

**DDoS attack Detection Model based on Entropy Computing and Software Defined
Networking**

Mark Njore

138014

CNS

Supervisor Name

Dr. Victor Rop

**Submitted in Partial Fulfillment of the Requirements of the Bachelor of Science in
Computer Networks and Cybersecurity at the Strathmore University**

School of Computing and Engineering Science

Strathmore University

Nairobi, Kenya

May 2023

Declaration and Approval

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the research proposal contains no material previously published or written by another person except where due reference is made in the research proposal itself.

Student Name: Mwaura Mark Njore

Admission Number: 138014

Student Signature: _____ Date: _____

The Proposal of **Mwaura Mark Njore** has been reviewed and approved by **Dr. Victor Rop**

Supervisor Signature: _____ Date: _____

Acknowledgement

Even for a moment, I don't fool myself into believing that this work would have been possible without the help of others. That being said I hope I will be able to thank everyone who has helped me in the next few lines. First, I would like to thank my supervisor, Dr. Rop, for all support provided to me during this period. I have him to thank for guiding me through the process and helping with the few corrections made in my research. I would also like to thank our lecturer Mr. Tiberius for providing tips on how to work on the project and how to manage the project timewise. I would also like to express my gratitude towards my colleagues at the university whose inputs were also crucial for my research. Finally, I would like to thank my family for their support during this period.

Abstract

The increase in distributed denial of service (DDoS) attacks poses a significant threat to network infrastructures and necessitates the development of robust detection mechanisms. The purpose of this research proposal is to introduce a DDoS detection model that uses the concept of entropy for effective anomaly detection. Entropy, as a measure of randomness and uncertainty, can capture irregular patterns and anomalies in network traffic, enabling early detection of DDoS attacks. This model also includes aspects of SDN as a controller is used instead of a router. This model involves collecting network traffic data and extracting entropy-based characteristics to capture the underlying characteristics of the traffic, resulting in accurate and efficient detection of DDoS attacks.

The outcome of this research is a model capable of DDoS attack detection that improves network security by proactively identifying DDoS attacks using entropy computing and Software Defined Networking. The model was developed using the agile methodology and included thorough testing to ensure proper function. The model aims to improve detection accuracy and thereby minimize the impact of DDoS attacks on network availability and performance by utilizing entropy-based features. The research findings should help advance the network security field and provide valuable insight into the effectiveness of entropy-based approaches to DDoS detection. Ultimately, the proposed model could be deployed to strengthen security measures and defend against the growing threat landscape of DDoS attacks. This can improve network availability, reduce downtime, and improve overall network security in the face of evolving DDoS threats.

Table of Contents

Declaration and Approval	ii
Acknowledgement	iii
Abstract	iv
Table of Contents	v
List of Figures	ix
List of Tables	x
List of Equations	xi
List of Abbreviations	xii
Chapter 1: Introduction	1
1.1 Background	1
1.2 Problem Statement	2
1.3 Objectives	3
1.3.1 General Objective	3
1.3.2 Specific Objectives	3
1.4 Research Questions	3
1.5 Justification	4
1.6 Scope	4
1.7 Limitations and Delimitations	4
Chapter 2: Literature Review	6
2.1 Introduction	6
2.2 Types of DDoS attacks	6
2.3 Impacts of DDoS attacks	8
2.4 Review of existing systems	10
2.4.1 Machine Learning Techniques	10
2.4.2 Signature-based detection systems	10

2.4.3 Anomaly-based detection systems.....	11
2.5 Existing gaps	11
2.6 Conceptual Framework	12
Chapter 3: Methodology	15
3.1 Introduction	15
3.2 Research approach.....	15
3.3 Methodology	15
3.3.1 Plan	16
3.3.2 Design.....	16
3.3.3 Development.....	17
3.3.4 Testing	17
3.3.5 Deployment	17
3.3.6 Review	17
3.3.7 Launch	17
3.4 Deliverables.....	17
3.5 Tools and Techniques.....	18
3.5.1 VirtualBox	18
3.5.2 Mininet	18
3.5.3 Python programming language.....	18
3.5.4 Scapy	18
Chapter 4: System Analysis and Design.....	19
4.1 Introduction	19
4.2 System Requirements.....	19
4.2.1 Functional Requirements.....	19
4.2.2 Non-Functional Requirements.....	19
4.3 System Analysis Diagrams.....	20

4.3.1 Use Case diagram	20
4.3.2 Sequence diagram.....	21
4.3.3 Flowchart.....	22
4.3.4 State Transition diagram.....	23
4.3.5 Systems Architecture diagram.....	24
4.3.6 Network Topology.....	25
Chapter 5: Implementation and Testing.....	26
5.1 Introduction	26
5.2 Discussion of Implementation Environment.....	26
5.2.1 Hardware Specifications.....	26
5.2.2 Software Specifications	27
5.3 Discussion of Testing Environment	28
5.3.1 Functional Requirements Testing.....	28
5.3.2 Non-Functional Requirements Testing.....	29
Chapter 6: Conclusions and Recommendations	31
6.1 Conclusions	31
6.2 Recommendations	31
6.3 Future Works.....	32
Bibliography	33
Appendices:.....	36
Appendix 1: Gantt Chart	36
Appendix 2: Scapy installation screenshots	37
Appendix 3: Pip installation screenshots	38
Appendix 4: Mininet installation screenshots	39
Appendix 5: POX installation screenshots.....	40
Appendix 6: Vim installation screenshots.....	41

Appendix 7: XTerm installation screenshots	42
Appendix 8: Functional Requirement 1 testing screenshot.....	42
Appendix 9: Functional Requirement 2 testing screenshot.....	43
Appendix 10: Functional Requirement 3 testing screenshot.....	43
Appendix 11: Functional Requirement 4 testing screenshot.....	44

List of Figures

Figure 2.1 A DDoS attack.....	6
Figure 2.2 DDoS attack Classification.....	8
Figure 2.3 Conceptual framework	13
Figure 3.1 Agile methodology overview	16
Figure 4.1 Use Case diagram	20
Figure 4.2 Sequence diagram.....	21
Figure 4.3 Flowchart	22
Figure 4.4 State Transition diagram.....	23
Figure 4.5 Systems Architecture diagram.....	24
Figure 4.6 The Network Topology	25

List of Tables

Table 5.1	Table of Hardware specifications.....	26
Table 5.2	Table of Functional Requirements Testing.....	29
Table 5.3	Table of Non-Functional Requirements Testing.....	30

List of Equations

Equation 2.1 The mathematical equation for Entropy calculation	13
--	----

List of Abbreviations

CPU – Central Processing Unit

DDoS – Distributed Denial of Service

DNS - Domain Name System

DoS – Denial of Service

HTTP - Hypertext Transfer Protocol

ICMP - Internet Control Message Protocol

ML – Machine Learning

SDN – Software Defined Networking

TCP - Transmission Control Protocol

UDP - User Datagram Protocol

VM – Virtual Machine

Chapter 1: Introduction

1.1 Background

In today's world, the internet is increasingly becoming an essential aspect of our day-to-day lives. As of April 2023, there were 5.8 billion active Internet users worldwide (Petrosyan, 2023). This accounted for 62% of the world's total population. The number of users is still increasing as there are approximately 27,000 new Internet users every hour (Flynn, 2023). This growth represents not only technological progress but also an increase in opportunities for cybercriminals to take advantage of. This creates the challenge of enforcing security over the internet due to the cyber-attacks that exploit these opportunities.

Cyber-attacks are becoming more complex by the day, with new methods being discovered so frequently that it is becoming a problem to keep up with them. Among these cyber-attacks is a type called a DDoS attack. A Denial-of-Service (DoS) attack by definition is an explicit attempt by attackers to overwhelm a computer target's ability to handle incoming communications, prohibiting legitimate users from accessing the service (Douligeris & Mitrokotsa, 2004; Whitman & Mattord, 2017). A Distributed Denial-of-Service (DDoS) attack is a variation of this whereby the attacker coordinates an attack by using multiple devices at the same time, instead of one machine like in a normal DoS attack. This can be done using bots in a botnet or using zombies (Hoque et al., 2015). The attacker can attempt to flood a network, preventing legitimate network traffic, disrupting connections between two machines, preventing access to a service, or even attempting to prevent a particular individual from accessing a service (Weiler, 2002). These would all be considered DoS attacks as the accessibility of the system or information is affected.

DDoS attacks have been prevalent since as early as the 1990s. In September 1996, an attack on an ISP known as Panix became the first documented case of a DoS attack. The attack was directed at the devices on the networks including mail, news, name, and web servers (Patrikakis et al., 2019). DDoS attacks are becoming more and more frequent. The world has been seeing an increase in DDoS attacks year over year, with there being a 150 percent increase between Q2 of 2021 and Q2 of 2022 and it's only expected to get worse (Turner, 2022). Another example occurred in February 2020, when Amazon Web Services (AWS), a well-known cloud computing service platform saw as many as 2.3 Terabits per second coming into its servers as

a result of the DDoS attack (Nicholson, 2020). This attack surpassed the previous record held by the 2018 GitHub attack at 1.35 Tbps and is considered one of the worst attacks seen to date (Kottler, 2018).

DDoS attacks can last for multiple days or even weeks depending on the goal of the attack. An attack can last for several hours, days, or even weeks. In 2015, a DDoS attack on GitHub lasted for a total of 118 hours before it ended (Gheorge, 2015). The effects and consequences to the target host or the target network could then be seen even after the attack ends. One effect is that the system under attack would be unable to access the system or part of the system as a result of the attack. The system would also suffer from a lack of available bandwidth. The DDoS traffic would take up most, if not all of the bandwidth which would then lead to wastage. The resources on the network would also suffer from strain during the attack. Resources such as CPU memory would be used up in processing the traffic from the attack and would not be able to serve the system properly. The impact of these attacks depends on the severity of the attack especially when the target of the attack is an organization and not a single individual.

It is then crucial to detect these attacks when they are happening before it is too late to minimize the damage inflicted.

1.2 Problem Statement

The problem addressed by this research project is the lack of a comprehensive and adaptive DDoS detection system that can accurately detect DDoS attacks in real time. Current detection systems often suffer from high false-positive rates, limited scalability, and insufficient responsiveness to emerging attack patterns. Therefore, there is a need for a DDoS detection system that can overcome these limitations and provide reliable protection for individuals and organizations.

The current state-of-the-art DDoS detection systems face several challenges. Firstly, many existing detection mechanisms suffer from a high false-positive rate, leading to unnecessary disruption of legitimate network traffic. This can result in unnecessary service interruptions and impact the overall user experience. False positives also lead to wastage as network resources are used to investigate the traffic. Secondly, the scalability of detection systems becomes a concern when faced with large-scale attacks. The ability to handle and process vast

amounts of network traffic in real-time is crucial for the timely detection of DDoS attacks. Lastly, as attack techniques evolve, detection systems must be adaptive and capable of identifying new and emerging attack patterns. This necessitates the use of advanced techniques such as machine learning to enhance the system's ability to recognize unknown attacks. Addressing these challenges and developing a DDoS detection model is the primary focus of this research project.

1.3 Objectives

1.3.1 General Objective

The general objective is to develop a model capable of detecting DDoS attacks using entropy computing.

1.3.2 Specific Objectives

- i. To conduct further research on DDoS attacks
- ii. To assess the challenges brought about by DDoS attacks
- iii. To design and develop the proposed DDoS attack detection model
- iv. To test and validate the DDoS attack detection model

1.4 Research Questions

- i. What are the types of DDoS attacks?
- ii. What are the challenges brought about by DDoS attacks?
- iii. How can a DDoS detection model be designed and developed?
- iv. How can the DDoS model be tested and validated?

1.5 Justification

The proposed project is essential due to the significant threat that is posed today by DDoS attacks. By developing an entropy-based DDoS detection model, the timely detection of DDoS attacks would result in the effects of the attack being minimized greatly. The resources that would have been targeted would have uninterrupted access and their integrity would also be preserved. The losses that would have been incurred by organizations as a result of DDoS attacks would be avoided almost completely, whether financial or reputational.

This project is important as it could assist organizations and network administrators in safeguarding their systems against DDoS attacks. This could directly benefit organizations where the availability of resources is a key part of their business, for example, online banking, e-commerce, streaming services, and more. Therefore, the project is addressing a pressing need for improved DDoS detection capabilities. The findings and research from this project could also potentially improve the field of cybersecurity and provide material for future research.

1.6 Scope

The scope of this project will cover the design, development, and evaluation of a DDoS detection model based on entropy computing. It will focus on detecting different types of DDoS attacks. The project will cover the software implementation of the model. The scope of this project does not include the mitigation of DDoS attacks, as the focus is mainly on detection. Various performance metrics, such as detection accuracy and response time, will be considered to evaluate how effective the detection system is. The evaluation will involve comparing the proposed model with existing DDoS detection techniques to assess its performance. The scope also does not cover the detection of DDoS attacks using other methods such as machine learning. The scope however does not touch on the deployment of the system in a live environment or the evaluation of its performance under high-volume DDoS attacks. The focus will instead be on evaluating the model in a simulated environment.

1.7 Limitations and Delimitations

While this research project aims to develop a DDoS detection model, it is essential to acknowledge certain limitations that may impact the scope and capabilities of the proposed

model. One limitation is time. Due to the project being a one-semester project, it had a specified timeframe as it was to be completed in a few months. For this reason, the project scope is focusing on the detection of DDoS attacks and not its mitigation or prevention. Another limitation is the lack of access to real-life network environments. With this access, the project could have involved real-life testing of the model for evaluation. The alternative chosen was to run the test in a simulated/virtual environment.

Among the limitations was the lack of high-performance computing resources. The device used for this project could only handle a certain number of components on a network and could not be used for evaluation of the model on a large system architecture or a large-scale DDoS attack. For this reason, the project is focusing on detecting attacks on networks that are not very large in size. Another limitation was a limited skill set. The team working on the proposed project does not involve expert programmers or software developers in its personnel and as such does not involve high-level programming. Due to this, the project complexity will not be as high and will only utilize relatively simpler tools and programming languages.

Chapter 2: Literature Review

2.1 Introduction

This section provides a comprehensive overview of the existing literature on Distributed Denial of Service (DDoS) attacks and detection systems. Its purpose is to examine the current state of research and industry practice regarding DDoS attacks and their detection by reviewing a range of research papers, academic articles, and industry reports that focus on this topic. The literature review should serve as a foundation for understanding challenges and advances in DDoS detection and guides the development of the proposed DDoS detection model.

2.2 Types of DDoS attacks

A DDoS attack as stated before, is an explicit attempt by attackers, to use multiple devices at the same time to overwhelm a computer target's ability to handle incoming communications, prohibiting legitimate users from accessing the service (Douligeris & Mitrokotsa, 2004; Whitman & Mattord, 2017). These attacks are a common problem in today's world of internet security. According to Kaspersky, an average of 824 attacks per day were detected in the month of August 2022 by its DDoS Intelligence system (Kupreev et al., 2022). In this section, we will look further into DDoS attacks to get a deeper understanding of the problem.

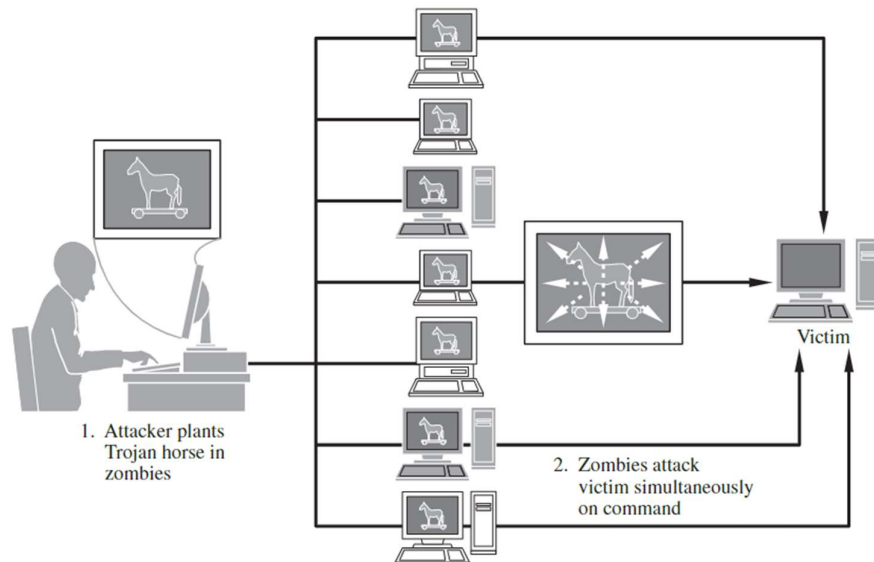


Figure 2.1 A DDoS attack (Pfleeeger et al., 2015)

A DDoS attack can be broken down into 4 stages, namely agent selection, compromise, communication, and execution. In the first stage, the attacker goes through the process of selecting. In the first stage, the attacker must find a way to take ownership of various devices. Devices could include PCs, mobile phones, or even IoT devices such as smart TVs. There are many ways hackers can discover these devices and take responsibility for them. This then leads to the next step, compromise. This step is the reason why the attacker specifically crawls the internet to find devices with known vulnerabilities. The attacker will have to find a way to compromise the devices and make them 'slave' devices. This can be done in various ways such as phishing.

The third step, communication is where the attacker establishes a means of communication with the compromised device. This is done so that the device can comply with any directions the programmer sends to the gadget at any given time. This can be done by installing a small program on the compromised device after compromising it. The final step, execution is where the attack takes place. Once the hacker has established a huge number of gadgets under his control, he/she can execute the DDoS attack.

There are a lot of different types of DDoS attacks that can be carried out today and a wide range of classifications have been proposed in the literature, over the past years. The classifications are based on the different features of DDoS attacks such as the architectural model, exploited vulnerability, weakness exploited, protocol level, degree of automation, attack rate, impact, scanning strategy, propagation strategy, and resources involved (Gupta et al., 2009). Among these, the protocol level is mostly used in the classification of these attacks since it is extremely simple to group DDoS attacks (De Donno et al., 2018). Basing the attack on the protocol level used, it is possible to separate the attacks into two categories: Application Level attacks and Network Level attacks (Alomari et al., 2012; Zargar et al., 2013).

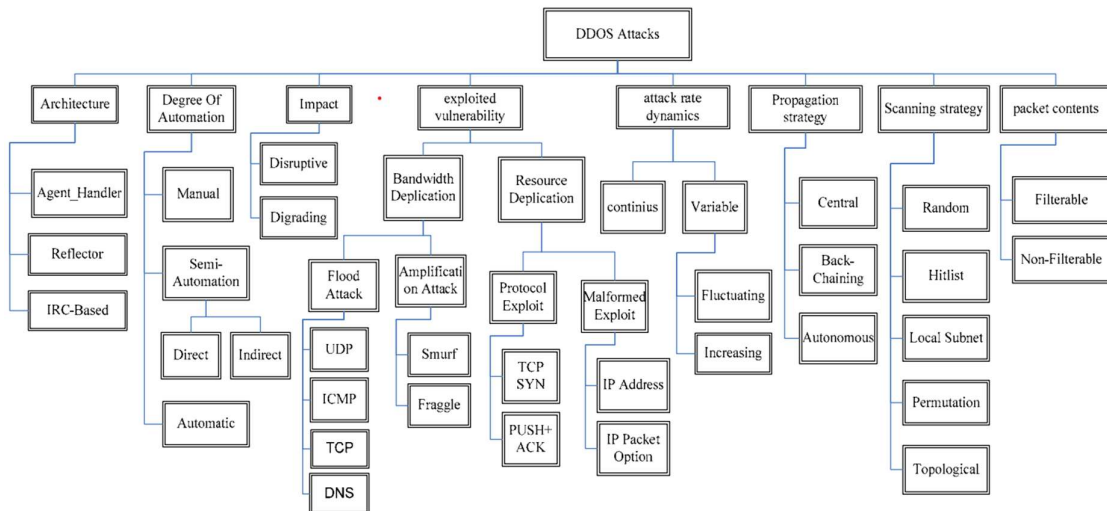


Figure 2.2 DDoS attack Classification (Asosheh & Ivaki, 2008)

The first type, Application-Level attacks, target application-level protocols to exhaust the system resources. Examples of these attacks are DNS Amplification attacks, DNS Flood attacks, and HTTP Flood attacks. Network Level attacks on the other hand target either network or transport layer protocols to perform the attack. Examples of this type of attack are UDP Flood attacks, ICMP Flood attacks, PUSH and ACK attacks, and TCP SYN attacks (Bishop, 2006).

2.3 Impacts of DDoS attacks

DDoS attacks have emerged as a significant threat to individuals, organizations, and even entire industries. The aim is to explore the impacts of DDoS attacks, highlighting the consequences they have on targeted entities.

One of the most straightforward and visible effects of a DDoS attack is the disruption of online services. By overloading a target system or network with traffic, an attacker can exhaust available resources and prevent the system from responding to legitimate requests. This results in service unavailability, downtime, and lost productivity. For businesses and organizations that rely heavily on online operations, such as e-commerce platforms, financial institutions, and government agencies, service interruptions can lead to significant financial losses and customer dissatisfaction.

DDoS attacks can also have a severe monetary impact on both individuals and businesses. Individuals are rarely affected but there have been cases where criminal groups have threatened their victims with a DDoS attack unless they paid 5 bitcoins which at the time was more than \$5,000 (Makrushin, 2017). Organizations targeted by DDoS attacks often experience revenue loss because they are unable to generate sales, fulfil orders or provide services during the attack. This is especially harmful to e-commerce platforms in particular, as every minute of downtime can mean significant financial losses. For instance, the security of Bitfinex, a cryptocurrency trading company, was breached and \$72 million in Bitcoins were stolen on 2nd August 2016. This resulted in all trading being halted for one week, hence, no trades were observed on the exchange during this period (Baldwin, 2016). Another example of this effect is the attack on the International Netherlands Group which resulted in a significant negative change in company stock price (van den Dool, 2013). Additionally, organizations may incur additional costs associated with incident response, mitigation services, and system repairs and updates to better defend against future attacks. These financial stresses can have long-term effects on an organization's financial stability and growth.

Reputation is another asset that can be affected by a DDoS attack. DDoS attacks can cause severe reputational damage to the targeted companies. Customers experiencing service interruptions or extended downtime can lose confidence in the company and its ability to protect their data and provide reliable services. This loss of trust can result in customer dissatisfaction, loss of market share and long-term damage to the brand's reputation. Research done by Abhishta (2019) shows that there are significant changes in customer behaviour in the wake of a large, successful DDoS attack on a provider whose business model includes protecting customers against such attacks. Furthermore, these changes in behaviour are not just temporary, but lasting changes in customer behaviour and permanent loss of customers were observed. Negative publicity about a successful DDoS attack can also attract media attention, adding to the reputational damage and damaging a company's image.

It can be seen that DDoS attacks have extensive effects beyond immediate service disruption. Commercial loss from attacks, together with reputational damage, can significantly weaken businesses and organizations. It is then imperative for organizations to recognize the severity of the impact of DDoS attacks and implement security measures to mitigate these risks. By

protecting against DDoS attacks, businesses can protect their online presence, maintain customer trust and ensure operational stability and resilience in a digital environment.

2.4 Review of existing systems

2.4.1 Machine Learning Techniques

Machine learning is the field of computer science that focuses on providing computers with the ability to solve problems through learning, such as in humans (Mitchell, 1997). It is a subset of artificial intelligence where computer algorithms are used to learn from data. The Machine learning model is trained using historical data. The data is then analysed by a learning algorithm, which generates a reasoning function that could be used to map new inputs that have not been seen before. ML techniques can be categorized by their purpose into supervised, unsupervised, semi-supervised, reinforcement and deep learning (Sarker, 2021).

One advantage of this technique is that they identify patterns indicative of a DDoS attack with high accuracy. ML models trained on tagged datasets can learn to distinguish between normal and malicious traffic, reducing false positives and false positives which results in higher accuracy. According to Idhammad et al., (2018), various experiments were performed using three public datasets namely NSL-KDD, UNB ISCX 12 and UNSW-NB15. An accuracy of 98.23%, 99.88% and 93.71% and false positive rates of 0.33%, 0.35% and 0.46% were achieved respectively. Another advantage is it can detect attacks in real time. ML algorithms can detect anomalies and suspicious patterns, trigger proactive responses and stop attacks before they disrupt any service. Research done by Bhayo et al. (2023) the HADEC framework takes less than 5 min to process 1 GB of a log file having 15.83 GB of generated live traffic.

2.4.2 Signature-based detection systems

These systems deploy signature or pattern detection and store the signatures of known attacks in a database. Each communication is monitored and compared with the database entries to discover any occurrences of DDoS attacks. The database is updated with new attack signatures occasionally to detect new types of attacks. Snort and Bro are examples of systems that use signature-based attack detection.

The advantage of this technique is that they are very effective in detecting well-known and documented attacks (Chandola et al., 2009). By matching network traffic against the signatures in the database, this method can accurately identify specific attack types. Another reason why these systems are used is that they produce low false positive rates (Mirkovic & Reiher, 2004). By using predefined patterns, these methods effectively distinguish malicious traffic from normal network traffic, minimizing the possibility of misidentifying legitimate user activity as an attack.

2.4.3 Anomaly-based detection systems

Systems that deploy anomaly-based detection have a model of normal system behaviour, such as normal traffic levels or the expected system performance. The current state of the system is periodically compared with the models to detect any differences or anomalies.

According to Zekri et al., (2017), the main advantage of these systems is that they excel at detecting unknown and zero-day attacks. By identifying deviations from expected patterns, detection of even the most novel and sophisticated DDoS attacks is possible, providing a proactive defence mechanism against evolving threats. Another benefit is that anomaly-based techniques tend to have low false positive rates. These methods reduce the likelihood of misidentifying legitimate traffic as malicious by focusing on deviations from normal behaviour.

2.5 Existing gaps

Gaps are existing in DDoS attack detection as the systems are all inadequate in some way. This section will focus on some of the gaps and limitations of these systems.

One limitation is the inability to detect zero-day attacks. Signature-based DDoS detection systems are unable to detect zero-day attacks (A. Fakeeh, 2016). This is because they can only identify attacks by matching them with known attack patterns.

Data availability is also another issue in detection systems, especially those using Machine Learning. ML algorithms rely heavily on training data for accurate recognition. Due to the limited availability of such data, it can be difficult to obtain large and diverse datasets containing labelled DDoS attacks. Furthermore, if the training data is biased toward a particular

attack type, the model may have difficulty detecting new and evolving attack techniques, resulting in reduced effectiveness.

Another issue is the scalability of these systems. ML and signature-based systems rely on training datasets and signature databases respectively. If these do not get updated regularly the systems will become inefficient in detecting the attacks. This process of updating requires significant effort and resources in itself.

High computational requirements are also another limitation of these systems. The process of monitoring network traffic while using ML, anomaly-based or signature-based detection systems can consume significant computing resources. This can lead to overhead which can impact network performance and latency, especially during large-scale DDoS attacks. Training and deploying the ML models also requires substantial processing power and storage resources.

Another limitation of these systems is high false negative rates. One of the major challenges of anomaly-based techniques is their susceptibility to high false negative rates (Mirkovic & Reiher, 2004). Because these techniques rely on detecting deviations from normal behaviour, they may misidentify normal behaviour as an attack.

2.6 Conceptual Framework

This section presents a conceptual framework for the proposed DDoS detection model. It outlines the key components and processes involved in the detection system, including inputs, processes, storage and outputs.

Entropy can be described as a measure of uncertainty. Entropy is a scientific concept that is most commonly connected with randomness, state of disorder, or uncertainty (Pardhi et al., 2022). This is the main reason for considering entropy as a DDoS detection system. The higher the randomness the higher the entropy is and vice versa. So, whenever the entropy is less than a threshold value, we can say that a DDoS attack occurs. The threshold entropy value can be determined by observing the model under normal traffic conditions. The following equation can then be used to calculate the value of the entropy, H .

$$H = - \sum_{x=1}^n P_i \log_2 P_i$$

Equation 2.1 The mathematical equation for Entropy calculation

The model would include a network topology with an SDN controller, multiple switches and hosts and this is where the target of the attack would be as seen in the diagram below. A predefined window size would need to be defined as we will calculate the entropy on a per-window basis. The window size used will be 50. The model would also have a warning counter to count the number of warnings before confirming that an attack has occurred.

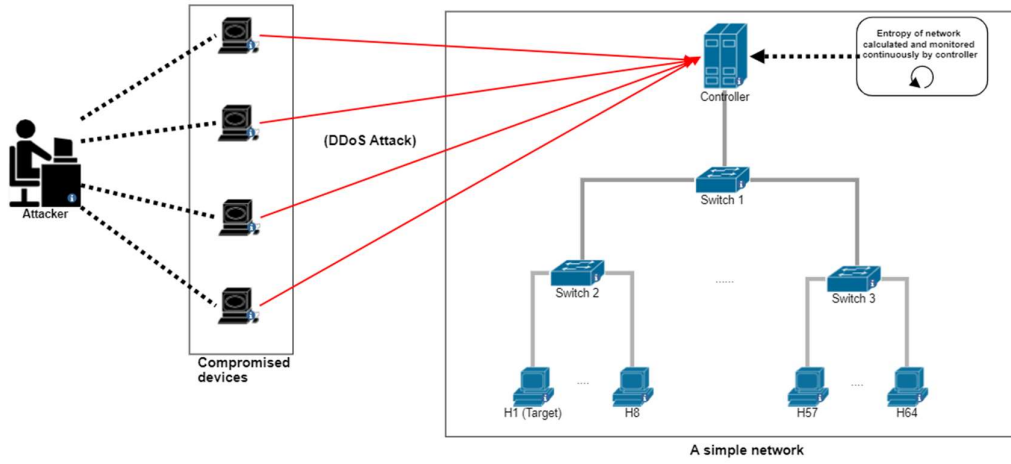


Figure 2.3 Conceptual framework

As new packets would enter the network, the controller would check the destination address of the packet and adds it to a list. It then checks if the packets have reached a pre-set number, 50, and if it has not, it does nothing and continues monitoring packets. The pre-set number is the window size spoken about earlier. If the number of entries in the list is 50, the contents are moved to a hash table where the controller calculates the entropy of that window and the list is cleared after. If the value of the entropy is above the threshold value, the system continues

monitoring the incoming packets normally. If the value of the entropy value is below the threshold, the system adds one to the warning counter value and warns the administrator of a potential attack. If the warning counter reaches a pre-set value, in our case, 5, a DDoS attack notification is sent.

Chapter 3: Methodology

3.1 Introduction

System development methodology refers to a framework that is used to structure, plan and control the process of developing an information system. This chapter involves a deeper description of the approaches and methodologies to be used in the development and testing of the proposed entropy-based DDoS detection model.

3.2 Research approach

The research approach that will be used is Object-Oriented Analysis and Design (OOAD). By allowing the identification of key entities, their relationships, and the behaviour of the system, OOAD offers a systematic approach to building software systems. The technique includes multiple phases, such as requirements collecting, system analysis, system design, and system implementation. These can also be seen in the Gantt Chart (see Appendix 1).

3.3 Methodology

For this project, the Agile methodology has been chosen. Agile methodology has gained significant recognition and popularity in project management due to its iterative and flexible approach. One of the main reasons for choosing the Agile methodology is its inherent adaptability. Highly structured methodologies struggle to manage changes and unplanned deployments, leading to delays and inefficient resource allocation. However, Agile methodologies consider change to be a natural part of the development process. Agile uses iterative cycles called sprints to enable teams to respond quickly to changing requirements, adjust priorities, and adjust project scope accordingly.

Another reason is that Agile methodologies focus on iterative development so that the work steps of a project can be carried out on a regular basis. This iterative approach allows for faster feedback loops, leading to the early detection of problems and deviations from project goals. This also allows changes to be incorporated early, minimizing the risk of rework. This iterative nature of Agile methodology ensures a more efficient and leaner development process.

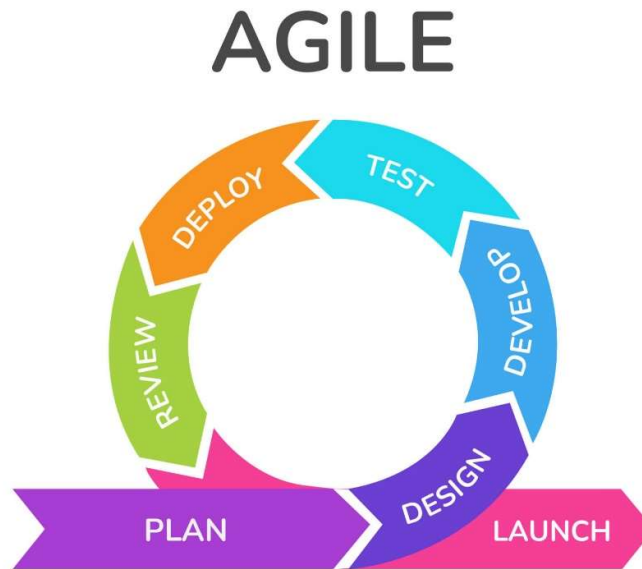


Figure 3.1 Agile methodology overview

3.3.1 Plan

This is the first step of the agile methodology. It is where the vision of the project is created and involves gathering relevant data, mapping processes and coming up with the scope. The data in this phase is analysed to ensure they are realistic and achievable. The functional requirement of the proposed system is that it should be able to detect DDoS attacks. The non-functional requirement is that it should be able to show the user what is happening during the attack.

3.3.2 Design

This step uses the data gotten from the planning phase to come up with the design of the system to optimize it with the right technology. According to the data obtained, the system should include a simple network topology.

3.3.3 Development

This phase involves the actual creation and coding of the system based on the designs obtained from the design phase. The system will be created using Python as a suitable programming language for the design.

3.3.4 Testing

This is the phase where the developed system is put to the test to find out if there are any bugs or problems in it. White box testing will be used for this phase.

3.3.5 Deployment

This is the phase that involves installation, configuration and making changes to optimize the performance of the model.

3.3.6 Review

This is the phase where the data obtained from the deployment phase is carefully examined to validate the quality, functionality and other components of the model. After doing this, further improvements can be made to the model or it can be launched.

3.3.7 Launch

This is the final phase of the methodology where the model is officially complete and ready. This is usually done after closely examining the reviews gotten earlier and not identifying any issues.

3.4 Deliverables

These will be the products of the project. These are:

- i. The proposed DDoS detection model
- ii. The research proposal and documentation of the project

3.5 Tools and Techniques

3.5.1 VirtualBox

Oracle VM VirtualBox is cross-platform virtualization software that allows users to extend their existing computer to run multiple operating systems. It will be used to run the Ubuntu 20.04 Virtual Machine which is the model environment.

3.5.2 Mininet

This is a high-performance network emulator that will be used to create the simulated network.

3.5.3 Python programming language

This will be the programming language used to write the code that simulates normal traffic, the DDoS attack, and the entropy calculations. Specifically, Python 3 will be used.

3.5.4 Scapy

This is a powerful interactive packet manipulation library in Python which can forge or decode packets of a wide number of protocols. It will be used to generate the packets used in the model.

Chapter 4: System Analysis and Design

4.1 Introduction

This chapter focuses on understanding the requirements of the system, analysing its components, and designing the structure that meets the project goal. To better understand the design, analysis and design diagrams have been drawn to visually represent the system's architecture and functions. These diagrams include a Use Case diagram, Sequence diagram, State Transition diagram, Systems Architecture diagram, Network Topology and Flowchart.

4.2 System Requirements

This section is divided into two parts, functional and non-functional requirements.

4.2.1 Functional Requirements

- i. The model should be capable of continuously monitoring incoming network traffic, analysing traffic details to detect potential DDoS attacks.
- ii. The model must be able to identify abnormal traffic patterns using calculations and differentiate them from normal traffic to detect potential DDoS attacks.
- iii. The model should generate warnings to notify administrators about any suspected DDoS attacks.
- iv. The model should promptly generate real-time alerts to notify administrators about detected DDoS attacks.

4.2.2 Non-Functional Requirements

- i. The DDoS detection process should be reliable and continuous, meaning minimal downtime.
- ii. The model should detect the DDoS attacks early.
- iii. It should be accurate, having few false negatives and false positives, ensuring accurate detection of DDoS attacks and preventing unnecessary disruptions.
- iv. The model should allow administrators to configure thresholds according to their network's requirements.

- v. The model should identify insider threats or suspicious behaviour from within the network, not only outer threats.
- vi. The model should be flexible and adaptable, detecting new DDoS attack techniques.

4.3 System Analysis Diagrams

4.3.1 Use Case diagram

This diagram illustrates the actors and the use cases of the model. The actors will be the User, who will generate traffic by simply using the computer on the network, the Attacker, who would be trying to launch a DDoS attack, and the Network Administrator, who will be in charge of managing the network. The model should be able to monitor the traffic as it enters or leaves the network and capture packets which it will then analyse to detect any potential DDoS attacks. If an attack is detected, it should then indicate this information to the Network Administrator.

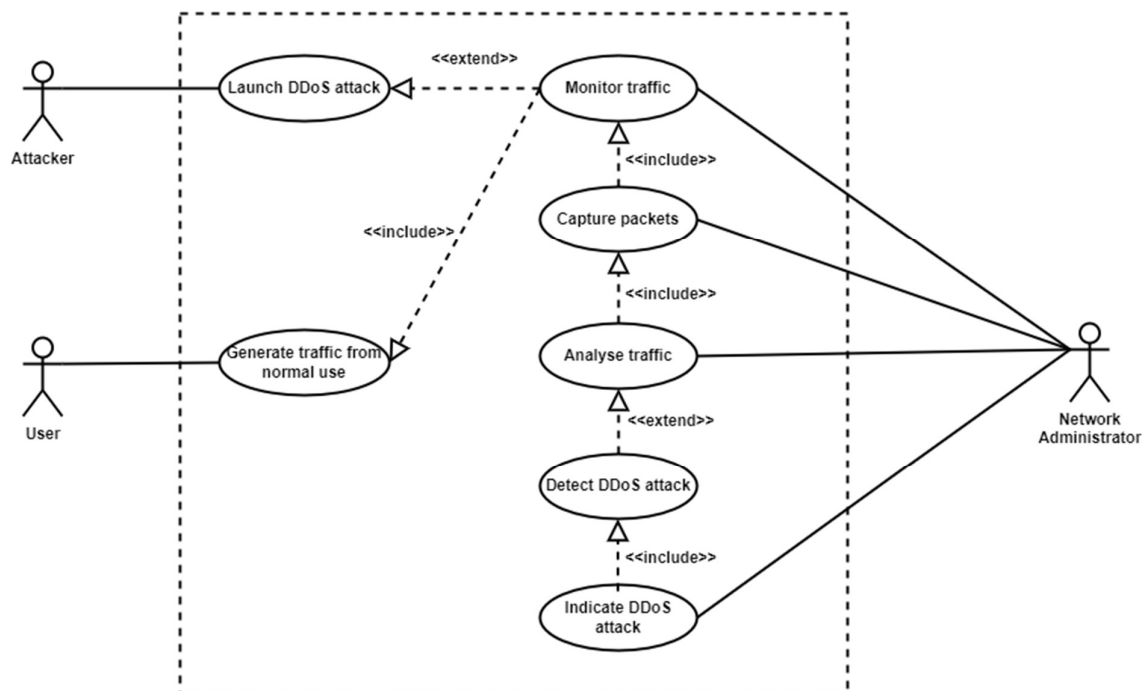


Figure 4.1 Use Case diagram

4.3.2 Sequence diagram

This diagram is used to visualize the interactions and communication flow between objects in the proposed model. The Attacker, would launch the DDoS attack on the network, targeting a specific host. The Network Controller as an object, would first analyse the incoming packets' destination IP addresses before forwarding them to their destination. The Switch would then forward the packets to the User who might start experiencing the effects of the attack. It would then use these destination IP addresses to calculate the entropy which it would compare to the predefined threshold. The Network Controller would then warn the Network Administrator of a potential DDoS attack if the threshold has been reached. Five consecutive warnings would then result in the Network Administrator being notified of the attack. The Network Administrator would then analyse the network and prepare mitigation measures.

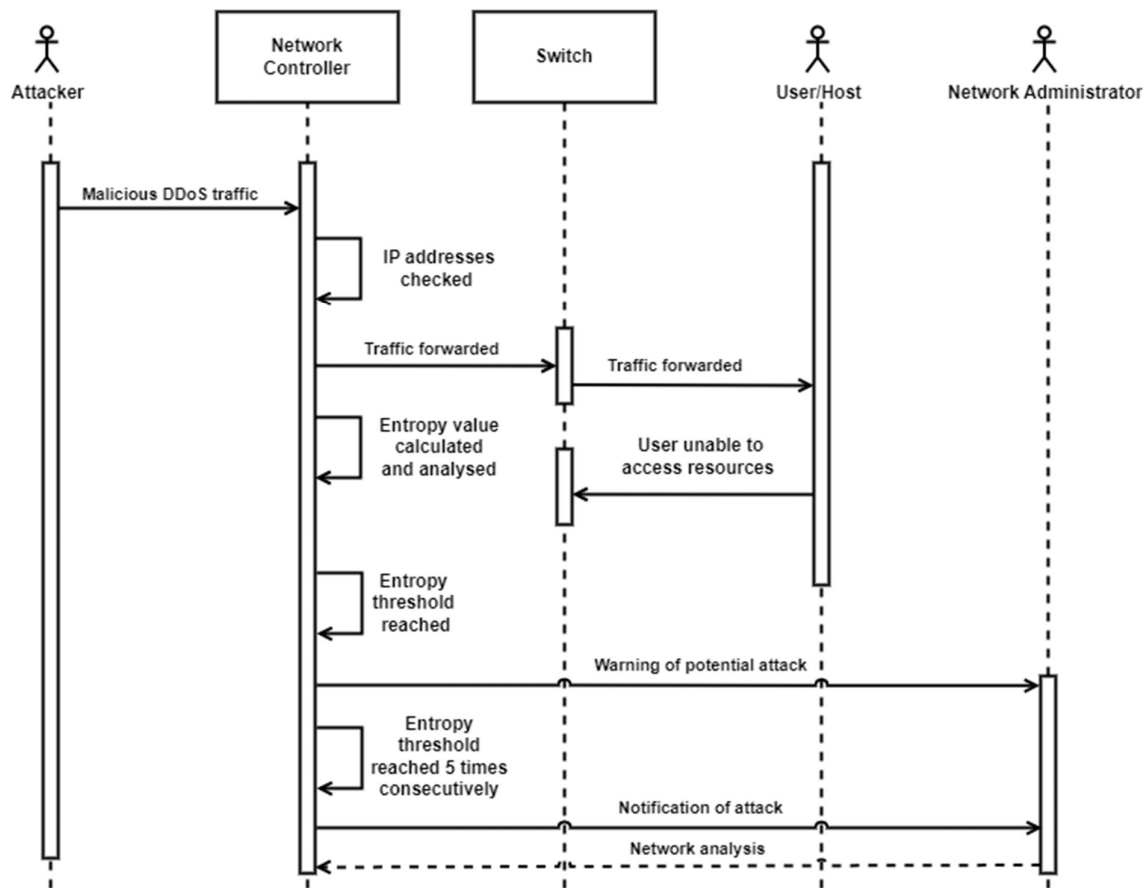


Figure 4.2 Sequence diagram

4.3.3 Flowchart

This diagram will help in understanding the data flow in the model. First, traffic enters the network. The traffic is then captured and parsed for the destination IP addresses which are the used to calculate the entropy. The entropy is then compared to the threshold set by the network administrator. If it is higher, it continues capturing packets, otherwise it checks if that is the 5th consecutive time it has occurred. If it is not, it sends out a warning of a potential attack. If it is, it notifies the administrator of a detected DDoS attack.

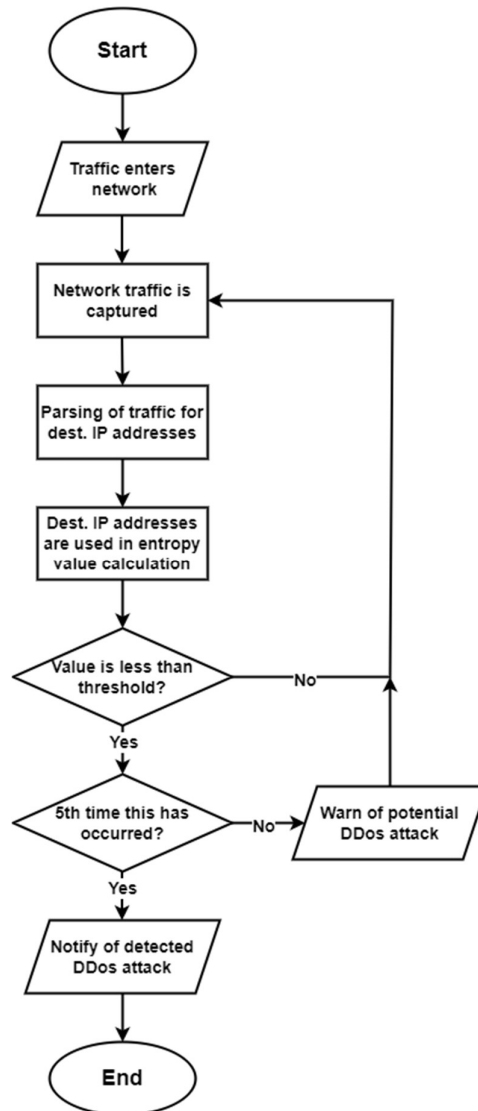


Figure 4.3 Flowchart

4.3.4 State Transition diagram

This diagram is used to explain the states gone through during the operation of the model. It first starts in the Idle state whereby it is on standby. It then transitions to the Packet capturing state when packets start being sent in and out of the network. It then transitions to the Calculating state when it has the information needed from the packets to calculate the entropy i.e., the destination IP addresses. It then transitions to the Evaluating state where it evaluates the entropy value by comparing it to the threshold value. If nothing is detected it transitions back to the Idle state. If an attack is detected, it transitions to the Attack detected state and notifies the administrator of this attack.

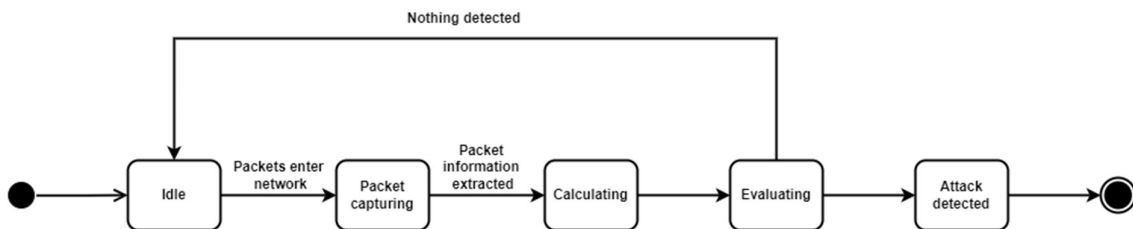


Figure 4.4 State Transition diagram

4.3.5 Systems Architecture diagram

This diagram provides an overview of how various elements of the model interact and work together. The attacker launches a DDoS attack using compromised devices. The Packet Collector collects the packets and the Packet Parser extracts the destination IP addresses and places them in an IP address list before the packets are forwarded to the intended destination. Once the IP address list is full with 50 addresses, its contents are copied to a Hash Table which stores each address together with a number showing how many times the address has occurred. The contents of the Hash Table are then forwarded to the Entropy Calculator which uses them to calculate the entropy. The entropy value is then sent to the Comparison Tool which first compares this value to the threshold value. If it is lower than the threshold, the tool then checks the Warning Counter to see how many warnings have been issued. If there have already been 4 warning issued, it sends an Attack Notification to the Network Administrator.

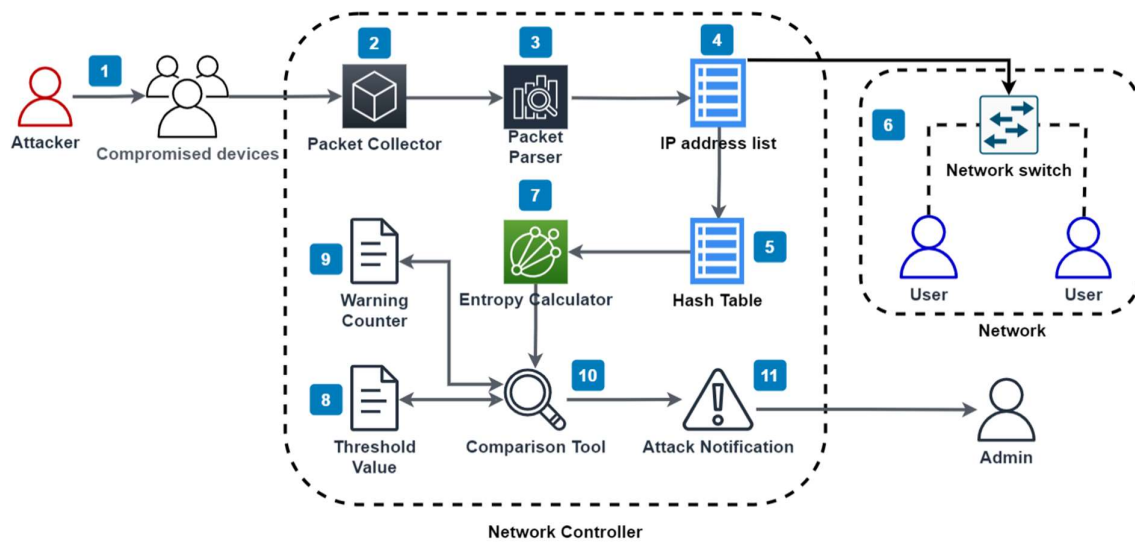


Figure 4.5 Systems Architecture diagram

4.3.6 Network Topology

This is a simple illustration of how the network will look. The network controller will have the IP address 127.0.0.1. There will be 9 switches in two layers, one layer with one and another with 8. Each switch will have 8 hosts each with their own IP address ranging from 10.0.0.1 to 10.0.0.64.

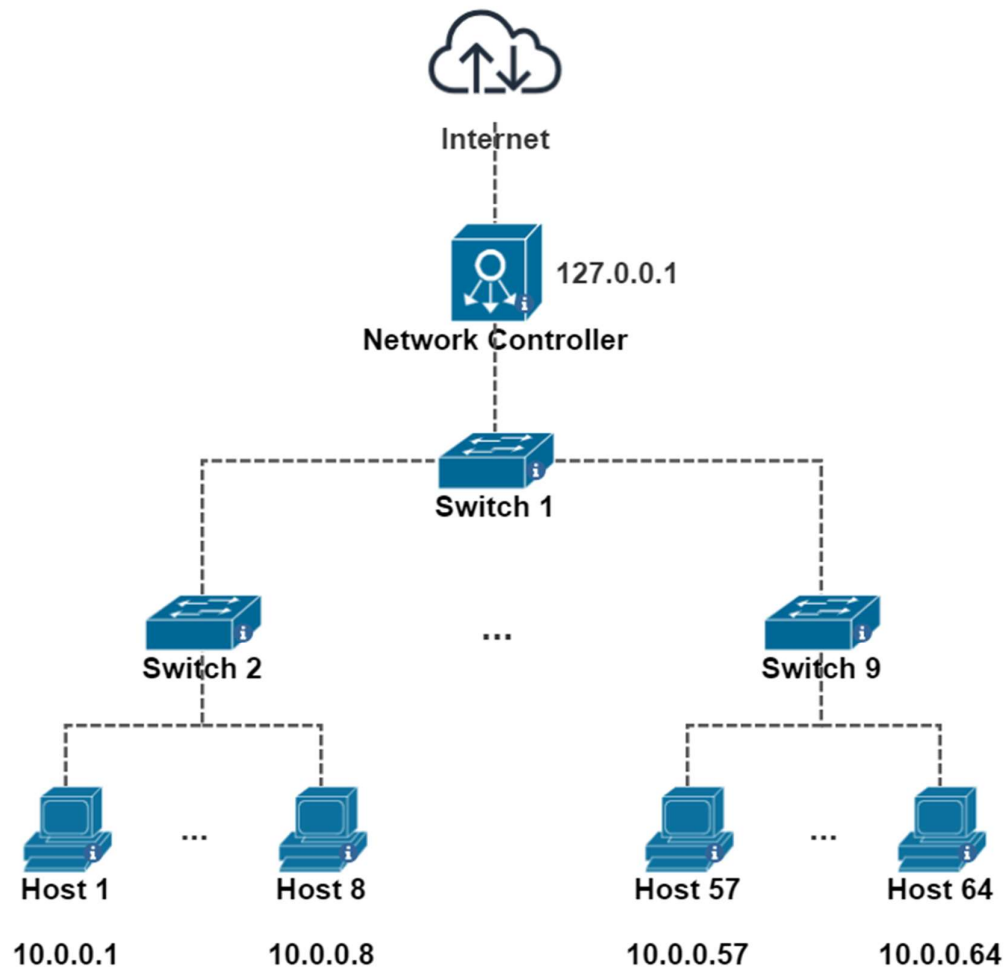


Figure 4.6 The Network Topology

Chapter 5: Implementation and Testing

5.1 Introduction

This chapter presents the implementation details and testing procedures of the project. In this section, we delve into the technical aspects of how the system was developed and the various tools and technologies employed. Additionally, the chapter discusses the testing environment set up to validate the functionality and performance of the system.

5.2 Discussion of Implementation Environment

This section provides an overview of the hardware and software specifications used in the project.

5.2.1 Hardware Specifications

The project was implemented on a laptop with the following specifications:

Manufacturer	Lenovo
Model	81WD
Memory (RAM)	8 GB
Storage	475 GB
Processor	Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz
Graphics adapter	Intel UHD Graphics G1 (Ice Lake 32 EU)
Networking	802.11 a/b/g/n/ac

Table 5.1 Table of Hardware specifications

5.2.2 Software Specifications

The software used in the project included a range of tools and programming languages. The software specifications are as follows:

1. Windows 11 Pro (64 bit): This is the OS that was running on the laptop used. Any version of Windows from Windows 8.1 onwards could be used here since we are using VirtualBox 7.
2. Oracle VM VirtualBox: VirtualBox version 7.0.8 was used for this project as it was the latest version at the time. It was used to run the Ubuntu VM.
3. Ubuntu (20.04): This OS was installed as a Virtual Machine inside VirtualBox. This specific version was used because the python version needed by the POX controller was between 3.7 to 3.9. Ubuntu 20.04 already came with Python 3.8 installed.
4. Python 3.8: This is the programming language used to write the code throughout the project.
5. Scapy: Scapy is the tool used in generation of packets and spoofing the source IP address of the packets. Refer to Appendix 2 for installation screenshots.
6. Pip: “Pip Installs Packages” or “Pip Installs Python” is a cross-platform package manager for installing and managing Python packages. It is used here to install POX. Refer to Appendix 3 for installation screenshots.
7. Mininet: Mininet is the network emulator used for the creation of the network topology used in this project. Refer to Appendix 4 for installation screenshots.
8. POX: This is the controller used in the project. The choice was between Opendaylight, Ryu, Floodlight, Beacon, POX, ONOS, OpenMUL and Maestro but POX is used because it runs on Python. Refer to Appendix 5 for installation screenshots.
9. Vim: Vim is the text editor used to write the Python code used in the project. It has been chosen because it runs in Linux. Refer to Appendix 6 for installation screenshots.

10. Xterm: XTerm is the standard terminal emulator that is used for the hosts on the network. Refer to Appendix 7 for installation screenshots.

5.3 Discussion of Testing Environment

The testing phase was instrumental in verifying the correctness and reliability of the system. White Box testing is the paradigm that was used for the testing of the model. This was possible because I had access to the source code and had an understanding of it. This level of transparency would allow better understanding of the underlying issues, allowing the pinpointing of issues in the model.

The testing environment was similar to the implementation environment as it was done on the same device and virtual environment. This section outlines the testing environment used to conduct various tests and validations.

5.3.1 Functional Requirements Testing

Functional Requirement	Test Data	Expected Result	Actual Result	Evidence
Continuous monitoring of incoming network traffic	Python script simulating normal network traffic	The traffic should be monitored for as long as the network is active. No packets should be missed.	Pass	Refer to Appendix 8
Identification of abnormal traffic patterns using calculations	Python script simulating DDoS traffic	The entropy value should be calculated as long as there is traffic flowing through the network.	Pass	Refer to Appendix 9

DDoS attack warning	Python script simulating DDoS traffic	All four warnings should be generated whenever the DDoS traffic is introduced.	Pass	Refer to Appendix 10
DDoS attack alert	Python script simulating DDoS traffic	The alert should only be generated when there is DDoS traffic in the network. It should also not come up unless all four warnings have been generated beforehand.	Pass	Refer to Appendix 11

Table 5.2 Table of Functional Requirements Testing

5.3.2 Non-Functional Requirements Testing

Non-Functional Requirement	Result
The DDoS detection process should be reliable and continuous, meaning minimal downtime.	The model was successful in meeting this requirement. The traffic was analysed for as long as the machine was turned on.
The model should detect the DDoS attacks early.	The model was successful in meeting this requirement. By analysing the entropy after every 50 packets, it would take at most approximately 250 packets of DDoS traffic to alert of the attack.
It should be accurate, having few false negatives and false positives, ensuring	The model was successful in meeting this requirement. By including four warnings

accurate detection of DDoS attacks and preventing unnecessary disruptions.	before the attack indication, the false positive rate and false negative rate is reduced.
The model should allow administrators to configure thresholds according to their network's requirements.	The model was successful in meeting this requirement. The administrator could configure the threshold easily but they would need a basic understating of Python to do this.
The model should identify insider threats or suspicious behaviour from within the network, not only outer threats.	The model was successful in meeting this requirement. Since the entropy is measured for both outgoing and incoming packets, both insider and outsider DDoS threats could be detected equally.
The model should be flexible and adaptable, detecting new DDoS attack techniques.	The model should be successful in meeting this requirement. Since it falls under the category of anomaly detection techniques, any new technique should be detected.

Table 5.3 Table of Non-Functional Requirements Testing

Chapter 6: Conclusions and Recommendations

6.1 Conclusions

In conclusion, this project has successfully developed an entropy-based DDoS detection model. By analysing and processing packet information for potential DDoS behaviour, the model enhances cybersecurity measures and provides an additional layer of protection to the network and its users. The implementation and testing phases of this project have demonstrated that the system meets the predefined requirements and is accurate in detecting DDoS attacks.

6.2 Recommendations

Based on the findings and results of the project, the following recommendations are proposed to further improve the model:

- i. Real-World testing: Conducting real-world deployment to evaluate the performance and effectiveness of the entropy-based DDoS detection model in live network environments would make the results from the testing phase more accurate. This would provide valuable insights and validate the model's practical use.
- ii. Signature-based analysis: Adding behavioural analysis techniques to the DDoS detection system would improve the detection process significantly. The analysis of traffic behaviour patterns combined with entropy-based techniques would provide a more accurate and reliable defence against different types of DDoS attacks.
- iii. Anomaly categorization: Categorizing and classifying DDoS attacks based on their severity and characteristics would help in getting an overview of which attacks it can detect more accurately. This could help in prioritizing responses and dealing with the most critical DDoS threats first.
- iv. Adding a Graphical User Interface (GUI): Developing a user-friendly GUI that allows network administrators to easily interact with the system would simplify the process of managing the threshold.

6.3 Future Works

While the entropy-based DDoS detection model has made significant progress, there are several potential directions for future research and development:

- i. Integration with mitigation mechanisms: Integrating entropy-based detection with one or more mitigation mechanisms would create a more comprehensive defence strategy. When an attack is detected, automated mitigation tools can be triggered to block or redirect malicious traffic and prevent damage to the network resources.
- ii. Machine Learning integration: Integrating machine learning algorithms with the entropy-based detection would improve the system's ability to adapt to new attack patterns. By continuously learning from new data, the system can improve its accuracy and detection capabilities greatly.
- iii. Multiparameter entropy analysis: Investigating multi-parameter entropy analysis that considers not only packets' destination IP addresses but also other network parameters may provide a more comprehensive analysis of network behaviour relating to DDoS attacks.

Bibliography

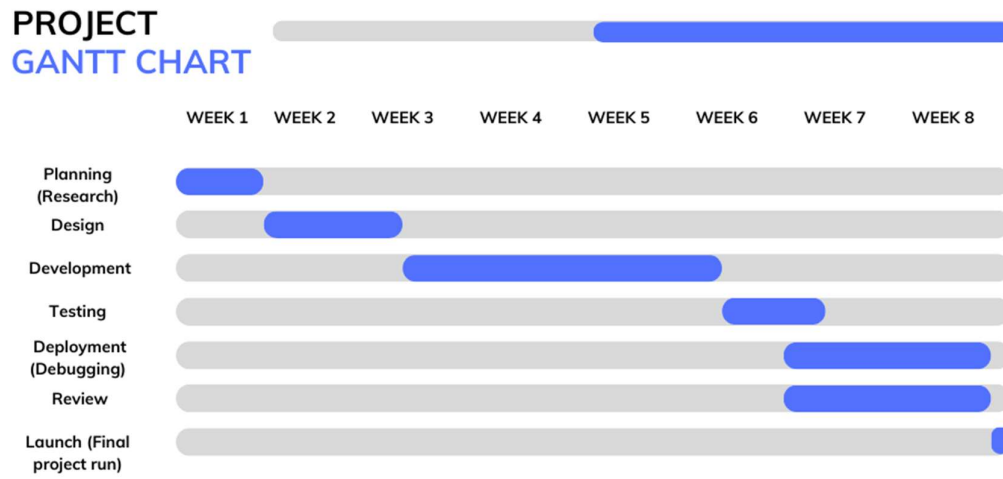
- A. Fakeeh, K. (2016). An Overview of DDOS Attacks Detection and Prevention in the Cloud. *International Journal of Applied Information Systems*, 11(7), 25–34.
<https://doi.org/10.5120/ijais2016451628>
- Abhishta, A. (2019). *The blind man and the elephant* [PhD, University of Twente].
<https://doi.org/10.3990/1.9789036549127>
- Alomari, E., Manickam, S., Gupta, B. B., Karuppayah, S., & Alfari, R. (2012). Botnet-based Distributed Denial of Service (DDoS) Attacks on Web Servers: Classification and Art. *International Journal of Computer Applications*, 49(7), 24–32. <https://doi.org/10.5120/7640-0724>
- Asosheh, A., & Ivaki, N. (2008). A comprehensive taxonomy of DDOS attacks and defense mechanism applying in a smart classification. *WSEAS Transactions on Computers*, 7, 281–290.
- Baldwin, C. (2016, August 3). Bitcoin worth \$72 million stolen from Bitfinex exchange in Hong Kong. *Reuters*. <https://www.reuters.com/article/us-bitfinex-hacked-hongkong-idUSKCN10E0KP>
- Bhayo, J., Shah, S. A., Hameed, S., Ahmed, A., Nasir, J., & Draheim, D. (2023). Towards a machine learning-based framework for DDOS attack detection in software-defined IoT (SD-IoT) networks. *Engineering Applications of Artificial Intelligence*, 123, 106432.
<https://doi.org/10.1016/j.engappai.2023.106432>
- Bishop, M. (2006). *Introduction to Computer Security* (1st ed.). Addison-Wesley Professional.
- Chandola, V., Banerjee, A., & Kumar, V. (2009). *Anomaly Detection: A Survey*.
<http://cucis.ece.northwestern.edu/projects/DMS/publications/AnomalyDetection.pdf>
- De Donno, M., Dragoni, N., Giaretta, A., & Spognardi, A. (2018). DDoS-Capable IoT Malwares: Comparative Analysis and Mirai Investigation. *Security and Communication Networks*, 2018, e7178164. <https://doi.org/10.1155/2018/7178164>
- Douligeris, C., & Mitrokotsa, A. (2004). DDoS attacks and defense mechanisms: Classification and state-of-the-art. *Computer Networks*, 44(5), 643–666.
<https://doi.org/10.1016/j.comnet.2003.10.003>
- Flynn, J. (2023, January 12). How Many People Use The Internet? [2023]: 35 Facts About Internet Usage In America And The World. *Zippia*. <https://www.zippia.com/advice/how-many-people-use-the-internet/>
- Gheorge, A. (2015, May 31). *GitHub under 'Largest DDoS in Site History'*.
<https://www.bitdefender.com/blog/hotforsecurity/github-under-largest-ddos-in-site-history/>
- Goncharov, M. (2012). *Russian Underground 101*.
<https://go.trendmicro.com/archive/docs/wp-russian-underground-101.pdf>

- Gupta, B. B., Joshi, R. C., & Misra, M. (2009). Defending against Distributed Denial of Service Attacks: Issues and Challenges. *Information Security Journal: A Global Perspective*, 18(5), 224–247. <https://doi.org/10.1080/19393550903317070>
- Hoque, N., Bhattacharyya, D. K., & Kalita, J. K. (2015). Botnet in DDoS Attacks: Trends and Challenges. *IEEE Communications Surveys & Tutorials*, 17(4), 2242–2270. <https://doi.org/10.1109/COMST.2015.2457491>
- Idhammad, M., Afdel, K., & Belouch, M. (2018). Semi-supervised machine learning approach for DDoS detection. *Applied Intelligence*, 48(10), 3193–3208. <https://doi.org/10.1007/s10489-018-1141-2>
- Kottler, S. (2018, March 1). February 28th DDoS Incident Report. *The GitHub Blog*. <https://github.blog/2018-03-01-ddos-incident-report/>
- Kupreev, O., Gutnikov, A., & Shmelev, Y. (2022, November 7). *Report on DDoS attacks in Q3 2022*. <https://securelist.com/ddos-report-q3-2022/107860/>
- Makrushin, D. (2017, March 23). *The cost of launching a DDoS attack*. <https://securelist.com/the-cost-of-launching-a-ddos-attack/77784/>
- Mirkovic, J., & Reiher, P. (2004). A taxonomy of DDoS attack and DDoS Defense mechanisms. *ACM SIGCOMM Computer Communication Review*, 34. <https://doi.org/10.1145/997150.997156>
- Mitchell, T. M. (1997). Does Machine Learning Really Work? *AI Magazine*, 18(3), Article 3. <https://doi.org/10.1609/aimag.v18i3.1303>
- Nicholson, P. (2020, June 24). *AWS hit by Largest Reported DDoS Attack of 2.3 Tbps | A10 Networks*. <https://www.a10networks.com/blog/aws-hit-by-largest-reported-ddos-attack-of-2-3-tbps/>
- Pardhi, P. R., Rout, J. K., & Ray, N. K. (2022). A Study on Performance Comparison of Algorithms for Detecting the Flooding DDoS Attack. *2022 OITS International Conference on Information Technology (OCIT)*, 433–438. <https://doi.org/10.1109/OCIT56763.2022.00087>
- Patrikakis, C., Masikos, M., & Zouraraki, O. (2019, August 26). *Distributed Denial of Service Attacks—The Internet Protocol Journal—Volume 7, Number 4—Cisco*. <https://web.archive.org/web/20190826143507/https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-30/dos-attacks.html>
- Petrosyan, A. (2023, May 22). *Internet and social media users in the world 2023*. Statista. <https://www.statista.com/statistics/617136/digital-population-worldwide/>
- Pfleeger, C. P., Pfleeger, S. L., & Margulies, J. (2015). *Security in Computing* (5th ed.). Pearson.
- Sarker, I. H. (2021). Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*, 2(3), 160. <https://doi.org/10.1007/s42979-021-00592-x>

- Turner, G. (2022, August 4). DDoS attack durations see sharp rise throughout Q2 2022. *DIGIT*. <https://www.digit.fyi/ddos-attack-durations-see-sharp-rise-throughout-q2-2022/>
- van den Dool, P. (2013, April 10). *ING ondanks maatregelen getroffen door nieuwe DDos-aanval*. NRC. <https://www.nrc.nl/nieuws/2013/04/10/ing-nieuwe-cyberaanval-sneller-afgeslagen-door-maatregelen-a1435706>
- Weiler, N. (2002). Honeypots for distributed denial-of-service attacks. *Proceedings. Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 109–114. <https://doi.org/10.1109/ENABL.2002.1029997>
- Whitman, M. E., & Mattord, H. J. (2017). *Principles of Information Security* (6th ed.). Cengage Learning.
- Zargar, S. T., Joshi, J., & Tipper, D. (2013). A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. *IEEE Communications Surveys & Tutorials*, 15(4), 2046–2069. <https://doi.org/10.1109/SURV.2013.031413.00127>
- Zekri, M., Kafhali, S. E., Aboutabit, N., & Saadi, Y. (2017). DDoS attack detection using machine learning techniques in cloud computing environments. *2017 3rd International Conference of Cloud Computing Technologies and Applications (CloudTech)*, 1–7. <https://doi.org/10.1109/CloudTech.2017.8284731>

Appendices:

Appendix 1: Gantt Chart



Appendix 2: Scapy installation screenshots

```
andrew@andrew-VM20:/media/andrew/VBox_GAs_7.0.8$ sudo apt-get install python3-scapy
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  python3-backcall python3-decorator python3-ipython python3-ipython-genutils
  python3-jedi python3-parso python3-pickleshare python3-prompt-toolkit python3-pygments
  python3-traitlets python3-wcwidth
Suggested packages:
  python-ipython-doc python-pygments-doc ttf-bitstream-vera graphviz python3-matplotlib
  python3-pyx sox tclreplay wireshark
The following NEW packages will be installed:
  python3-backcall python3-decorator python3-ipython python3-ipython-genutils
  python3-jedi python3-parso python3-pickleshare python3-prompt-toolkit python3-pygments
  python3-scapy python3-traitlets python3-wcwidth
0 upgraded, 13 newly installed, 0 to remove and 180 not upgraded.
Need to get 2,684 kB of archives.
After this operation, 16.6 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ke.archive.ubuntu.com/ubuntu focal/universe amd64 python3-backcall all 0.3.0-2 [11.2 kB]
Get:2 http://ke.archive.ubuntu.com/ubuntu focal/main amd64 python3-decorator all 4.4.2-0ubuntu1 [10.3 kB]
Get:3 http://ke.archive.ubuntu.com/ubuntu focal/universe amd64 python3-parso all 0.5.2-1ubuntu1 [62.0 kB]
Get:4 http://ke.archive.ubuntu.com/ubuntu focal/universe amd64 python3-jedi all 0.15.2-1 [382 kB]
Get:5 http://ke.archive.ubuntu.com/ubuntu focal/universe amd64 python3-pickleshare all 0.7.5-2 [7,360 B]
Get:6 http://ke.archive.ubuntu.com/ubuntu focal/main amd64 python3-wcwidth all 0.1.8+dfsg1-3 [17.4 kB]
Get:7 http://ke.archive.ubuntu.com/ubuntu focal/universe amd64 python3-prompt-toolkit all 2.0.10-2 [220 kB]
Get:8 http://ke.archive.ubuntu.com/ubuntu focal-updates/main amd64 python3-pygments all 2.3.1+dfsg-1ubuntu2.2 [579 kB]
Get:9 http://ke.archive.ubuntu.com/ubuntu focal/universe amd64 python3-ipython-genutils all 0.2.0-1ubuntu1 [21.1 kB]
Get:10 http://ke.archive.ubuntu.com/ubuntu focal/universe amd64 python3-traitlets all 4.3.3-1 [59.8 kB]
Get:11 http://ke.archive.ubuntu.com/ubuntu focal/universe amd64 python3-ipython all 7.13.0-1 [494 kB]
Get:12 http://ke.archive.ubuntu.com/ubuntu focal/universe amd64 python3-scapy all 2.4.3-4 [694 kB]
Fetched 2,684 kB in 1s (379 kB/s)
Selecting previously unselected package python3-backcall.
(Reading database ... 15989 files and directories currently installed.)
Preparing to unpack .../00-python3-backcall_0.3.0-2_all.deb ...
Unpacking python3-backcall (0.3.0-2) ...
Selecting previously unselected package python3-decorator.
Preparing to unpack .../01-python3-decorator_4.4.2-0ubuntu1_all.deb ...
Unpacking python3-decorator (4.4.2-0ubuntu1) ...
Selecting previously unselected package python3-parso.
Preparing to unpack .../02-python3-parso_0.5.2-1ubuntu1_all.deb ...
Unpacking python3-parso (0.5.2-1ubuntu1) ...
Selecting previously unselected package python3-jedi.
Preparing to unpack .../03-python3-jedi_0.15.2-1_all.deb ...
Unpacking python3-jedi (0.15.2-1) ...
Selecting previously unselected package python3-pickleshare.
Preparing to unpack .../04-python3-pickleshare_0.7.5-2_all.deb ...
Unpacking python3-pickleshare (0.7.5-2) ...
Selecting previously unselected package python3-wcwidth.
Preparing to unpack .../05-python3-wcwidth_0.1.8+dfsg1-3_all.deb ...
Unpacking python3-wcwidth (0.1.8+dfsg1-3) ...
Selecting previously unselected package python3-prompt-toolkit.
Preparing to unpack .../06-python3-prompt-toolkit_2.0.10-2_all.deb ...
Unpacking python3-prompt-toolkit (2.0.10-2) ...
Selecting previously unselected package python3-pygments.
Preparing to unpack .../07-python3-pygments_2.3.1+dfsg-1ubuntu2.2_all.deb ...
Unpacking python3-pygments (2.3.1+dfsg-1ubuntu2.2) ...
Selecting previously unselected package python3-ipython-genutils.
Preparing to unpack .../08-python3-ipython-genutils_0.2.0-1ubuntu1_all.deb ...
Unpacking python3-ipython-genutils (0.2.0-1ubuntu1) ...
Selecting previously unselected package python3-traitlets.
Preparing to unpack .../09-python3-traitlets_4.3.3-1_all.deb ...
Unpacking python3-traitlets (4.3.3-1) ...
Selecting previously unselected package python3-ipython.
Preparing to unpack .../10-python3-ipython_7.13.0-1_all.deb ...
Unpacking python3-ipython (7.13.0-1) ...
Selecting previously unselected package python3-scapy.
Preparing to unpack .../11-python3-scapy_2.4.3-4_all.deb ...
Unpacking python3-scapy (2.4.3-4) ...
Setting up python3-backcall (0.3.0-2) ...
Setting up python3-parso (0.5.2-1ubuntu1) ...
Setting up python3-ipython-genutils (0.2.0-1ubuntu1) ...
Setting up python3-decorator (4.4.2-0ubuntu1) ...
Setting up python3-wcwidth (0.1.8+dfsg1-3) ...
Setting up python3-pickleshare (0.7.5-2) ...
Setting up python3-scapy (2.4.3-4) ...
Setting up python3-traitlets (4.3.3-1) ...
Setting up python3-prompt-toolkit (2.0.10-2) ...
Setting up python3-jedi (0.15.2-1) ...
Setting up python3-ipython (7.13.0-1) ...
Setting up python3 (7.13.0-1) ...
Processing triggers for man-db (2.9.1-1) ...
```

Appendix 3: Pip installation screenshots

```
andrew@andrew-VM20:/media/andrew/VBox_GAs_7.0.8$ sudo apt install python3-pip

Linux Ubuntu 20.04 [Running] - Oracle VM VirtualBox
7 Jun 23:59
andrew@andrew-VM20:/media/andrew/VBox_GAs_7.0.8

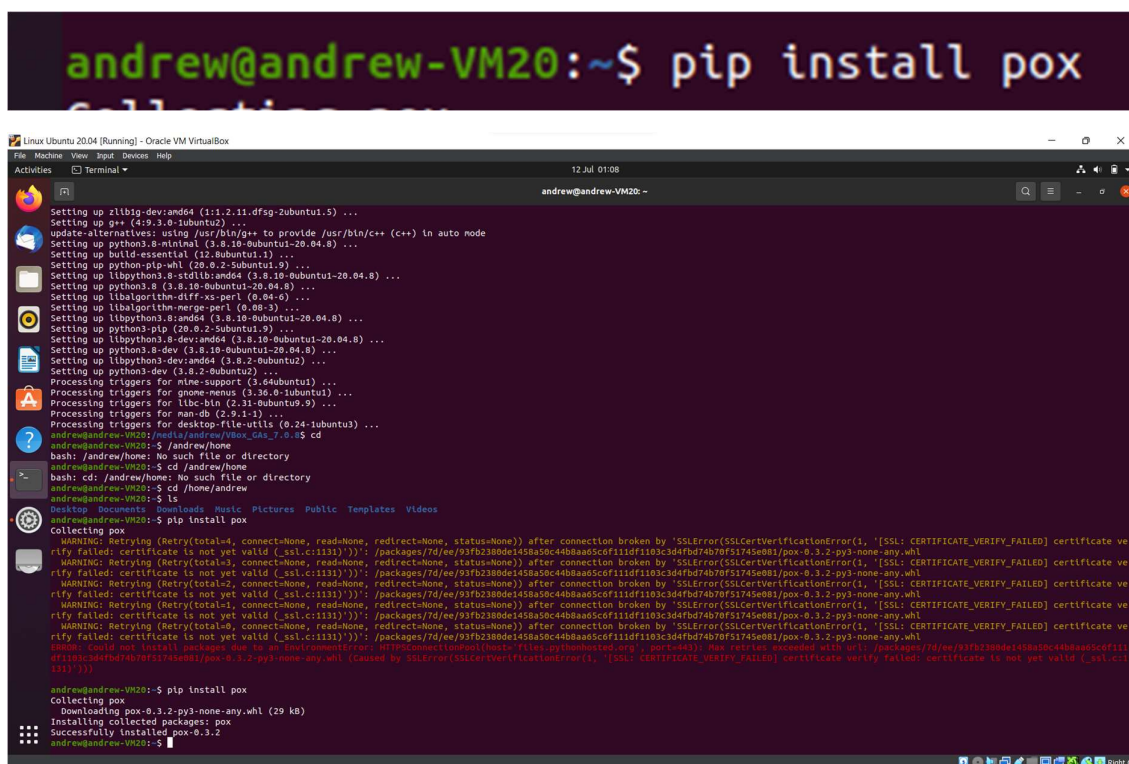
andrew@andrew-VM20:/media/andrew/VBox_GAs_7.0.8$ sudo apt install python3-pip
[sudo] password for andrew:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  build-essential dpkg-dev fakeroot g++ g++-9 libalgorithm-diff-perl libalgorithm-diff-xs-perl
  libalgorithm-merge-perl libexpat1-dev libfakeroot libpython3-dev libpython3.8
  libpython3.8-dev libpython3.8-minimal libpython3.8-stdlib libstdc++-9-dev python-pip-whl
  python3-dev python3-distutils python3-setuptools python3-wheel python3.8 python3.8-dev
  python3.8-minimal zlib1g-dev
Suggested packages:
  debhelper g++-multilib g++-9-multilib gcc-9-doc libstdc++-9-doc python-setuptools-doc
  python3.8-venv python3.8-doc binfmt-support
The following NEW packages will be installed:
  build-essential dpkg-dev fakeroot g++ g++-9 libalgorithm-diff-perl libalgorithm-diff-xs-perl
  libalgorithm-merge-perl libexpat1-dev libfakeroot libpython3-dev libpython3.8-dev
  libstdc++-9-dev python-pip-whl python3-dev python3-distutils python3-pip python3-setuptools
  python3-wheel python3.8 python3.8-dev
The following packages will be upgraded:
  libpython3.8 libpython3.8-minimal libpython3.8-stdlib python3.8 python3.8-minimal
5 upgraded, 21 newly installed, 0 to remove and 375 not upgraded.
Need to get 18.3 MB/24.6 MB of archives.
After this operation, 77.9 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://ke.archive.ubuntu.com/ubuntu focal-updates/main amd64 libstdc++-9-dev amd64 9.4.0-1ubuntu1-20.04.1 [1,722 kB]
Get:2 http://ke.archive.ubuntu.com/ubuntu focal-updates/main amd64 g++-9 amd64 9.4.0-1ubuntu1-20.04.1 [8,420 kB]
Get:3 http://ke.archive.ubuntu.com/ubuntu focal/main amd64 g++ amd64 4:9.3.0-1ubuntu2 [1,064 B]
Get:4 http://ke.archive.ubuntu.com/ubuntu focal-updates/main amd64 dpkg-dev all 1.19.7ubuntu3.2 [679 kB]
Get:5 http://ke.archive.ubuntu.com/ubuntu focal-updates/main amd64 build-essential amd64 12.8ubuntu1.1 [4,664 B]
Get:6 http://ke.archive.ubuntu.com/ubuntu focal/main amd64 libfakeroot amd64 1.24-1 [25.7 kB]
Get:7 http://ke.archive.ubuntu.com/ubuntu focal/main amd64 fakeroot amd64 1.24-1 [62.6 kB]
Get:8 http://ke.archive.ubuntu.com/ubuntu focal/main amd64 libalgorithm-diff-perl all 1.19.03-2 [6.6 kB]
Get:9 http://ke.archive.ubuntu.com/ubuntu focal/main amd64 libalgorithm-diff-xs-perl amd64 0.04-6 [11.3 kB]
Get:10 http://ke.archive.ubuntu.com/ubuntu focal/main amd64 libalgorithm-merge-perl all 0.08-3 [12.0 kB]
Get:11 http://ke.archive.ubuntu.com/ubuntu focal-updates/main amd64 libexpat1-dev amd64 2.2.9-1ubuntu0.6 [110 kB]
Get:12 http://ke.archive.ubuntu.com/ubuntu focal-updates/main amd64 libpython3.8-dev amd64 3.8.10-0ubuntu1-20.04.8 [3,950 kB]
Get:13 http://ke.archive.ubuntu.com/ubuntu focal/main amd64 libpython3-dev amd64 3.8.2-0ubuntu2 [7,236 B]
Get:14 http://ke.archive.ubuntu.com/ubuntu focal-updates/universe amd64 python-pip-whl all 20.0.2-Subunit9.9 [5,089 kB]
Get:15 http://ke.archive.ubuntu.com/ubuntu focal-updates/main amd64 python3-dev amd64 3.8.10-0ubuntu1-20.04.8 [155 kB]
Get:16 http://ke.archive.ubuntu.com/ubuntu focal-updates/main amd64 python3.8-dev amd64 3.8.10-0ubuntu1-20.04.8 [514 kB]
Get:17 http://ke.archive.ubuntu.com/ubuntu focal-updates/main amd64 python3-distutils all 3.8.10-0ubuntu1-20.04.8 [141 kB]
Get:18 http://ke.archive.ubuntu.com/ubuntu focal-updates/main amd64 python3-setuptools all 45.2.0-1ubuntu0.1 [330 kB]
Get:19 http://ke.archive.ubuntu.com/ubuntu focal-updates/main amd64 python3-wheel all 0.34.2-0ubuntu1 [73.9 kB]
Get:20 http://ke.archive.ubuntu.com/ubuntu focal-updates/main amd64 python3-pip all 20.0.2-Subunit9.9 [221 kB]
Get:21 http://ke.archive.ubuntu.com/ubuntu focal-updates/universe amd64 python3-pip all 20.0.2-Subunit9.9 [221 kB]
Fetched 18.3 MB in 16s (1,163 kB/s)
(Reading database ... 10759 files and directories currently installed.)
Preparing to unpack .../00-python3.8_3.8.10-0ubuntu1-20.04.8_amd64.deb ...
Unpacking python3.8 (3.8.10-0ubuntu1-20.04.8) over (3.8.10-0ubuntu1-20.04.6) ...
Preparing to unpack .../01-libpython3.8_3.8.10-0ubuntu1-20.04.8_amd64.deb ...
Unpacking libpython3.8:amd64 (3.8.10-0ubuntu1-20.04.8) over (3.8.10-0ubuntu1-20.04.6) ...
Preparing to unpack .../02-libpython3.8-stdlib_3.8.10-0ubuntu1-20.04.8_amd64.deb ...
Unpacking libpython3.8-stdlib:amd64 (3.8.10-0ubuntu1-20.04.8) over (3.8.10-0ubuntu1-20.04.6) ...
Preparing to unpack .../03-python3.8-minimal_3.8.10-0ubuntu1-20.04.8_amd64.deb ...
Unpacking python3.8-minimal (3.8.10-0ubuntu1-20.04.8) over (3.8.10-0ubuntu1-20.04.6) ...
Preparing to unpack .../04-libpython3.8-minimal_3.8.10-0ubuntu1-20.04.8_amd64.deb ...
Unpacking libpython3.8-minimal:amd64 (3.8.10-0ubuntu1-20.04.8) over (3.8.10-0ubuntu1-20.04.6) ...
Selecting previously unselected package libstdc++-9-dev:amd64.
Preparing to unpack .../05-libstdc++-9-dev_9.4.0-1ubuntu1-20.04.1_amd64.deb ...
Unpacking libstdc++-9-dev:amd64 (9.4.0-1ubuntu1-20.04.1) ...
Selecting previously unselected package g++-9.
Preparing to unpack .../06-g++-9_9.4.0-1ubuntu1-20.04.1_amd64.deb ...
Unpacking g++-9 (9.4.0-1ubuntu1-20.04.1) ...
Selecting previously unselected package g++.
Preparing to unpack .../07-g++_4K3a9.3.0-1ubuntu2_amd64.deb ...
Unpacking g++ (4:9.3.0-1ubuntu2) ...
Selecting previously unselected package dpkg-dev.
Preparing to unpack .../08-dpkg-dev_1.19.7ubuntu3.2_all.deb ...
Unpacking dpkg-dev (1.19.7ubuntu3.2) ...
Selecting previously unselected package build-essential.
Preparing to unpack .../09-build-essential_12.8ubuntu1.1_amd64.deb ...
Unpacking build-essential (12.8ubuntu1.1) ...
Selecting previously unselected package libfakeroot:amd64.
Preparing to unpack .../10-libfakeroot_1.24-1_amd64.deb ...
Unpacking libfakeroot:amd64 (1.24-1) ...
Selecting previously unselected package fakeroot.
Preparing to unpack .../11-fakeroot_1.24-1_amd64.deb ...
Unpacking fakeroot (1.24-1) ...
Selecting previously unselected package libalgorithm-diff-perl.
Preparing to unpack .../12-libalgorithm-diff-perl_1.19.03-2_all.deb ...
Unpacking libalgorithm-diff-perl (1.19.03-2) ...
Selecting previously unselected package libalgorithm-diff-xs-perl.
Preparing to unpack .../13-libalgorithm-diff-xs-perl_0.04-6_amd64.deb ...
Unpacking libalgorithm-diff-xs-perl (0.04-6) ...
Selecting previously unselected package libalgorithm-merge-perl.
Preparing to unpack .../14-libalgorithm-merge-perl_0.08-3_all.deb ...
Unpacking libalgorithm-merge-perl (0.08-3) ...
Selecting previously unselected package libexpat1-dev:amd64.
Preparing to unpack .../15-libexpat1-dev_2.2.9-1ubuntu0.6_amd64.deb ...
Unpacking libexpat1-dev:amd64 (2.2.9-1ubuntu0.6) ...
Selecting previously unselected package libpython3.8-dev:amd64.
Preparing to unpack .../16-libpython3.8-dev_3.8.10-0ubuntu1-20.04.8_amd64.deb ...
Unpacking libpython3.8-dev:amd64 (3.8.10-0ubuntu1-20.04.8) ...
Selecting previously unselected package libpython3-dev:amd64.
Preparing to unpack .../17-libpython3-dev_3.8.2-0ubuntu2_amd64.deb ...
Unpacking libpython3-dev:amd64 (3.8.2-0ubuntu2) ...
Selecting previously unselected package python-pip-whl.
Preparing to unpack .../18-python-pip-whl_20.0.2-Subunit9.9_all.deb ...
Unpacking python-pip-whl (20.0.2-Subunit9.9) ...
Preparing to unpack .../19-python-pip-whl_20.0.2-Subunit9.9_all.deb ...
Unpacking python-pip-whl (20.0.2-Subunit9.9) ...
```

Appendix 4: Mininet installation screenshots

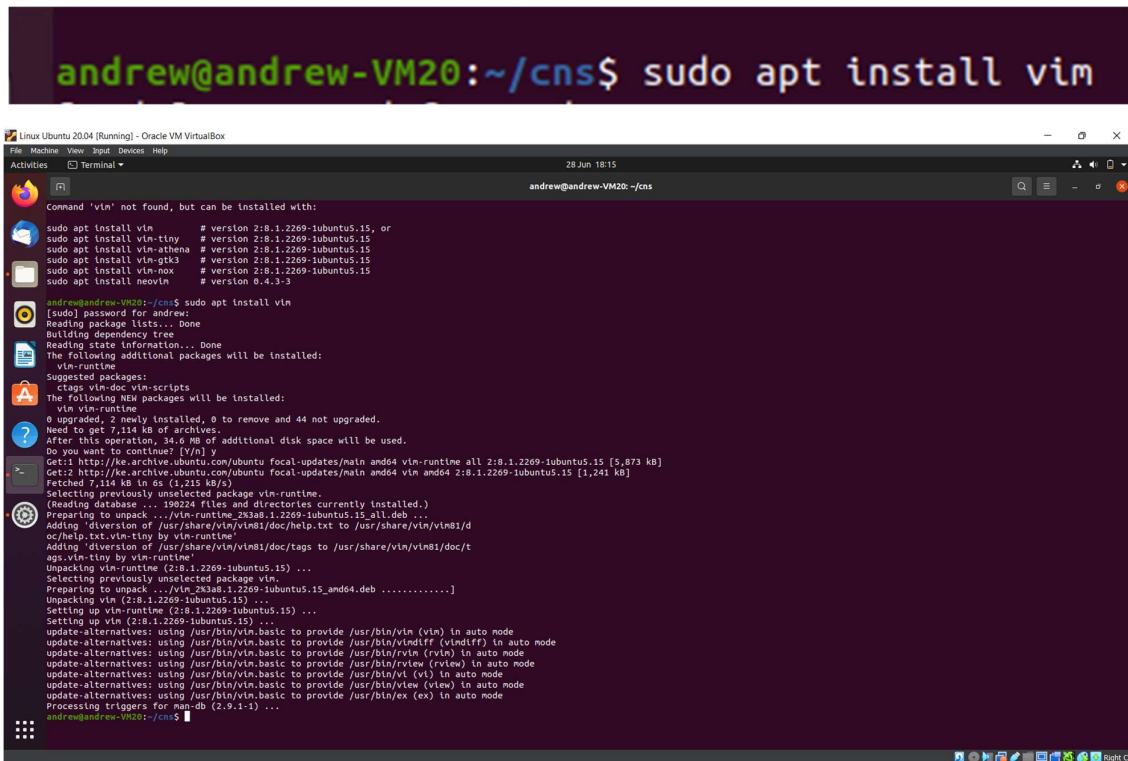
```
andrew@andrew-VM20:/media/andrew/VBox_GAs_7.0.8$ sudo apt install mininet

Linux Ubuntu 20.04 [Running] - Oracle VM VirtualBox
7 Jun 23:54
andrew@andrew-VM20:/media/andrew/VBox_GAs_7.0.8$ sudo apt install mininet
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  cgroup-tools iperf libcgroup1 libpython2.7-minimal libpython2.7-stdlib
  libunbound8 net-tools openssl-switch-common openssl-switch python-pkg-resources python2
  python2-minimal python2.7 python2.7-minimal python3-openssl python3-sortedcontainers
  socat
Suggested packages:
  openssl-doc python-setuptools python2-doc python-tk python2.7-doc binfmt-support
  python-sortedcontainers-doc
The following NEW packages will be installed:
  cgroup-tools iperf libcgroup1 libpython2.7-minimal libpython2.7-stdlib
  libunbound8 mininet net-tools openssl-switch-common openssl-switch python-pkg-resources
  python2 python2-minimal python2.7 python2.7-minimal python3-openssl python3-sortedcontainers
  socat
0 upgraded, 19 newly installed, 0 to remove and 180 not upgraded.
Need to get 7,940 kB of archives.
After this operation, 35.1 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ke.archive.ubuntu.com/ubuntu focal-updates/universe amd64 libpython2.7-minimal amd64 2.7.18-1-20.04.3 [336 kB]
Get:2 http://ke.archive.ubuntu.com/ubuntu focal-updates/universe amd64 python2.7-minimal amd64 2.7.18-1-20.04.3 [1,280 kB]
Get:3 http://ke.archive.ubuntu.com/ubuntu focal/universe amd64 python2-minimal amd64 2.7.17-2ubuntu4 [27.5 kB]
Get:4 http://ke.archive.ubuntu.com/ubuntu focal-updates/universe amd64 libpython2.7-stdlib amd64 2.7.18-1-20.04.3 [1,888 kB]
Get:5 http://ke.archive.ubuntu.com/ubuntu focal-updates/universe amd64 python2.7 amd64 2.7.18-1-20.04.3 [248 kB]
Get:6 http://ke.archive.ubuntu.com/ubuntu focal-updates/main amd64 libpython2-stdlib amd64 2.7.17-2ubuntu4 [7,072 B]
Get:7 http://ke.archive.ubuntu.com/ubuntu focal/universe amd64 python2 amd64 2.7.17-2ubuntu4 [26.5 kB]
Get:8 http://ke.archive.ubuntu.com/ubuntu focal-updates/main amd64 libcgroup1 amd64 0.41-10 [42.9 kB]
Get:9 http://ke.archive.ubuntu.com/ubuntu focal/universe amd64 cgroup-tools amd64 0.41-10 [66.2 kB]
Get:10 http://ke.archive.ubuntu.com/ubuntu focal/universe amd64 iperf amd64 2.0.13+dfsg1-1build1 [76.5 kB]
Get:11 http://ke.archive.ubuntu.com/ubuntu focal-updates/main amd64 libunbound8 amd64 1.9.4-2ubuntu1.4 [350 kB]
Get:12 http://ke.archive.ubuntu.com/ubuntu focal/main amd64 net-tools amd64 1.60-gt1018026-eeb88e-1ubuntu1 [196 kB]
Get:13 http://ke.archive.ubuntu.com/ubuntu focal-updates/universe amd64 python-pkg-resources all 44.0.0-2ubuntu0.1 [130 kB]
Get:14 http://ke.archive.ubuntu.com/ubuntu focal-updates/main amd64 python3-sortedcontainers all 2.1.0-2 [27.3 kB]
Get:15 http://ke.archive.ubuntu.com/ubuntu focal/main amd64 socat amd64 1.7.3-3-2 [123 kB]
Get:16 http://ke.archive.ubuntu.com/ubuntu focal/universe amd64 mininet amd64 2.2.2-2ubuntu1 [125 kB]
Get:17 http://ke.archive.ubuntu.com/ubuntu focal-updates/main amd64 openssl-switch-common amd64 2.13.8-0ubuntu1.2 [1,156 kB]
Get:18 http://ke.archive.ubuntu.com/ubuntu focal-updates/main amd64 python3-openssl all 2.1.1.0-2 [95.0 kB]
Get:19 http://ke.archive.ubuntu.com/ubuntu focal-updates/main amd64 openssl-switch-switch amd64 2.13.8-0ubuntu1.2 [1,540 kB]
Fetched 7,940 kB in 10s (641 kB/s)
Selecting previously unselected package libpython2.7-minimal:amd64.
(Reading database ... 186110 files and directories currently installed.)
Preparing to unpack .../0-libpython2.7-minimal-2.7.18-1-20.04.3_amd64.deb ...
Unpacking libpython2.7-minimal:amd64 (2.7.18-1-20.04.3) ...
Selecting previously unselected package python2.7-minimal.
Preparing to unpack .../1-python2.7-minimal-2.7.18-1-20.04.3_amd64.deb ...
Unpacking python2.7-minimal (2.7.18-1-20.04.3) ...
Selecting previously unselected package python2-minimal.
Preparing to unpack .../2-python2-minimal-2.7.17-2ubuntu4_amd64.deb ...
Unpacking python2-minimal (2.7.17-2ubuntu4) ...
Preparing to unpack .../3-python2-minimal-2.7.17-2ubuntu4_amd64.deb ...
Unpacking python2-minimal (2.7.17-2ubuntu4) ...
Selecting previously unselected package libpython2.7-stdlib:amd64.
Preparing to unpack .../4-libpython2.7-stdlib-2.7.18-1-20.04.3_amd64.deb ...
Unpacking libpython2.7-stdlib:amd64 (2.7.18-1-20.04.3) ...
Selecting previously unselected package python2.7.
Preparing to unpack .../5-python2.7-2.7.18-1-20.04.3_amd64.deb ...
Unpacking python2.7 (2.7.18-1-20.04.3) ...
Selecting previously unselected package libpython2-stdlib:amd64.
Preparing to unpack .../6-libpython2-stdlib-2.7.17-2ubuntu4_amd64.deb ...
Unpacking libpython2-stdlib:amd64 (2.7.17-2ubuntu4) ...
Setting up python2.7-minimal (2.7.18-1-20.04.3) ...
Setting up python2-minimal (2.7.17-2ubuntu4) ...
Linking and byte-compiling packages for runtime python2.7...
Setting up python2-minimal (2.7.17-2ubuntu4) ...
Selecting previously unselected package python2.
(Reading database ... 186857 files and directories currently installed.)
Preparing to unpack .../8-python2-2.7.17-2ubuntu4_amd64.deb ...
Unpacking python2 (2.7.17-2ubuntu4) ...
Selecting previously unselected package libcgroup1:amd64.
Preparing to unpack .../01-libcgroup1-0.41-10_amd64.deb ...
Unpacking libcgroup1:amd64 (0.41-10) ...
Selecting previously unselected package cgroup-tools.
Preparing to unpack .../02-cgroup-tools-0.41-10_amd64.deb ...
Unpacking cgroup-tools (0.41-10) ...
Selecting previously unselected package iperf.
Preparing to unpack .../03-iperf-2.0.13+dfsg1-1build1_amd64.deb ...
Unpacking iperf (2.0.13+dfsg1-1build1) ...
Selecting previously unselected package libunbound8:amd64.
Preparing to unpack .../04-libunbound8-1.9.4-2ubuntu1.4_amd64.deb ...
Unpacking libunbound8:amd64 (1.9.4-2ubuntu1.4) ...
Selecting previously unselected package net-tools.
Preparing to unpack .../05-net-tools-1.60-gt1018026-eeb88e-1ubuntu1_amd64.deb ...
Unpacking net-tools (1.60-gt1018026-eeb88e-1ubuntu1) ...
Selecting previously unselected package python-pkg-resources.
Preparing to unpack .../06-python-pkg-resources-44.0.0-2ubuntu0.1_all.deb ...
Unpacking python-pkg-resources (44.0.0-2ubuntu0.1) ...
Selecting previously unselected package python3-sortedcontainers.
Preparing to unpack .../07-python3-sortedcontainers-2.1.0-2_all.deb ...
Unpacking python3-sortedcontainers (2.1.0-2) ...
Selecting previously unselected package socat.
Preparing to unpack .../08-socat-1.7.3-3-2_amd64.deb ...
Unpacking socat (1.7.3-3-2) ...
Selecting previously unselected package mininet.
Preparing to unpack .../09-mininet-2.2.2-2ubuntu1_amd64.deb ...
Unpacking mininet (2.2.2-2ubuntu1) ...
Selecting previously unselected package openssl-switch-common.
Preparing to unpack .../10-openssl-switch-common-2.13.8-0ubuntu1.2_amd64.deb ...
Unpacking openssl-switch-common (2.13.8-0ubuntu1.2) ...
Selecting previously unselected package python3-openssl.
Preparing to unpack .../11-python3-openssl-2.1.1.0-2_all.deb ...
Unpacking python3-openssl (2.1.1.0-2) ...
```


Appendix 5: POX installation screenshots



Appendix 6: Vim installation screenshots



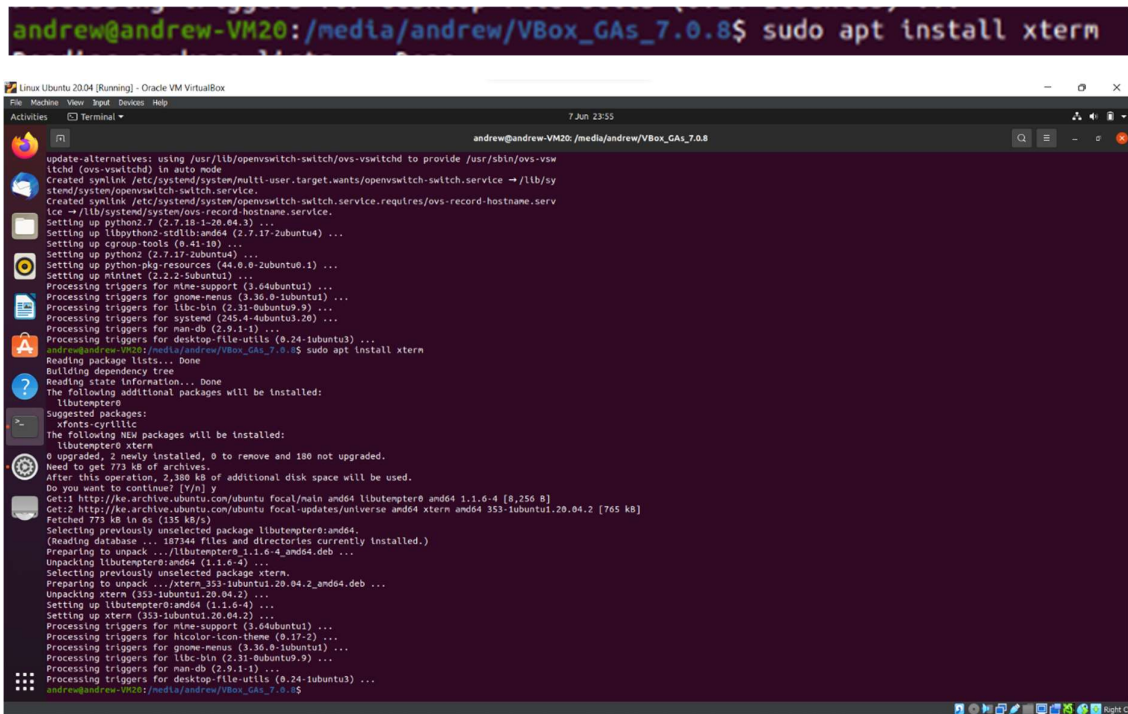
```
andrew@andrew-VM20:~/cns$ sudo apt install vim

Command 'vim' not found, but can be installed with:
sudo apt install vim          # version 2:8.1.2269-1ubuntu5.15, or
sudo apt install vim-tiny     # version 2:8.1.2269-1ubuntu5.15
sudo apt install vim-athena   # version 2:8.1.2269-1ubuntu5.15
sudo apt install vim-gtk3     # version 2:8.1.2269-1ubuntu5.15
sudo apt install vim-nov      # version 2:8.1.2269-1ubuntu5.15
sudo apt install neovim       # version 0.4.3-3

andrew@andrew-VM20:~/cns$ sudo apt install vim
[sudo] password for andrew:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  vim-runtime
Suggested packages:
  ctags vim-doc vim-scripts
The following NEW packages will be installed:
  vim vim-runtime
0 upgraded, 2 newly installed, 0 to remove and 44 not upgraded.
Need to get 7,114 kB of archives.
After this operation, 34.6 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ke.archive.ubuntu.com/ubuntu focal-updates/main amd64 vim-runtime all 2:8.1.2269-1ubuntu5.15 [5,873 kB]
Get:2 http://ke.archive.ubuntu.com/ubuntu focal-updates/main amd64 vim amd64 2:8.1.2269-1ubuntu5.15 [1,241 kB]
Fetched 7,114 kB in 1s (1,215 kB/s)
Selecting previously unselected package vim-runtime.
(Reading database ... 190224 files and directories currently installed.)
Preparing to unpack .../vim-runtime_2:8.1.2269-1ubuntu5.15_all.deb ...
Adding 'diversion of /usr/share/vim/vim81/doc/help.txt to /usr/share/vim/vim81/d
oc/help.txt.vim-tiny by vim-runtime'
Adding 'diversion of /usr/share/vim/vim81/doc/tags to /usr/share/vim/vim81/doc/t
ags.vim-tiny by vim-runtime'
Unpacking vim-runtime (2:8.1.2269-1ubuntu5.15) ...
Selecting previously unselected package vim.
Preparing to unpack .../vim_2:8.1.2269-1ubuntu5.15_amd64.deb ...
Unpacking vim (2:8.1.2269-1ubuntu5.15) ...
Setting up vim-runtime (2:8.1.2269-1ubuntu5.15) ...
Setting up vim (2:8.1.2269-1ubuntu5.15) ...
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/vim (vim) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/vimdiff (vimdiff) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/rvim (rvim) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/rview (rview) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/vi (vi) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/view (view) in auto mode
update-alternatives: using /usr/bin/vim.basic to provide /usr/bin/ex (ex) in auto mode
Processing triggers for man-db (2.9.1-1) ...

andrew@andrew-VM20:~/cns$
```

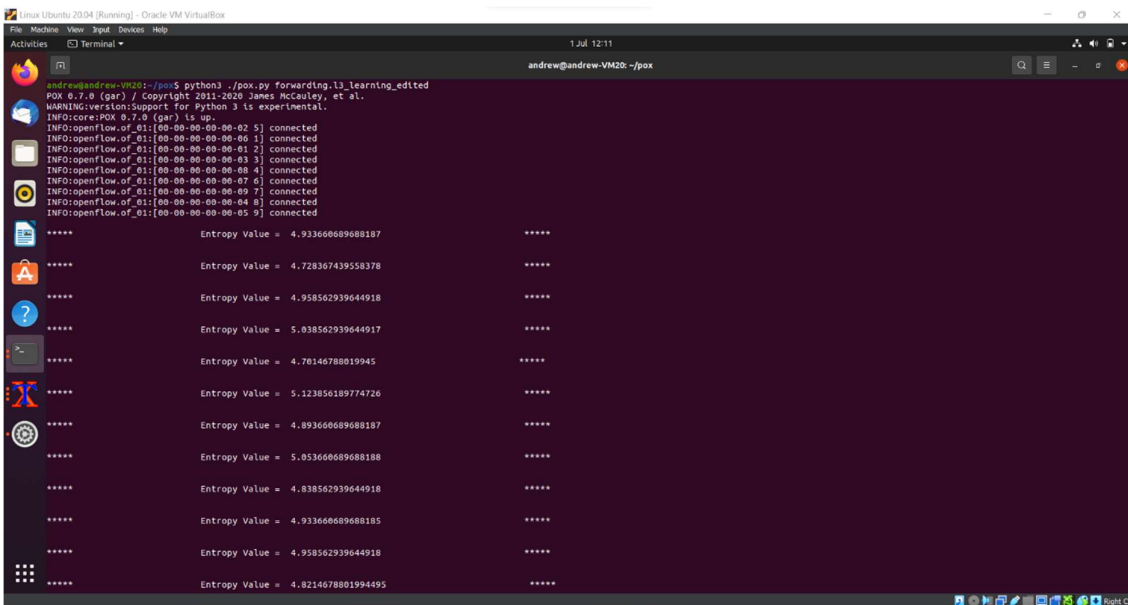
Appendix 7: XTerm installation screenshots



The screenshot shows a terminal window in a VM titled 'Linux Ubuntu 20.04 [Running] - Oracle VM VirtualBox'. The user is at the prompt 'andrew@andrew-VM20:/media/andrew/VBox_GAs_7.0.8\$' and has entered the command 'sudo apt install xterm'. The terminal output shows the package manager's process of installing 'xterm' and 'libutempter0'. It lists various dependencies, the disk space requirements, and the progress of downloading and unpacking the packages. The installation is successful, and the prompt returns to the user.

```
andrew@andrew-VM20:/media/andrew/VBox_GAs_7.0.8$ sudo apt install xterm
update-alternatives: using /usr/lib/openswitch-switch/ovs-vsitchd to provide /usr/sbin/ovs-vs
itd (ovs-vsitchd) in auto mode
Created symlink /etc/systemd/system/multi-user.target.wants/openswitch-switch.service → /lib/sy
stemd/system/openswitch-switch.service.
Created symlink /etc/systemd/system/openswitch-switch.service.requires/ovs-record-hostname.serv
ice → /lib/systemd/system/ovs-record-hostname.service.
Setting up python3.7 (2.7.18-120.64.3) ...
Setting up libpython2.7-stdlib:amd64 (2.7.17-2ubuntu4) ...
Setting up python2.7 (2.7.17-2ubuntu4) ...
Setting up python-pkg-resources (44.0.0-2ubuntu1) ...
Setting up mininet (2.2.2-Subuntul) ...
Processing triggers for mline-support (3.64ubuntu1) ...
Processing triggers for gnome-menus (3.36.0-1ubuntu1) ...
Processing triggers for libc-bin (2.31-0ubuntu9) ...
Processing triggers for systemd (245.4-4ubuntu3.20) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for desktop-file-utils (0.24-1ubuntu3) ...
andrew@andrew-VM20:/media/andrew/VBox_GAs_7.0.8$ sudo apt install xterm
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libutempter0
Suggested packages:
  xfonts-cyrillic
The following NEW packages will be installed:
  libutempter0 xterm
0 upgraded, 2 newly installed, 0 to remove and 180 not upgraded.
Need to get 773 kB of archives.
After this operation, 2,380 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://ke.archive.ubuntu.com/ubuntu focal/main amd64 libutempter0 amd64 1.1.6-4 [8,256 B]
Get:2 http://ke.archive.ubuntu.com/ubuntu focal-updates/universe amd64 xterm amd64 353-1ubuntu1.20.04.2 [765 kB]
Fetched 773 kB in 6s (135 kB/s)
Selecting previously unselected package libutempter0:amd64.
(Reading database ... 19746 files and directories currently installed.)
Preparing to unpack .../libutempter0_1.1.6-4_amd64.deb ...
Unpacking libutempter0:amd64 (1.1.6-4) ...
Selecting previously unselected package xterm.
Preparing to unpack .../xterm_353-1ubuntu1.20.04.2_amd64.deb ...
Unpacking xterm (353-1ubuntu1.20.04.2) ...
Setting up libutempter0:amd64 (1.1.6-4) ...
Setting up xterm (353-1ubuntu1.20.04.2) ...
Processing triggers for nine-support (3.64ubuntu1) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for gnome-menus (3.36.0-1ubuntu1) ...
Processing triggers for libc-bin (2.31-0ubuntu9) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for desktop-file-utils (0.24-1ubuntu3) ...
andrew@andrew-VM20:/media/andrew/VBox_GAs_7.0.8$
```

Appendix 8: Functional Requirement 1 testing screenshot

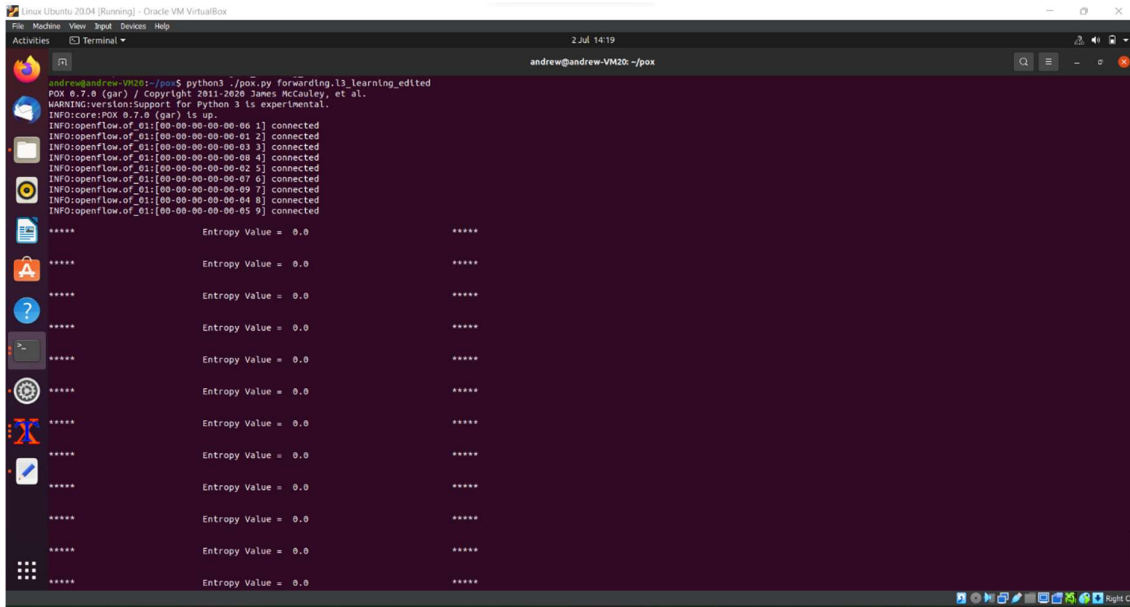


The screenshot shows a terminal window in a VM titled 'Linux Ubuntu 20.04 [Running] - Oracle VM VirtualBox'. The user is at the prompt 'andrew@andrew-VM20:~/pox\$' and has entered the command 'python3 ./pox.py forwarding_13_learning_edited'. The terminal output shows the POX 0.7.0 (gar) version, a warning about Python 3 support, and a list of connected openflow endpoints. Below this, a series of 'Entropy Value' calculations are displayed, each followed by a line of asterisks.

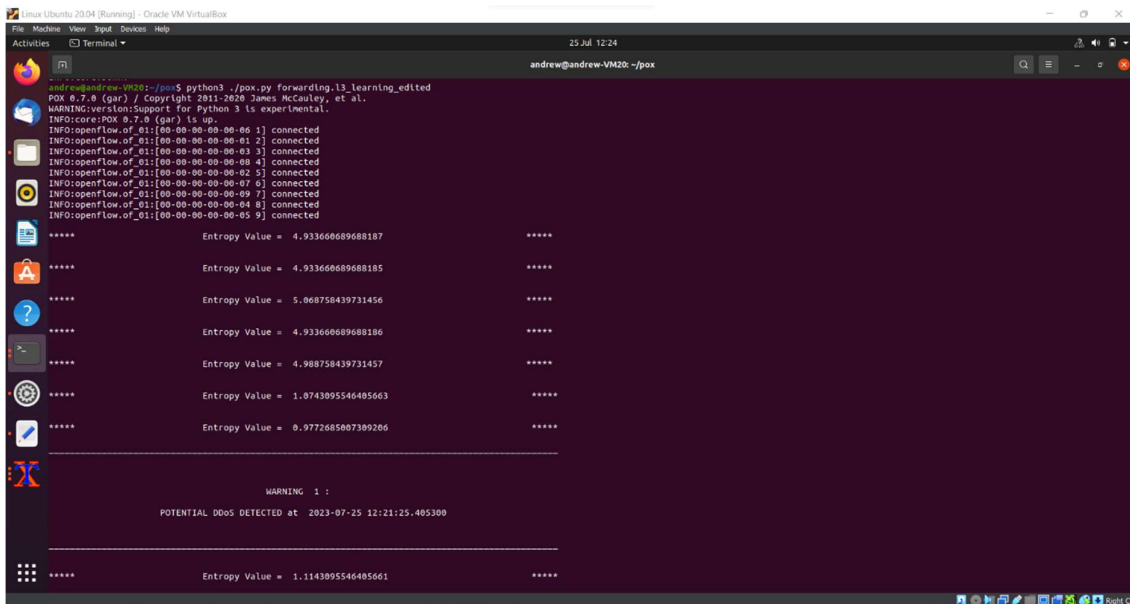
```
andrew@andrew-VM20:~/pox$ python3 ./pox.py forwarding_13_learning_edited
POX 0.7.0 (gar) / Copyright 2011-2020 James McCauley, et al.
WARNING:version:Support for Python 3 is experimental.
INFO:core:POX 0.7.0 (gar) is up.
INFO:openflow.of_01:[00-00-00-00-00-02 5] connected
INFO:openflow.of_01:[00-00-00-00-00-06 1] connected
INFO:openflow.of_01:[00-00-00-00-00-01 2] connected
INFO:openflow.of_01:[00-00-00-00-00-03 3] connected
INFO:openflow.of_01:[00-00-00-00-00-08 4] connected
INFO:openflow.of_01:[00-00-00-00-00-07 6] connected
INFO:openflow.of_01:[00-00-00-00-00-09 7] connected
INFO:openflow.of_01:[00-00-00-00-00-04 8] connected
INFO:openflow.of_01:[00-00-00-00-00-05 9] connected

*****
Entropy Value = 4.93366689688187 *****
*****
Entropy Value = 4.728367439558378 *****
*****
Entropy Value = 4.958562939644918 *****
*****
Entropy Value = 5.038562939644917 *****
*****
Entropy Value = 4.76146788019945 *****
*****
Entropy Value = 5.123856189774726 *****
*****
Entropy Value = 4.89366689688187 *****
*****
Entropy Value = 5.05366689688188 *****
*****
Entropy Value = 4.838562939644918 *****
*****
Entropy Value = 4.93366689688185 *****
*****
Entropy Value = 4.958562939644918 *****
*****
Entropy Value = 4.8214678801994495 *****
```

Appendix 9: Functional Requirement 2 testing screenshot



Appendix 10: Functional Requirement 3 testing screenshot



Appendix 11: Functional Requirement 4 testing screenshot

The screenshot shows a terminal window titled "Linux Ubuntu 20.04 [Running] - Oracle VM VirtualBox". The terminal output displays a series of entropy calculations and DooS detection warnings. The output is as follows:

```
*****
Entropy Value = 0.9772685807309206
*****

WARNING 3 :
POTENTIAL DooS DETECTED at 2023-07-25 12:21:26.340614

*****
Entropy Value = 1.1143095546405661
*****

WARNING 4 :
POTENTIAL DooS DETECTED at 2023-07-25 12:21:26.784859

*****
Entropy Value = 0.9772685807309206
*****

ALERT!!!
DooS DETECTED at 2023-07-25 12:21:27.235559
TARGETED AT 10.0.0.64
```


Strathmore University
School of Computing and Engineering Sciences
Information Systems Project Documentation Assessment Guide

Student Number: 138014

Project Title: DDoS attack Detection Model based on Entropy Computing and Software Defined Networking

Evaluation Points						Weight	Score	Notes
Title-(informative, concise, focused and appropriate?)						2		
Abstract Updated to include chapter 1-6						3		
Chapter 1-3 *Checking previous proposal chapters for the correctness of title and problem statement, project scope as implemented and change of tenses								
Problem Statement						1		
Justification- value addition						2		
Scope, Limitations and Delimitations						2		
Literature Review (Theoretical, Empirical, and conceptual (2))						4		
Methodology (Methodology, Tools, and Deliverables)						3		
Chapter 4 Correct functional requirements.						2		
Correct non-functional requirements.						2		
At least 4 analysis and Design diagrams and Discussion (extra mark for correct captioning)						5		
System Architecture and accompanying Discussion						2		
Chapter 5 Setup Description: Hardware, software, support libraries, frameworks/IDEs, versions, and compatibility						6		
Description of how the solution works to meet problem and business needs.						3		
Description of the test environment, data, test case						6		
Functional Requirement	Test Data	Expected Result	Actual Result	Pass/Fail	Evidence			
*Check 3 core functional requirements and evidence of test available as appendix						3		
Discussion of results and implications of the results								
Chapter 6 Valid Conclusion						2		
Sound Recommendation						2		
Valid Spin-offs for Future works						2		
Presentation Document Structure as per template provided and grammar.						2		
Citation and References						2		
Document Numbering and Table of Contents/figures						2		
Existence of required appendices						2		
Total Marks						60		