is assumed to be flooding, and burst traffic is involved if not.



The attack pattern is conducted by implementing 2 high-rate and 2 low-rate attacks on 4 separate hosts, and then conducting a burst attack on a 5th system from a different attacker. The low-rate threshold, being the most sensitive values and being closest to the normal entropy averages, gets triggered at the start followed by the high-rate threshold. The multi-vector attack gets flagged as flood at the third point of detection. The burst rate attack was not conducted until this point. After the flood-based detection is triggered, the 5th system was targeted with the burst-rate attack, which gets detected as burst-rate at the 4th point of detection. As seen, the entropy values become highly unreliable at this stage and heavy fluctuations are introduced, however the averages react better to some extent, and had successfully detected the variation in packet rate. The detection times are logged as follows; 19 seconds for first detection, 20 seconds for second detection after the first, 20 seconds for detecting the flood rate; that is the third detection. The burst rate was started after a certain period of detecting the flood attack, and the detection time was calculated at 54 seconds.

7) Discussion

The results of the testing scenarios have showcased the use of the proposed metrics, and their fluctuations(differences) in comparison to two entropy values from Renyi's entropy collection and Shannon entropy. The foundation of this concept arises from the mathematical rule of the chosen entropy values having the same magnitudes at maximum uncertainty within the probability distribution. The entropy values themselves were used for most of the research that was referred to for this study, with added statistical measurements and/or machine learning aspects that followed up on the initial information theory detection solution. The attacks conducted for this study covered mainly one key aspect of the features of DDoS attacks, that is packet rates. The number of types of DDoS attacks are vast and there are a multitude of complex patterns involving different tools, however the most common and large-scale DDoS attack types are still based on volume and number of packets. The SDN technologies used in present day networks are equipped with different solutions, however DDoS attacks still exist and affect these networks. Statistical and information theory-based solutions offer a real-time approach to deal with this problem, as well as considering the importance of computational resources during the attack, which is extensively researched upon at the moment as seen from the literature review.

The presented solution for detecting multi-vector attacks was built upon the original idea of implementing entropy techniques to detect more than a certain type of attack; instead focusing on a range of attack patterns, which are used to trick existing detection systems into assuming it as regular traffic. Entropy values successfully deviate based on the network information that it has been given, however the values themselves are unreliable due to the expected chaos that ensues in any network, ranging from network spikes to inactivity. The proposed solution uses the relation between the collective entropy values, and uses an average of them to obtain a more reliable variable. A difference calculating mechanism is proposed that successfully detected traffic fluctuations from burst attacks, which would be a much more difficult challenge if it was meant to be determined using only entropy values. The initial first 4 rounds of testing belong to a subcategory of its own, as in they are flood based volumetric attacks. These attacks work on the principle of damage by numbers, but it also makes it easy for the detection mechanism to do its job. When patterns change however,

the initial entropy detection system ceases to perform as well as it used to. The comparison between entropy values and average values are seen, and it can be seen that the more complex the execution pattern becomes, the more deviations are seen in the entropy values, compared to the averages of the ratios. The difference of the averages is only taken into consideration when the entropy values keep changing due to the burst rate. The proposed solution detects the magnitude of change of the averages in either direction, since the burst traffic can work both ways in either reducing or increasing the averages. This feature combined with the thresholds of different levels of packet-rate based attacks obtained previously gives us an opportunity to detect more complex attacks. The proposed solution is not intended to be scalable, and the initial regular traffic entropy values are subjective to change because of the limitations of the normal traffic generation used. However, the solution is proposed to be considered as a foundation to use the entropy values by combining the aspect of its flexibility in detecting variations in the probability distribution; a key feature in calculating the values themselves.

8) Conclusion and Future work

The proposal of this study was to implement information theory solutions for detecting DDoS attacks that are complex and challenging to detect. The research done for this study revealed different approaches of applying detection solutions to different aspects of an attack, which was placed as a core principle for developing the project implemented. The resulting project involved using packet-rate based attacks, which arguably creates the most challenging situation for the proposed solution - entropy values. The study implemented a novel solution by using the averages of related entropy values in Renyi's entropy collection, and then using the same values to detect another particular type of packet-rate based attack; burst traffic attacks. The project in this study can be considered as implemented in a vacuum of sorts, where there are minimal external factors that affect the network traffic; the simulation solution cannot be considered realistic. Despite this factor, an added effort was put into setting up the network topology with some realistic resembling network features using TCLink. The project in this study is also carried out with special concern given to not exhausting the host machine resources, as prolonged testing periods have led to the host machine crashing and shutting down. Despite these setbacks, the results were successfully produced from the solution implemented for the question that was proposed.

The main aspect when considering future works based on this study should involve setting up classifications for the entropy range to the respective types of attacks with extended testing. This testing can be carried out either manually, which was done in this study, or by using some form of automation solution. The different types of attack patterns can be assigned to different ranges of the proposed average values, and using the difference values to determine whether it falls into either flooding or burst attack category. Implementing machine learning to this solution would be the logical next step when it comes to further study. A critical survey needs to be done beforehand on the computational factors since this adds an added expense in the detection mechanism. The entropy calculation mechanism can be built for volumetric based attacks that cause a deviation in other network features as well. A fleshed out multi-purpose DDoS detection component can be built on top of the proposed method, which could detect other types of DDoS attacks.

9) References

Asgari, S. and Akbari, B. (2022). DDoS attack detection in OpenFlow based networks. *27th International Computer Conference, Computer Society of Iran (CSICC)*, Tehran, Iran, Islamic Republic of, 2022, pp.1–7. doi:https://doi.org/10.1109/CSICC55295.2022.9780508.

Bhuiyan, Z.A., Islam, S., Islam, M.M., Ullah, Naz, F. and Rahman, M.S. (2023). On the (in)Security of the control plane of SDN architecture: A survey. *IEEE Access*, 11, pp.91550–91582. doi:https://doi.org/10.1109/ACCESS.2023.3307467.

Camilo, J., Jenny Cuatindioy Imbachi and Felipe, J. (2020). Security in SDN: A comprehensive survey. *Journal of Network and Computer Applications*, [online] 159, p.102595. doi:https://doi.org/10.1016/j.jnca.2020.102595.

Kumar, P., Tripathi, M., Nehra, A., Conti, M. and Lal, C. (2018). SAFETY: Early detection and mitigation of TCP SYN flood utilizing entropy in SDN. *IEEE Transactions on Network and Service Management*, 15, pp.1545–1559. doi:https://doi.org/10.1109/TNSM.2018.2861741.

Jiménez, María B, Fernández, D., Rivadeneira, J.E., Bellido, L. and Cárdenas, A. (2021). A survey of the main security issues and solutions for the SDN architecture. *IEEE Access*, 9, pp.122016–122038. doi:https://doi.org/10.1109/ACCESS.2021.3109564.

Li, R. and Wu, B. (2020). Early detection of DDoS based on φ-entropy in SDN networks. *4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, Chongqing, China, 2020, pp.731–735. doi:https://doi.org/10.1109/ITNEC48623.2020.9084885.

Patel, R., Patel, P., Shah, P., Patel, B. and Garg, D. (2022). Software defined network (SDN) implementation with POX controller. *3rd International Conference on Smart Electronics and Communication (ICOSEC)*, Trichy, India, 2022, pp.65–70. doi:https://doi.org/10.1109/ICOSEC54921.2022.9952123.

Renyi, A. (1961). *ON MEASURES OF ENTROPY AND INFORMATION*. [online] pp.1–15. Available at: https://projecteuclid.org/ebook/Download?urlid=bsmsp/1200512181&isFullBook=false.

Shannon, C.E. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*, [online] 27(4), pp.623–656. doi:https://doi.org/10.1002/j.1538-7305.1948.tb00917.x.

Shirsath, V.A., Chandane, M.M., Lal, C. and Conti, M. (2024). SYNTROPY: TCP SYN DDoS attack detection for software defined network based on rényi entropy. *Computer Networks*, [online] 244, p.110327. doi:https://doi.org/10.1016/j.comnet.2024.110327.

Singh, J. and Behal, S. (2021). A novel approach for the detection of DDoS attacks in SDN using information theory metric. *2021 8th International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, India, 2021,pp.512–516.

Singh, J. and Behal, S. (2020). Detection and mitigation of DDoS attacks in SDN: A comprehensive review, research challenges and future directions. *Computer Science Review*, [online] 37, p.100279. doi: https://doi.org/10.1016/j.cosrev.2020.100279.

Sinha, M., Bera, P. and Satpathy, M. (2023). DDoS vulnerabilities analysis in SDN controllers: Understanding the attacking strategies. *International Conference on Wireless Communications Signal Processing and Networking (WiSPNET)*, Chennai, India, 2023, pp.1–5. doi:https://doi.org/10.1109/WiSPNET57748.2023.10134518.

Sumantra, I. and Gandhi, I. (2020). DDoS attack detection and mitigation in software defined networks. *International Conference on System, Computation, Automation and Networking (ICSCAN)*, Pondicherry, India, 2020, pp.1–5. doi:https://doi.org/10.1109/ICSCAN49426.2020.9262408.

Swami, R., Dave, M. and Ranga, V. (2021). Addressing spoofed DDoS attacks in software-defined networking. *6th International Conference for Convergence in Technology (I2CT)*, Maharashtra, India, 2021, pp.1–7. doi:https://doi.org/10.1109/I2CT51068.2021.9418202.

Tseng, Y., Nait-Abdesselam, F. and Khokhar, A. (2018). SENAD: Securing network application deployment in software defined networks. *IEEE International Conference on Communications (ICC)*, Kansas City, MO, USA, 2018, pp.1–6. doi:https://doi.org/10.1109/ICC.2018.8422405.

Valizadeh, P. and Taghinezhad-Niar, A. (2022). DDoS attacks detection in multi-controller based software defined network. *8th International Conference on Web Research (ICWR), Tehran, Iran, Islamic Republic of*, 2022, pp.34–39. doi:https://doi.org/10.1109/ICWR54782.2022.9786246.

Van, D., Huy, L.D., Truong, C.Q., Ninh, B.T. and Dinh, M. (2022). Applying dynamic threshold in SDN to detect DDoS attacks. *International Conference on Advanced Technologies for Communications (ATC), Ha Noi, Vietnam*, 2022, pp.344–349. doi:https://doi.org/10.1109/ATC55345.2022.9943031.

Wang, H. and Li, Y. (2024). Overview of DDoS attack detection in software-defined networks. *IEEE Access*, 12, pp.38351–38381. doi:https://doi.org/10.1109/ACCESS.2024.3375395.

Zhang, H., Zhou, L. and Lei, J. (2023). Renyi entropy-based DDoS attack detection in SDN-based networks. *IEEE 3rd International Conference on Electronic Technology, Communication and Information (ICETCI)*, Changchun, China, 2023, pp.334–337. doi:https://doi.org/10.1109/ICETCI57876.2023.10176631.