FINAL YEAR PROJECT INTERIM REPORT

<APPROVED PROJECT ID>
<Enhanced Detection Mechanism To Detect DOS Attacks in the IPv6 Network>

<1221304026>
<FLAVIAN NAVIN WENCESLAS>

<BACHELOR OF COMPUTER SCIENCE
B.CS (HONS) CYBERSECURITY >

<June 2025>

<APPROVED PROJECT ID>
< Enhanced Detection Mechanism To Detect DOS Attacks in the IPv6 network.>

BY

<1221304026 AND FLAVIAN NAVIN WENCESLAS >

PROJECT INTERIM REPORT SUBMITTED IN PARTIAL FULFILMENT OF THE

REQUIREMENT FOR THE DEGREE OF

<BACHELOR OF COMPUTER SCIENCE
B.CS (HONS) CYBERSECURITY >

in the

Faculty of Computing and Informatics

MULTIMEDIA UNIVERSITY
MALAYSIA

<June 2025>

Copyright

Declaration

I hereby declare that the work has been done by myself, and no portion of the work contained in this report has been submitted in support of any application for any other degree or qualification of this or any other university or institution of learning.

Name of candidate: Flavian Navin Wenceslas

Faculty of Computing & Informatics

Multimedia University

Date: 27: 06: 2025

Acknowledgements

I would like to thank everyone who helped make this initiative a success.

I owe my deepest gratitude to my supervisor, Dr. Navaneethan C. Arjuman, for his invaluable advice, inspiration and continuous support. His experience and constructive help enabled me to do this job on my part at best.

I would also like to thank Multimedia University, for offering, the other facilities and resources, which made this research possible.

Last but not least, my family and friends deserve my biggest thank you for always supporting, encouraging, and blessing me. All these factors kept reminding me to keep going.

Abstract

The implementation of IPv6 has created additional vulnerabilities, such as Denial of Service (DoS) attacks, particularly in home network situations. This project presents an enhanced detection mechanism for identifying and mitigating DoS attacks in IPv6 networks using machine learning. By concentrating on reducing false positives and increasing scalability, the system improves accuracy and speed, providing a strong method to protecting IPv6 networks against DoS attacks.

Table of Contents

List of Figures

List of Abbreviations/Symbols

1	DOS	Denial of Service
2	DDOS	Distributed Denial of Service
3	IPv6	Internet Protocol Version 6
4	ICMPv6	Internet Control Message Protocol version 6
5	IDS	Intrusion detection system
6	NDP	Neighbor discovery protocol
7	KNN	K-nearest Neighbors
8	SeND	Secure Neighbor Discovery
9	TCP	Transmission Control Protocol
10	SVM	Support Vector Machine
11	GR-AD-KNN	Information Gain Ratio Average Distance K-Nearest Neighbor
12	TAD-KNN	Threshold Adaptive Density Based on K-Nearest Neighbor
13	HTTP	Hypertext Transfer Protocol
14	FTP	File Transfer Protocol
15	UDP	User Datagram Protocol
16	TCP	Transmission Control Protocol
17	NS	Neighbour Solicitation
18	AR	Address Resolution
19	DAD	Disclosure, Alteration, Denial
20	IID	Interface Identifier
21	WINFIR	Windows Firewall
22	DT	Decision Tree
23	FPR	False Positive Rate
24	JLab	Jupyter Lab
25	NDP	Neighbor Discovery Protocol
26	NH	New Host
27	AT	Attacker
28	HIDS	Host Intrusion Detection System
29	NIDS	Network based intrusion detection system
30	SVM	Support Vector Machine
31	ANN	Artificial Neural Network

Chapter 1: Introduction

1.1 Overview

With the rapid growth of the Internet and the exhaustion of IPv4 addresses, the world is rapidly adopting IPv6. Where the previous version was found wanting in many respects, the new one has none of these weaknesses due to improvements in design that were made specifically to address such problems. IPv6 has a much larger address space, vastly improved security capabilities and is more consistent with modern networking needs--but as adoption of it expands, so do the chances for cyber-attack threats. Denial of Service (DoS) attacks are particularly prevalent.

They usually aim to clog a network or service by sending in more traffic than it can handle, thus making the targeted resource for traditional users quite impossible to use. Here the ingenuity comes in. For new IPv6 networks are characterized by an ever-expanding range of header structures with extensions as well as large address space, all very difficult for traditional intrusion detection systems (IDS) to cope with because those were first designed on IPv4 values.

This project addresses the above limitations by creating an “Enhanced Detection Mechanism” that is specifically adapted to DoS attacks on IPv6 networks. It integrates Snort, a popular signature-based IDS, with custom rules optimized for IPv6 traffic patterns and enhances them by using machine learning models such as Support Vector Machine (SVM) to analyse classified behaviours. We also put in place a real-time Python-based monitoring system essential for notifying upon any abnormal activities that arise and trigger alerts just in case something terrible happens.

Unlike older IDS, which are not sensitive enough to detect DoS attacks at the most complex level that IPv6 presents, this combination of rule-based and intelligent-behavioural detection greatly enhances the accuracy rate (and is corresponding reduction in false positives). By tapping into characteristics unique to IPv6–its special protocol qualities–and integrating anomaly detection together with behaviour analysis, this kind of system not only improves its detection capabilities but also raises its overall network intensity.

Ultimately, this project works to build a more dynamic, secure detection framework that can deal effectively with the new vulnerabilities brought in by IPv6 environments. Network administrators and network-security professionals now have a useful tool.

1.2 Problem Statement

Take IPv6 as an example, they also can be left open for attackers; as such this has led to a major threat to the security of communication networks. With different structures for IPv4 and IPv6 comes new attack surfaces that can be exploited. Traditional intrusion detection systems (IDS) are designed specifically for the IPv4 based network. That configuration has caused them to fail at correctly identifying and dispensing with DoS attacks on an IPv6 network., and the signatures of these traditional IDS solutions are mostly designed for detection, causing a high degree of false positives and false negatives in signal output. It's hard to distinguish ordinary variations of IPv6 traffic from real attacks for this reason. Simultaneously, the increasing complexity of IPv6 traffic has posed real-time analysis and threat identification problems.

It is in direct response to this challenge that this project introduces the enhanced detection techniques which provide a combination of custom Snort rule method, machine learning-based anomaly detection and analysis on real-time traffic. Thus, by combining these three methods the detection of DoS attacks in IPv6 networks can become more accurate and efficient than before. This comprehensive solution enhances the security and robustness of IPv6 networks, decreasing the number of false alarms at the same time as increasing their detection capabilities.

1.3 Objectives

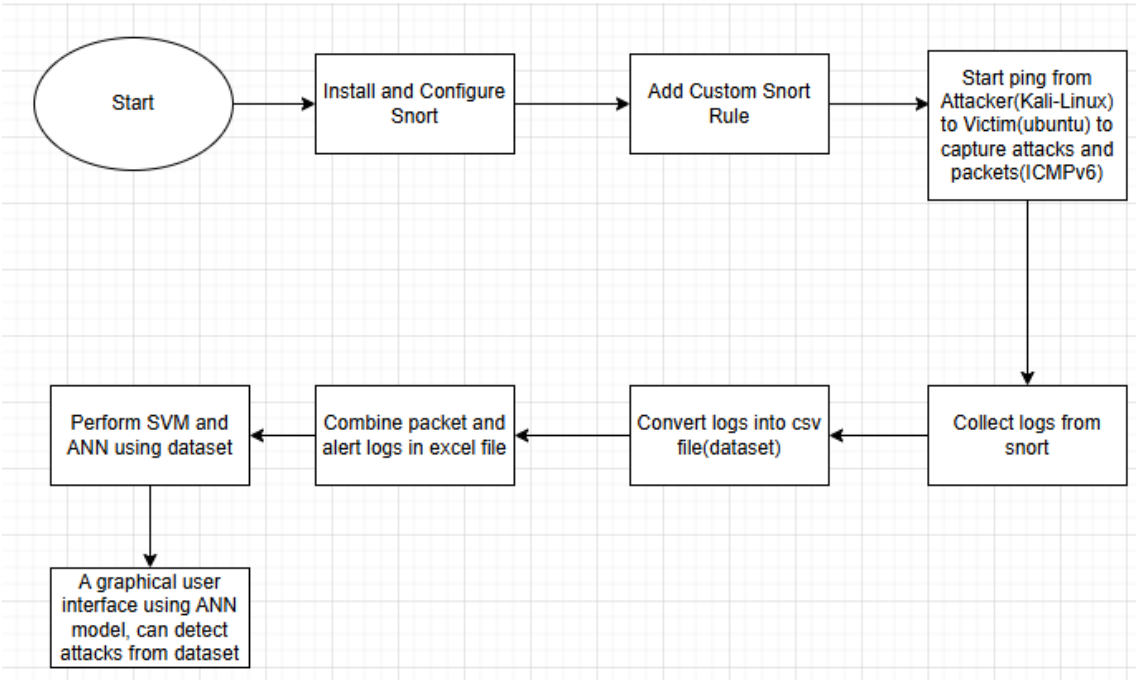
1) Create custom snort rules to accurately detect ICMPv6 Flood.
2) Develop an enhanced SVM and ANN model for DOS Detection in ICMP Packets
3) Optimize the SVM and ANN Model for improved Detection Accuracy and reduce False Positive and False Negative

1.4 Project Scope

This work aims to create a better way to capture and fend off DoS attack in the IPv6 world, especially the ICMPv6 flooding attack. The base concept of the system combines Snort-based rules detection with machine learning (SVM and ANN) in terms of increasing detection accuracy and minimizing false positives. A prototype was developed and examined in a controlled static IPv6 environment with virtual machines. Its performance was determined in terms of accuracy, false positive rate, and feature importance.

1.5 Project Plan

1.5.1 Flowchart



1.5.2 Gantt chart

Task	Week																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Configure Snort																	
Collect logs from Snort																	
Convert logs into csv file(dataset)																	
Perform SVM and ANN																	
Finalizing report																	
Report submission																	
Poster Demonstration																	

Chapter 2: Literature Review

2.1 Wireless Communications and Mobile Computing

Those classic intrusion detection models, for example, the Snort, depend on such rules detecting many types of Denial-of-Service (DoS) attacks and will meet the significant challenges if the amount and complexity of IPv6 network flow grows rapidly in the next few years before October 2023. This reliance on rules reduces the rate of detected attacks by IDS with rising proportion of IPv6 traffic. In IPv6, the expansion of its address space introduced new address bits, but it did not, by itself, solve network attack concerns, and so similar types of network attacks were still possible, such as Denial of Service (DoS) attacks. So, the detection methods have to be pushed ahead quickly, which can be confronted with large amount of traffic and more complexity when IPv6 networks are applied.

This emphasizes the insufficiency of existing detection techniques against dynamic network threats and the need for more effective approaches that match up with the specific traits of IPv6 traffic. The authors were moved by the explosion of the IPv6 traffic and the limitation of existing intrusion detection systems (Snort), which are based on predetermined rules that might cause degradation in detection performance. With the development of IPv6 networks, the complexity of network traffic within IP networks are increasing, so a new and effective detection method for DoS attacks is in urgent need.

Objectives:

Improve the detection of DoS attacks on IPv6 network, of which traditional IDSs cannot handle due to more traffic.

In order to alleviate the limitations of conventional KNN methods, especially for a small set of samples, the GR-AD-KNN (Information Gain Ratio Average Distance KNN) method is also introduced which contributes to the umbrella term of classification accuracy.

These tests demonstrate how much good is the proposed GR-AD-KNN algorithm compared to the quite traditional detection methods to increase the detection rates and reduce the misclassifications.

Methodology

The optimized KNN (K-Nearest Neighbours) method (as the GR-AD-KNN (GR: Information Gain Ratio—AD: Average Distance KNN) algorithm) is the main backbone of the proposed defence solution in the paper to recognize Denial-of-Service (DoS) attacks in the IPv6 networks. The key techniques involved in the process to increase the detection performance are:

The algorithm will employ the double dimensionality reduction with the information gain rate to prune and aggregate decisive features of network traffic data by which the overall dimensions of the features for achieving the best detection process becomes reduced for IPv6 traffic.

The rate of information gain is treated as different weights of the features, and even the fact that non-all the features are equally significant in the detection process by this algorithm is established. Here, the weighting serves for the algorithm to concentrate on the useful features of the data and thus to improve its performance with respect for the detection of DoS attack.

GR-AD-KNN improves the decision of KNN with Offset Increment Average Distance. This allows the algorithm to adjust based on the contribution of each sample point to classification by their respective distance to the point to classify. It also helps to minimize the “small group classification disadvantage” where there are not too many samples.

Results:

This part of the experiment is comparing the performance of GR-AD-KNN with traditional KNN algorithm (TAD-KNN) based on precision P and recall R (F1-Score). In average F1-Score ten experiments are performed and six experiments sets are planned to check how the number of k-values actually works. This is for the most part how the pre-processing and the dimensional reduction for the data were carried out and the performance evaluation of the GR-AD-KNN method was conducted as well as

the focus on where the enhancement in detection capability of the DoS attack detection in the IPv6 network could result from.

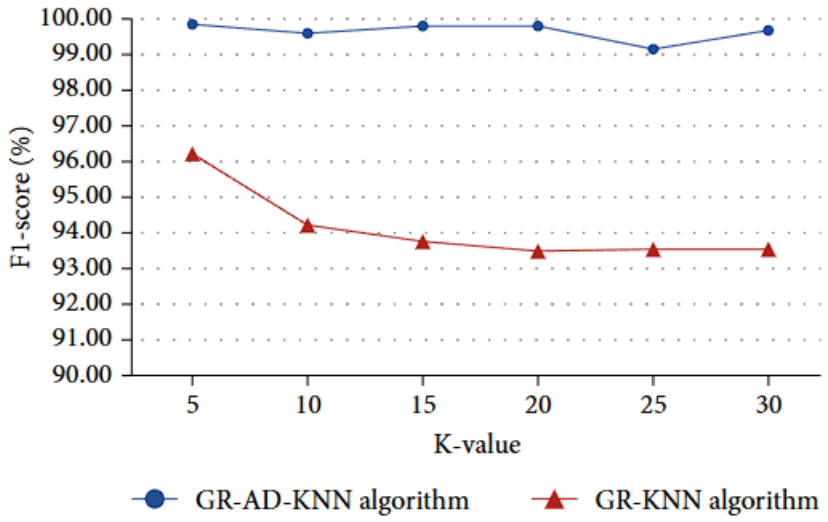
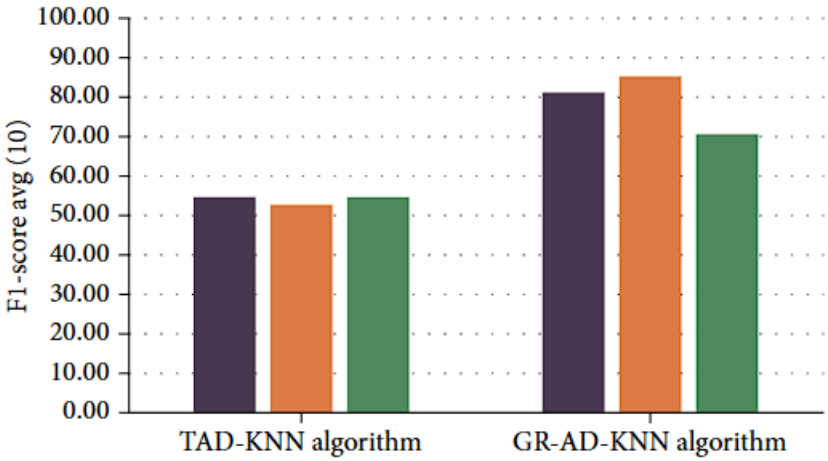


FIGURE 2: GR-KNN algorithm and GR-AD-KNN algorithm detection.



In this paper, we have compared these two methods experimentally; namely, GR-KNN and GR-AD-KNN were evaluated. You are trained on data generated from k = [6, 8, 15, 28, 30, 31] horizontal control experiments and record the experimental results. In addition, both datasets are normalized independently before the experiment (visual comparison of the performance of the algorithms). Detailed experimental results are shown in Figure 2. It can be seen from Table 2 that the classification result of the GR-AD-KNN was the optimal in the experimental classification. Furthermore, our proposed

method GR-AD-KNN is also less sensitive to k, leading to alleviate the encapsulation of the strictly requirement on k and mitigating the adverse effect on excessive sensitivity in tuning Step of the model. Conversely, in this paper, the averaged pattern-based TAD-KNN and GR-AD-KNN were introduced with detection performance perspective. For example, when kis 5, in each of three independent experiments, we performed all ten of these detection tests. Average F1-Score of 10 detection results of attack type Teardrop per game. By analysing the experimental results, we can find the ability to detect Teardrop type of attack through the two algorithms comparison. It can be found that the GR-AD-KNN algorithm has a better detection ability for the Teardrop type of attack, which means that the initial detection ability has been improved. Therefore, Improved KNN Algorithm for IPv6 based DoS intrusion detection has better classification accuracy and detection rate.

Advantages

The K-Nearest Neighbour (KNN) algorithm yields significant advantages in identifying the Denial-of-Service (DoS) attacks in IPv6 networks due to its lightweight classifying background that does not require pre-training and it is very suitable for real-time scenarios. The feature dimension is reduced by double dimensionality reduction with information gain rate, achieving the aim of enhancing the detection efficiency. Which also yields a way to score the critical features in pair, optimize for the operation of the Euclidean distance, and for bypassing the deficiency of classification performance when number of affected people is quite limited. The experimental findings of the proposed GR-AD-KNN variant also verify its power to make it more insusceptible to parameter selection, thereby improving the stability of detection, obtaining higher F1-scores, and becoming more efficient in the detection of small and hard to detect types of attacks. Because of its excellent performance, flexibility, and better precision, KNN is quite useful to identify the DoS attacks for IPv6.

2.2 Enhanced Windows Fuzzy firewall DOS attack prevention

This paper shows how Windows Firewall can be enhanced by Winguardian with respect to block Denial of Service (DoS) attacks. It demonstrates that the default Windows Firewall contains such types of susceptibilities to identify DoS attacks also for ICMP, UDP and TCP traffic, as well as HTTP and FTP packets.

Incorporating fuzzy logic as one of its main components, not only Winguardian offer a more intelligent way to protect your network it, can analyse the traffic pattern and automatically detect possible Denial of Service attack more efficiently! It's going to show how Winguardian was conceived, built, tested and made to run; day offering you better fault-remediation than that annoying old hardware firewalls are cantankerous software to this day you find lying around like greasy old sofas made of icky heuristics, even on that IPv6 you're not blocking, under your old crusty firewall policies.

To conclude, this paper highlights the necessity for enhanced firewall solutions about the present threats and the growing intricacy of network traffic in today’s world of cyber security.

The article makes the point that the built-in Windows firewall does not protect against Denial-of-Service (DoS)attacks. (3) More specifically, the generated code points to important directions with respect to the following:

DoS (denial of service) flood attacks than flood that cause its target to be unable to serve its legitimate users is not able to prevent such attacks from the network instead of attempts to is the standard Windows Firewall.

However, the examples of advanced DoS attacks like DDoS attacks, which are based on the use of a few hijacked devices, indicate a real threat depreciating the significance of segmenting a corporate network with traditional firewall systems.

It is high time for advanced firewalling techniques that can outperform old-fashioned firewalls, especially in IPv6-enabled environments, where they are exposed to DoS attacks, known and unknown.

Objectives of the work This work concentrates on applying the fuzzy logic in the design implementation of firewall to obtain better decisions which would comparatively be more effective in the dealing with various levels of threat include proper control of the HTTP and FTP traffic.

Objectives:

Improving the Features of the Firewall: Designing a successful firewall system which has better features compared to the in-built Windows filtering mechanism in detecting against several types of DOS/DDOS attacks the name of our firewall is called Winguardian.

Use of Fuzzy Logic: Implementation of event based Fuzzy logic in firewall for making make it more dependable to take decision on unknown behaviours of nature of network traffic and attack.

Multi-Protocol Supported: Winguardian would support other protocols including HTTP, FTP, TCP, UDP, and ICMP so that it could manage and guard other kind of attacks, this would make Winguardian implementable at different situations.

Methodology

The approach presents some critical aspects, for both the design, the implementation and the assessment of the improved Winguardian system. This can be accomplished by implementing a mechanism for identifying vulnerabilities in network protocols such as IPV6, HTTP and FTP based on the movement of the basic units in response to the system firewall to downsize DoS attack more effectively. A.Fuzzy Reasoning for A Firewall Fuzzy reasoning is to denote the degree of truth takes the value between 0 and 1 which is different from the traditional binary logic only take true or false, etc. What I mean is, that the rules you define to pass or deny network traffic and security that you implement in a firewall are not written in stone, like we see them in Figure 2. Fuzzy logic, however, allows the firewall to determine the degree of applicability of an individual condition or attribute so that an agent may better understand actionism, giving a more detailed and multi-faceted view of activity within the network. B.OverviewAllow or Block the Windows Firewall (Windows Firewall with Advanced security) type, allowing and blocking options for all and each types of network traffic,

and also "pfirewall. log" forehead-slapper that's fully exposed and available to the entire planet. 3.2 DoS attacks. Here the good explanation of three sorts: UDP, TCP and ICMP (ICMPv4 and ICMPv6) DoS attacks has been given. What's more fun is the user needs to boost up the below as it blocks the incoming in the "inbound rules" ICMP (ICMPv4, ICMPv6), UDP & TCP Traffic. And this is part of the "advanced security". The higher level of security also lets you configure inbound-firewall rules to block incoming HTTP and FTP requests at each end port - 80 and 21 - filtering by protocol, local and remote ports, and action (block/allow).

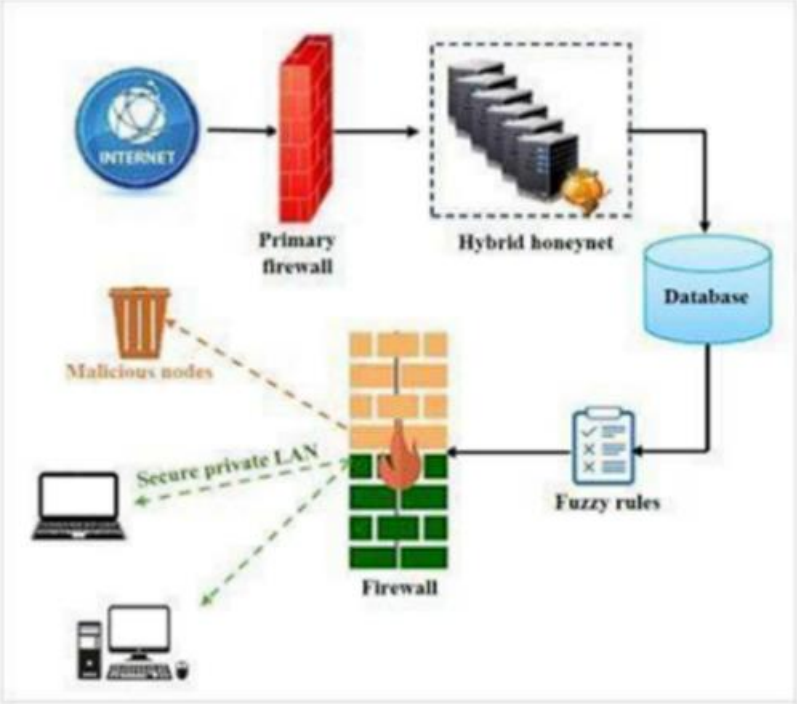


Fig.2 Firewall with fuzzy logic implementation [12]

System Design

Disallowing packet forwarding in out and in direction a user can be prevented from sending WHEREPP in order to send check is a click at HTTP the. For allow or deny HTTP and FTP packets to forward" (see "advanced security"). The drawback to this choice is that it filters HTTP and FTP packets upon power up and will not permit them in or out, depending on the conditions imposed by the rules. That would whitelist normal HTTP and FTP traffic over and above normal HTTP and FTP packets. The

bottom part is block against and these are the features: applicable profiles selected by the user with the possibility of which the user has to, via use LUT-MMP and a custom and net rule: a If the user request and by is valid target, part of the systems two based systems i counteracts any Does dropping You and user credential and an attacker ftp you root and HTTP and capture: Besides that the paper propose to improve a Winguardian using reasoning. The first is Windows firewall and fuzzy reasoning engine. The Windows firewall, installed by default, is capable of logging traffic to the “firewall. log", a plain text file. This log file is processed to extract some parameters such as average HTTP and FTP packet size, HTTP and FTP request rate, which are passed to the fuzzy reasoning system. It clarifies to Windows Firewall for distinguish HTTP & FTP DoS attack and there attack severity.

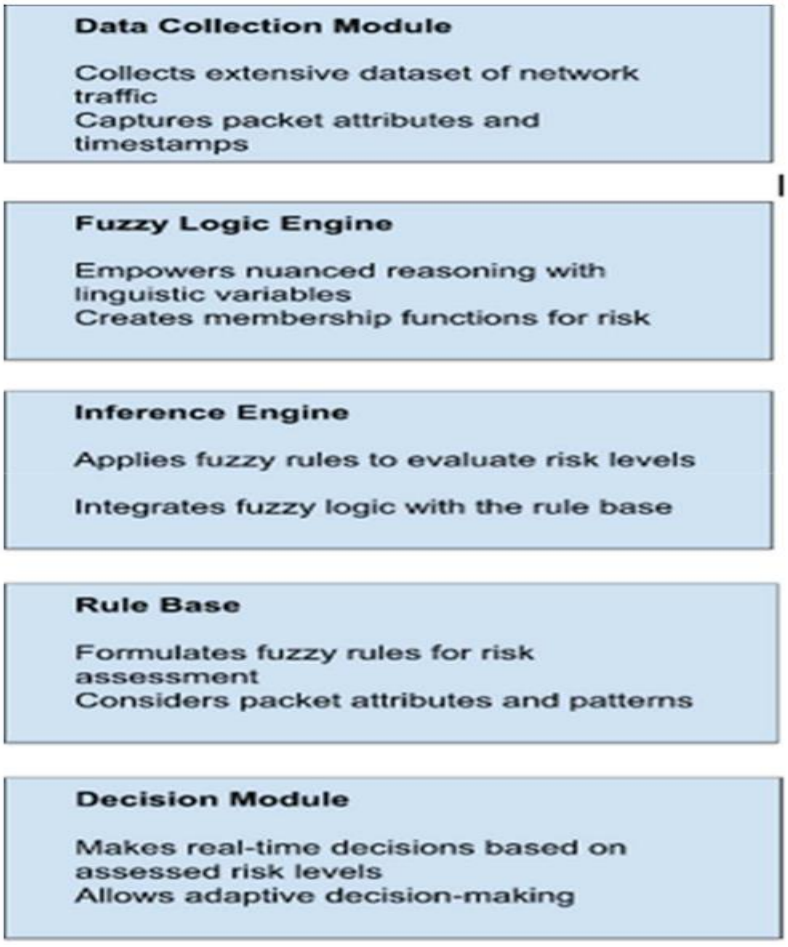


Fig 4 shows the systems architecture.

An Integrated and flexible Network Security systems Architecture: The design is then laid out for an integrated and flexible network security systems architecture in the paper. It includes dynamic decision making power, unconventional wisdom risk assessment, massive data collection, logging of transparence, and alerting that give notice to the insurance policy built against unacceptable circumstances. This is the objective of a fuzzy-based network security system (depicted in Figure 4) operating in the dynamic and interrelated world of cyber security, i.e., to improve the security of a network by introducing fuzzy logic in a much more complex system architecture.

Results:

The experiment results reveal that the attack of DoS & DDoS on both the HTTP and FTP packets is detected by the "Winguardian" as represented in Fig 5. This mileage covers the application of the flexible functioning of the fuzzy logic to understand/interpret the complexities, dynamicity and ambiguity of pattern of the network traffic and attacks. The brain of the architecture, thus "Winguardian Wall ", is made up of two key elements: a powerful Windows firewall and our fuzzy logic inference engine. Data is collected from pfirewall. text file for network logging. These include request frequencies and packet sizes for HTTP and FTP. In this, the system analyses possible attack degrees using fuzzy rules and membership functions and then suggests an appropriate action.

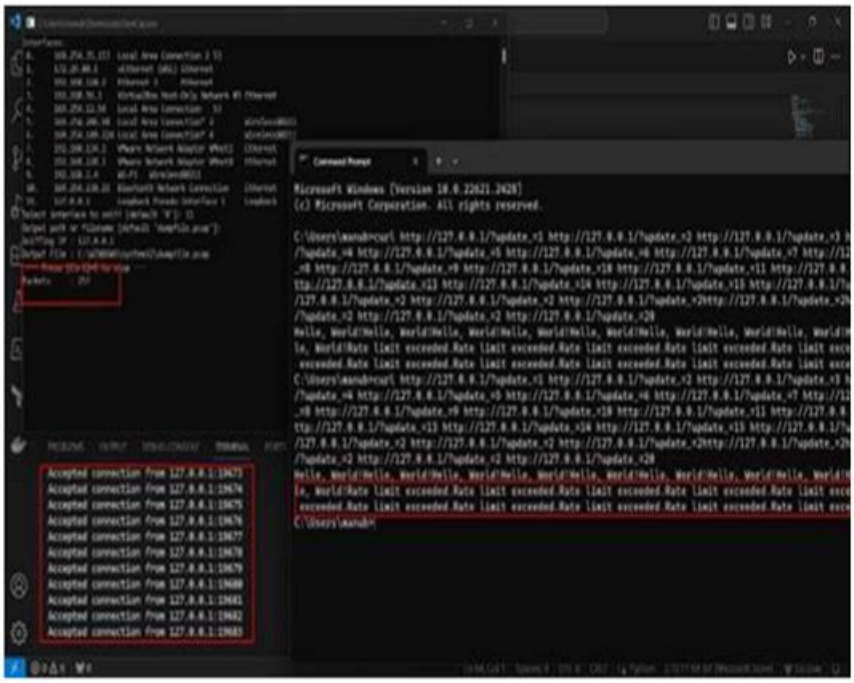


Fig.5 Demonstration of DOS attack blocking and showing error message, "Rate Limit Exceeded"

The method was tested on a simulated community context, in the course of which a variety of HTTP Supply and FTP Supply carried out a sequence of DoS and DDoS counteractions. Measures such as accuracy, false positive and false negative rates, and reaction time were used to assess the performance. The new Winguardian solution showed better accuracy, false positive and negative rates than the former solution, a Windows Transport Layer spread firewall called FR-Winfirewall, as well as a better reaction time. The results indicate that on the HTTP and FTP packets based DoS and DDoS attacks "Winguardian" is an effective remedy to defend a linear arrangement against both that run using the system. It enables the user to conveniently configure and tailor the solution as per user's challenge and likings. It likewise collaborates with Windows Firewall and can be expanded to explore extra kinds of dangers utilizing other conventions. It is available for a variety of platforms, including consoles as well as tablets and desktop computers. It is compatible with the newest Windows OS.

Advantages

It enables advanced analysis of incoming network data via fuzzy logic, adapting its choices and providing formidable defenses against new attack patterns. With support

for IPv4, IPv6, HTTP, and FTP protocols, it can protect networks from a wide range of attacks in a unified manner. Winguardian adds support for fuzzy logic over the standard Windows Firewall to increase detection speed. Simple to deploy and customize, extensible for new protocols and threats as they become available. It is Winguardian as its hybrid features provide versatility and intelligence against modern age cyber-attacks.

Conclusion

In this paper, we introduce a novel system referred to as "Winguardian": an Http and Ftp packet based technology to identify and alleviate DoS & DDoS attack at Windows Machines. A we do the data-based probabilistic model,whys(the uncertainties are described in fuzzy logic form for encoding unknow attack patterns,broad-based profiled network traffic. This solution employs fuzzy reasoning algorithm within a Windows Firewall. In this work, different kinds of network attacks such as Denial of Service (DoS) and a Distributed Denial of Service (DDoS) attacks using HTTP and FTP packets from different sources were performed over a network simulator. They evaluated the solution based on accuracy, false positive, false negative, and reaction time, and found that "Winguardian" is a more advanced version of the current solution, "FR-Winfirewall," which only works for detecting and preventing DoS and DDoS attacks through TCP, UDP, and ICMP packets. “The results also indicated that Winguardian is firewall profile and group agnostic and can filter both IPv4 and IPv6 traffic.

2.3 ICMPv6-Based DoS and DDoS Attacks Detection Using Machine Learning Techniques, Open Challenges, and Blockchain Applicability

IPv6 was created to solve address depletion problems caused by its predecessor protocol, IPv4, but it also gave rise to vulnerabilities like Internet control message Protocol version 6 (ICMPV6) and its features Neighbor Discovery Protocol.

It discusses at length about ICMPv6-based DoS attacks, which is one of the topics that raises great challenges for network security, as well as about the countermeasures.

Problem Statement

New types of DoS attack vulnerabilities have been created due to the introduction of IPv6 and related features, especially ICMPv6. These attacks attempt to render networks or hosts inaccessible to legitimate users by flooding them with malicious traffic. Most machine learning-based IDS mechanisms are also centralized in terms of traffic capturing process and data analysis. This centralization makes them vulnerable to single points of failure and contributes to latencies in attack detection and mitigation in confirmation window.

Objective

Survey Existing Studies: The main purpose is to survey the existing studies which are being conducted in detecting forms of ICMPv6-based DoS & DDoS attacks. This involves performing a thorough review of the current Intrusion Detection Systems (IDS) that employ machine learning methods.

Certain IDS Approaches: The paper seeks to classify various IDS approaches utilizing machine learning methods to recognize ICMPv6-based attacks. It serves to inform the reader of the classifiers employed, the feature selection techniques applied, as well as evaluation metrics employed by previous articles in this era.

Pinpointing Challenges: The authors aim to pinpoint and deliberate the existing challenges pertaining to intrusion detection systems for ICMPv6-based attacks. This involves knowing the state-of-the-art of current systems and when and how they fail, as well as the research challenges they present as follows.

Methodology

Through the literature review methodology, the article aims to analyze Intrusion Detection Systems (IDS) for ICMPv6-based DOS and DDOS attacks using embedded to machine learning techniques. It has classified the IDS techniques into single models and hybrid models and has explored algorithms such as Support Vector Machines and Artificial Neural Networks. The review provides both insight in endeavouring effective performance metrics with respect to Detection Rate (DR) and False Positive Rate(FPR),and opportunities through open issues which should be addressed in future research efforts as well as possible pathways using ensemble learning and blockchain

technology. Additional figures in the article including statistics on IPv6 users and a classification of IPv6 vulnerabilities provide a solid basis for the writing that follows regarding why effective detection mechanisms for the attacks are urgently needed.

Design

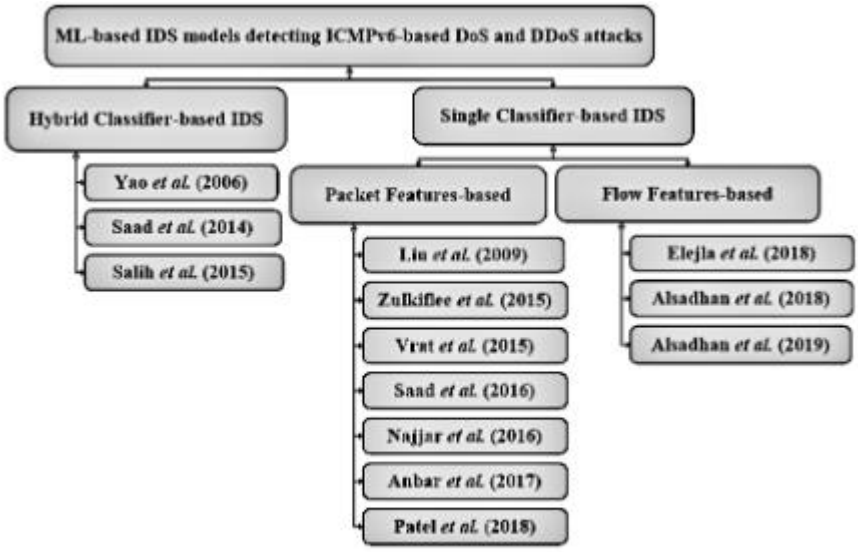


FIGURE 9. Classification of existing ML-based IDS models detecting ICMPv6-based DoS and DDoS attacks.

The classifications can be detected further as single classifier systems, which are used for a single machine learning method (for instance, Decision Trees (DT) or SVM), and hybrid systems, which use a mixture of approaches for a more accurate recognition. This hybrid can take the form of ensemble methods, training multiple models and combining outputs to improve overall performance.

This setup may make it more crucial to select features. For classifying ICMPv6 traffic, proper identification of relevant features of flow data is a prerequisite for effective intrusion detection systems (IDS). It could be preprocessing the traffic data, needed to parse the essential elements of the ICMPv6 packets, thus indicating benign activity or attack patterns.

All classifiers are trained on labelled datasets containing examples of normal and attack traffic. The model is trained on the given inputs using the extracted features to differentiate the two classes.

Following training, the models are then assessed for their detection capacity based with metrics like Detection Rate (DR) and False Positive Rate (FPR). This step is intentional in monitoring the performance of both models in a real-time environment.

And though different models and applications have been provided in Figure 9, the general architecture for detecting ICMPv6-based DoS and DDoS using the machine learning approach is normally consists of data collection, feature extraction, model classification, performance evaluation, and monitoring aspects.

Results

This paper discusses the effectiveness of different machine learning approach IDS for ICMPv6 based DoS and DDoS attack detection. Multiple studies exhibited impressive performance, with one reporting a detection rate of 85% and a false positive rate of 2% in controlled settings. Yet another example of a hybrid learning technique showed impressive precision, achieving an accuracy of 98.3% with an RMSE of 0.26, signifying its ability to accurately classify traffic. Some of the advanced methods, e.g. Locally Weighted Learning (LWL), were able to detect at a high rate (96.48% detection accuracy) and remarkably low false-positive rate (0.0004%). Nevertheless, the article further points out challenges existing models face, specifically in terms of feature selection methods: a crucial component for data preparation aimed at relevance. Also, the biases that can be introduced in datasets through adversarial attack simulations may limit real-world performance. The authors emphasize that due to the continuously progressive characteristics of DDoS attacks, models should be improved to keep up with them, and more research needs to be done on different detection tuples in this vector.

Advantage

Through the application of sophisticated algorithms capable of discerning intricate patterns within network traffic, these systems significantly improve detection accuracy, enabling them to differentiate between benign and malicious activities more effectively than traditional techniques

Furthermore, the real-time capability of these systems is a substantial benefit, enabling them to evaluate network traffic and act promptly against suspicious activities, thus reducing potential unattended operational hours and service unavailability from DoS attacks. Furthermore, the use of ensemble learning methods in a distributed setting leads to the development of more robust and smarter detection mechanisms that can help to reduce the likelihood of a single point of failure, which is a major weakness of traditional centralized detection methods.

2.4 Match-Prevention Technique Against Denial-of-Service attack on Adress Resolution and Duplicate Address Detection Processes in Ipv6 Link-Local Network

Neighbour Solicitation (NS) and Neighbour Advertisement (NA) messages play a key role in Address Resolution (AR) and Duplicate Address Detection (DAD) in IPv6's Neighbour Discovery Protocol (NDP). Whereas, NDP is not that much secured through it can be easily manipulated and Denial-of-Service (DoS) attacks can be possible. Two similar approaches (SeND (Secure NDP) and Trust-NDP (Trust-ND) are only partly successful but long time processing, high bandwith consumption and Denial of service (DoS) resistance 19[21]. This misuse could be addressed via Match-Prevention, which is a protection against target IP addresses as well as NS and NA exchanges. We evaluated the processing time and bandwidth utilization of Match-Prevention to investigate its performance potential, and progress that Match-Prevention can be added for resilience against these DoS attacks and can enhance the security provision of AR and DAD on IPv6 link-local networks.

Problem Statement:

As a result of the insecure design, the Address Resolution (AR) and Duplicate Address Detection (DAD) processes of the Neighbour Discovery Protocol (NDP) used in IPv6 networks are vulnerable to Denial-of-Service (DoS) attacks, as they have not been properly validated. For example, as one of the weakest attacks is node hostility in the same link-local network, which acts in such a way that the future of the node will be very easy for it to change the NDP messages (Neighbour Solicitation (NS) and Neighbour Advertisment (NA), in which value NDP messages can be changed during communication. Proposed existing Security approaches like Secure NDP (SeND) and Trust-NDP (Trust-ND) are marred by lengthy processing time and bandwidth constraints and susceptible to a DoS attack, hence cannot cope with these drawbacks.

Objective:

Cryptographic solution to prevent target ip address during Address Resolution(AR) and Duplicate Address Detection(DAD) in ipv6. Securing NDP Messages is to develop a method to secure neighbour solicitation and neighbour advertisement messages using

experimental NDP options. High Security is to provide a robust protection against attacks such as MITM, hash collision, IP spoofing, mitigating DOS threats to Address Resolution and Duplicate Address Detection.

Methodology:

The Match-Prevention approach protects Address Resolution and Duplicate Address Detection in IPV6 local networks via many phases. Hosts produce IP addresses with a privacy extension to increase security. The target host generates a random number, hashes it with the target IP's 64-bit Interface Identifier (IID), and then embeds the hash in the message's IP hash field. A match-option is added to standard Neighbour Solicitation and Neighbour Advertisement to create NS-match and NS-match messages. Recipients validate the match option, check the RIN field to avoid replay attacks, and compare hash values to confirm message authenticity. This comprises security analysis to safeguard the target IP and performance evaluation through scenarios, resulting in effective DoS attack prevention while maintaining network efficiency.

Design

Match Prevention Use such Marking/Protection Methods which are in Aling with current IPv6 network structure to ensure the NS and NA packets Identification by all of the IPv6 hosts in the chain; Does not need to change the architecture or add up a third party into it. The test-bed is comprised of six nodes, gateway router, switch, two existing hosts (EH_1, EH_2), new host (NH), and the attacker (AT). Figure 9 shows the experimental test-bed network topology used. Hardware and software specifications for the test-bed environment were selected according to the requirements of the experimentation and their availability in (the IPv6 environment of the National Advanced IPv6 Centre (Nav6) of University Sains Malaysia [37]), in order to effectively execute the experiments. (summarized in Table 5)

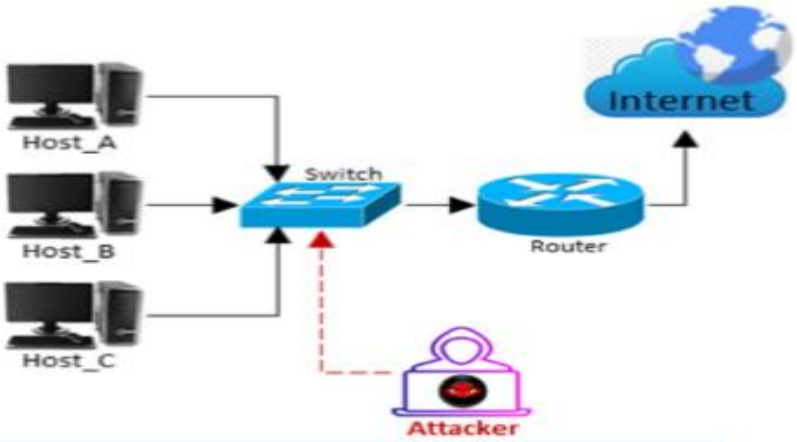


FIGURE 9. Test-bed setup environment.

TABLE 5. Hardware and software requirements for the test-bed environment.

Item Name	CPU	Memory	Operating System
Host_A	Intel(R) Core (TM)2 CPU Q8400 @ 2.66 GHz × 4	5.00 Gb	Windows 10 Pro
Host_B	Intel(R) Core (TM) i7-3770M CPU @ 3.40 GHz × 8	8.00 Gb	Ubuntu 16.04
Host_C	Intel(R) Core (TM) i7-2640M CPU @ 2. 30 GHz × 4	4.00 Gb	Ubuntu 16.04
Attacker	Intel(R) Core (TM) i7-2640M CPU @ 2.80 GHz × 4	4.00 Gb	Ubuntu 16.04

Item Name	Type
Switch	Cisco Catalyst 2960 Fast Ethernet
Router	Cisco Router C7200

Results:

Figure 10 compares total processing time for Address Resolution and Duplicate Adress Detection process across four techniques which are Standard Process, SeND, Trust-ND and Match Prevention.

SeND: Takes the longest processing time due to its computationally intensive design, averaging 147.39 ms for AR and 151.98 ms for DAD.

Trust-ND: Quicker than SeND, with 9.29 ms for AR and 10.98 ms for DAD, but still slower than Match-Prevention.

Match-Prevention: Demonstrates significant efficiency, with processing times of 6.63 ms for AR and 7.78 ms for DAD, achieving a balance of security and speed.

Standard-Process: The fastest but lacks any security measures, taking only 1.44 ms for AR and 1.86 ms for DAD.

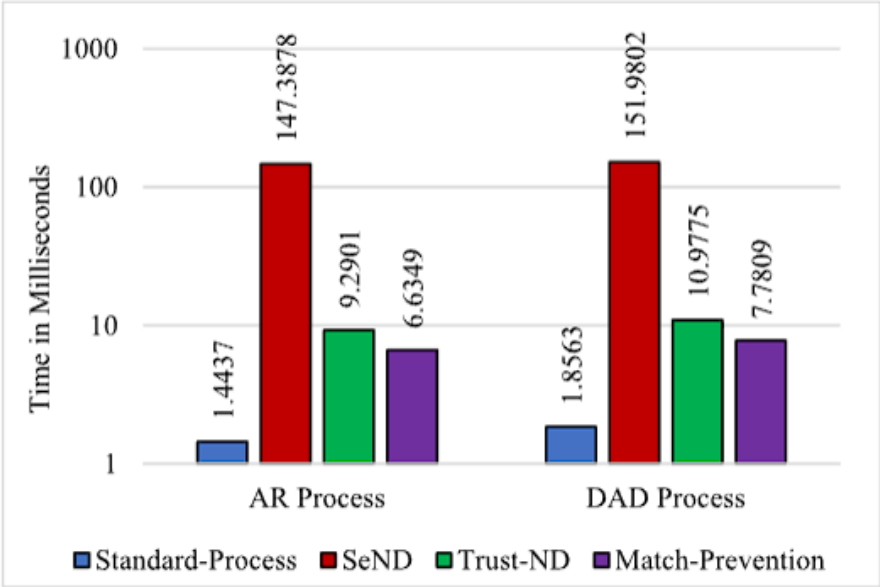


FIGURE 10. Total processing time of AR and DAD.

Bandwidth Consumption (Table 9)
The size of NDP messages affects bandwidth usage:

SeND: Largest message size (454 bytes) and highest bandwidth consumption (427% more than the baseline).

Trust-ND: Moderate size (118 bytes) with 37% more bandwidth usage than the baseline.

Match-Prevention: Smallest secure message size (102 bytes), consuming only 18% more bandwidth than the baseline.

SeND and Match-Prevention were a success preventing attacks while Trust-ND and Standard-Process failed to do so.

TABLE 9. Bandwidth consumption.

Technique Name	NDP Message Type (Size)		Bandwidth Consumption	
	NS	NA	NS	NA
Standard-Process (AR/DAD)	86	86	1.8989	1.1582
SeND	454	454	10.0247	6.1145
Trust-ND	118	118	2.6055	1.5892
Match-Prevention	102	102	2.2522	1.373

DoS on DAD Attack After DoS on DAD it attacks as DoS on DAD(Duplicate Address Detection) DoS on DAD a new entry for an IPv6 link-local network host is attacked in such a way that Duplicate address detection cannot be performed for the host. An attacker can send the new host replayed NS messages, and issue DoS-on-DAD attacks such that completeness of the DAD process, which makes sure that a tentative IP address is unique, is impossible. Table 10 shows the DoS-on-DAD attack experiment results.

TABLE 10. DoS-on-DAD attack experiment.

Technique Name	Number of Experimental Runs	Success DAD	Failure DAD	ProcessSR
Standard-DAD	20	0	20	0
SeND	20	20	0	1
Trust-ND	20	0	20	0
Match-Prevention	20	20	0	1

An example of the attack is a DoS-on-AR attack: this will be targeting a host that is using AR to obtain the MAC address of another host it plans on communicating with in the future. It aims to prevent the network by sending a different MAC address to the hosts to perform a Man-in-the-Middle attack. The DoS-on-AR attack can be carried out by replying to every NS message sent by the host from the attacker in completing the AR. The attack was implemented using THC tool and Scapy on Standard-AR, proposed Match-Prevention method, and existing SeND and Trust-ND protocols. The results of the DoS-on-AR attack are presented in Table 11.

TABLE 11. DoS-on-AR attack experiment.

Technique Name	Number of Experimental Runs	Success DAD	Failure DAD	ProcessSR
Standard-AR	20	0	20	0
SeND	20	20	0	1
Trust-ND	20	0	20	0
Match-Prevention	20	20	0	1

Experimental results confirm that fake NA messages may be verified by both SeND and Match-Prevention and prevent DoS-on-AR attacks and DoS-on-DAD attacks respectively in an IPv6 link-local network during AR and DAD processes. On the contrary however, Trust-ND and Standard-Process (Standard-AR and Standard-DAD) does not secure AR and DAD. In addition, although SeND can protect against DoS-on-AR and DoS-on-DAD attacks, this approach is vulnerable to flooding attacks, which is described in the next section.

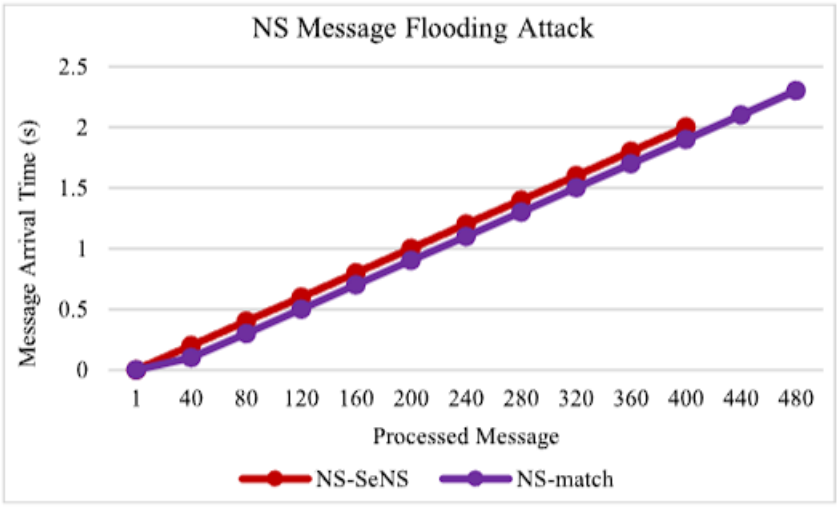


FIGURE 12. NS message flooding attack against the receiver host.

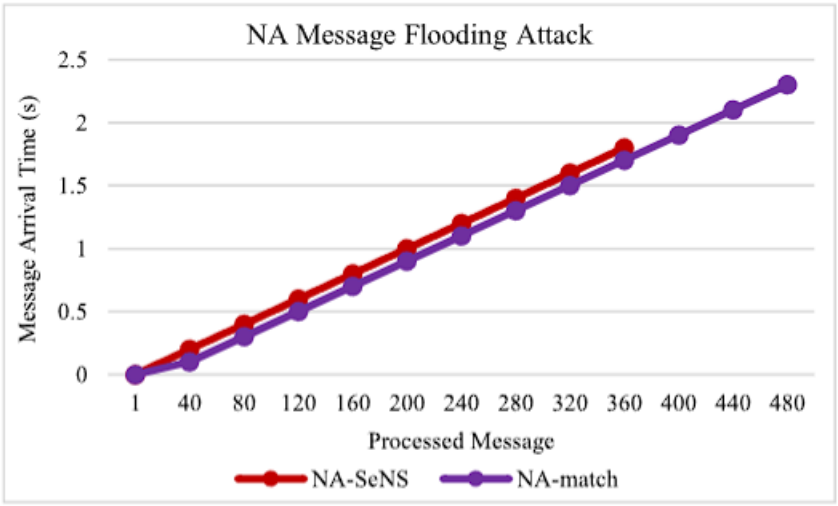


FIGURE 13. NA message flooding attack against the receiver host.

1. **SeND** machines crashed under heavy flooding
2. **Match-Prevention** managed to handle all flooding

The Match-Prevention technique outperforms SeND and Trust-ND by reducing processing times, conserving bandwidth, and providing robust security against DoS and flooding attacks. It offers an efficient and secure solution for IPv6 link-local networks.

Advantages:

Improved Efficiency (Reduced Processing Time and Bandwidth Consumption)-The Match-Prevention approach speeds up Address Resolution (AR) and Duplicate Address Detection (DAD) operations compared to other secure techniques like SeND and Trust-ND.

Robust Security Against DoS and Flooding Attacks- Match-Prevention avoids DoS attacks during AR and DAD operations by authenticating NS and NA messages. Unlike SeND, which is susceptible to flooding assaults and crashes in high traffic, Match-Prevention displays significant resilience, handling massive amounts of attack traffic without failure.

Conclusion:

In IPv6 link-local networks, AR (Address Resolution) and DAD (Duplicate Address Detection) leverage NS (Neighbor Solicitation) and NA (Neighbor Advertisement) message types, but these message types are unperturbed, unencrypted information rendering them vulnerable to Denial-of-Service attacks as the receiver has no opportunity to verify their content. This is resolved in the Match-Prevention manner by ensuring that the target IP address is locked with SHA-3 hashing algorithm. Match-Prevention from both Send and Trust-ND, preventing collision and brute-force attacks and avoiding DoS in AR and DAD, used less processing time and the bandwidth.

2.5 Collaborative Intrusion Detection System with Snort Machine Learning Plugin

The study provides evidence that the plugin detects Denial of Service (DoS) attacks and probing activities without producing false positives and false negatives making it a cost effective approach for real-time scenarios.

Most of the existing IDS used either Host-Based IDS (HIDS) or Network-Based IDS (NIDS) separately. This dependence meant that the two systems could not leverage the advantages of both types for detection, limiting it to half coverage at best.

Problems

Traditional IDS solutions like Snort were somewhat signature-based, meaning they were unable to adapt to new and emerging attack vectors unless defined specifically in a rule set. But this rigidity allowed new attacks that were never seen before to bypass these systems really easily. Making predictions and visualizations of the network traffic data in real time was another challenge which complicated the detection process. Traditional systems could not deliver the level of sophistication required for real-time data streams monitoring and analysis.

Objective

The authors aimed to create a CIDS through the integration of both HIDS and NIDS elements to deliver an improved and more precise form of intrusion detection and attack prevention.

The researchers aimed to include ML techniques, specifically through Snort plugins, for detecting attacks more accurately and adapting easily to newer attack types. The goal was to tackle difficult cyber threats that often remain unnoticed by conventional signature-based systems. IDLESS, which compares machine learning and IDS.

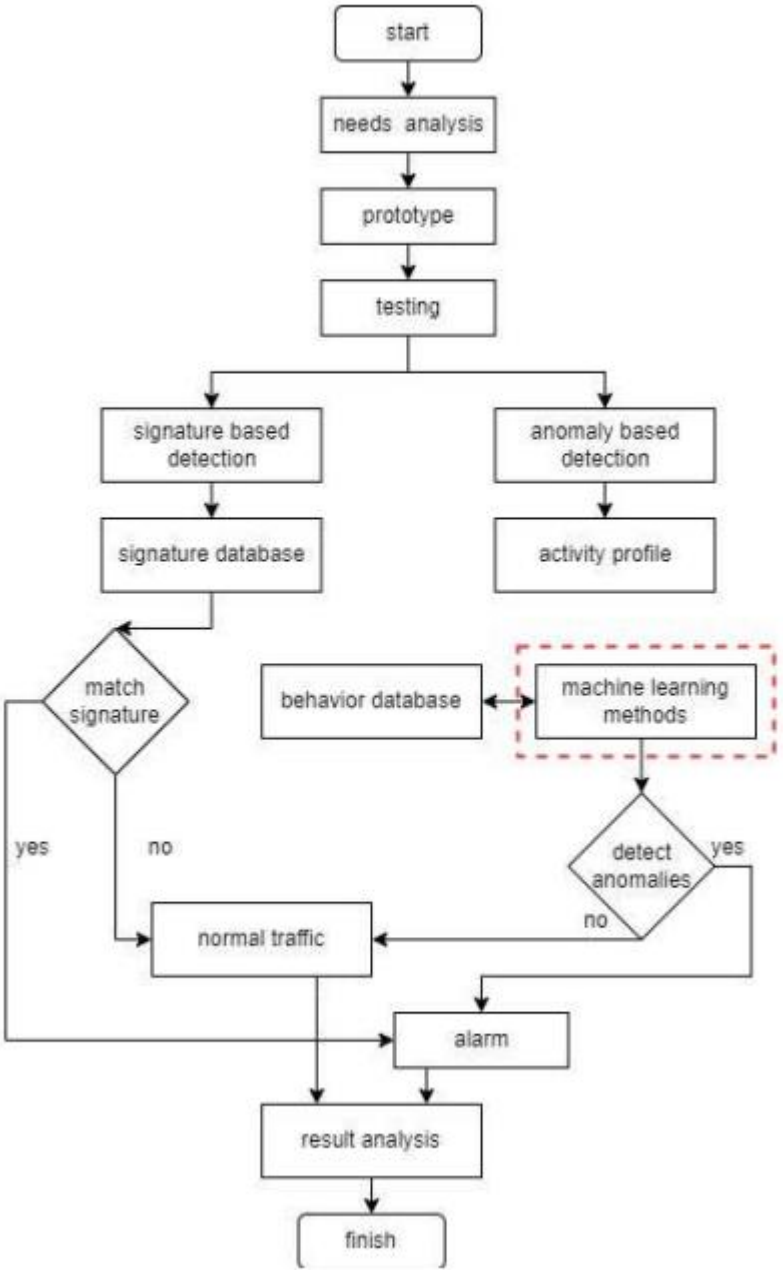


Figure 1: Flowchart of experiment

Design

In achieving this, researchers described the Collaborative Intrusion Detection System process setup along with its integrated components and configuration.

System was deployed on a virtual machine where they used a main host for multi-IDS setup and attacker host for mimicking attacks to create logs for the intrusion detection system. Ids function is implemented on ubuntu and windows 10. Main host used ubuntu and it was used for ids function, and on host we used windows 10 as the wazuh agent. They installed Wazuh, Snort, elasticsearch(log storage), kibana(data visualisation) and filebeat(manage logs) Components: They went on to set up Wazuh server to manage the agents that monitor all the hosts. This configuration allows to perform log analysis, file integrity checks, intrusion detection, and real-time alerting. As such, Elasticsearch was implemented also as the database management system to store and handle the alerts data generated by Wazuh. Configuration required detailing network host addresses, node names, and cluster settings for effective communication between Wazuh and Elasticsearch. We used Logstash to normalize and process the incoming logs so they would be in the correct format for visualization. Filebeat is installed to send the logs from Wazuh to Elasticsearch. We used a specific ruleset on Snort to identify different kinds of attacks as NIDS. Nevertheless, a separate machine learning Snort plugin was created to supplement the classical ruleset-based detection model with ML-based models. This plugin allowed Snort to employ machine learning techniques in order to grow key accuracy. On the visualization side, Kibana was integrated.

Results

REAL-TIME DOS ATTACK PREDICTION

Packet Flow	Machine Learning Plugin	IDS Log Detection
Incoming Packets	11.065	11.065
Total Packets	11.065	11.065

System identified a total of 12997 attacks including 11.065 DOS attack. In total, Wazuh generated 563 events.

The tasks with Sailor were implemented Support Vector Machine (SVM) algorithm and it obtained a high accuracy of attack detection. From a statistical performance perspective: The accuracy of the SVM-model was 97.7% on training data and 97.5% on testing data. We found that the detection rates given on the results for DoS attacks are equal to (99.3% accuracy), and Probe attacks (98.4% accuracy).

Real-time Prediction mechanisms of the system was able to match the recorded logs with both machine learning logs and IDS logs giving an equal number of packets being detected during the attack.

Using cross-validation metrics, the high performance of various attack types such as the detection accuracy, precision, recall, and F-Measure of DoS attacks achieved a value of 99%.

Advantage

The traffic pattern can be analysed by SVM, increasing the accuracy of attack detection. Because it relies on predefined rules (not human-based detection), SNORT is nothing more than a signature method of classification. In contrast, SVM can use the patterns learned from its training data just as easily to recognize new attacks.

Table of Comparison

Title	Methodology	Limitations	Advantages
GR-AD-KNN Algorithm for DoS Detection	Optimized KNN algorithm with double dimensionality reduction and weighted feature influence using information gain rate.	Relies heavily on effective selection of the k value for optimal accuracy and struggles with small group classification issues, leading to misjudgements.	Efficient, adaptable, improved accuracy, and reduced sensitivity to parameter selection.
Enhanced Windows Fuzzy Firewall for DOS Attack Prevention	Uses fuzzy logic for nuanced traffic analysis and integrates advanced security measures into the Windows Firewall.	High computational demands, potential misclassification of benign traffic, and binary traffic blocking policies that disrupt legitimate operations.	Robust against unpredictable attack patterns, supports multiple protocols, and user-friendly.
Match Prevention Technique Against DOS Attack on Address Resolution and DAD in IPV6	Utilizes the Match Prevention technique to secure AR and DAD processes using a cryptographic mechanism and lightweight design.	High processing complexity, scalability challenges in larger networks, and dependency on SHA-3, increasing resource consumption.	Reduced processing time and bandwidth consumption, strong resilience against high traffic, and effective DoS prevention.
IcmpV6-Based DOS and DDOS Attacks Detection Using Machine Learning Techniques, Open Challenges, and Blockchain Applicability	Uses machine language. Algorithms like Support Vector Machine (SVM), Artificial Neural Networks (ANN), Decision trees and hybrid models.	Lack effective feature selection schemes.	Reduced Processing Time and Bandwidth Consumption.

Collaborative Intrusion Detection System with Snort Machine Learning	SVM machine learning, Snort and Wazuh.	SVM with snort has involves additional complexity in terms of configurations.	SVM can analyse traffic patterns and improve the accuracy of attack detection.

Chapter 3: The Theoretical Framework

The following section gives an overview, theoretical background to the project represented by this document, concentrating detecting DOS attacks in IPv6 networks. The chapter first delves into the basics of DoS attacks and why they are difficult to detect, especially over IPv6. The proposed research first creates an introduction to the inherent terms then focuses on Snort as an Intrusion Detection System (IDS) for the monitoring of traffic then elucidates Support Vector Machine (SVM) as a machine learning significant technique to proactively administer such attacks. It is important to highlight that the theoretical framework also describes Jupyter Lab as the development environment for data analysis and implementation of the model. The chapter, then, describes the core research problem and objectives, which must transition towards more versatile detection means to overcome the current methods restrictions to secure better IPv6 networks in light of dynamic attack vectors.

3.1 Snort is a signature based Intrusion Detection Systems (IDS) used to identify network based intrusions, by performing packet logging and real-time traffic analysis of packet data. It has been effective at detecting many network attacks including DoS attacks. Snort functions as a packet-sniffing tool, monitoring network traffic in real time and checking it against signatures (patterns) for known types of malicious activity. This allows Snort to recognize well-known and already known attacks.

As part of this project, Snort will be evaluated as an IPv6 DoS detection mechanism. The tool will be needed to capture and analyze network traffic in real-time and maintain logs that can be analyzed for evidence of DoS attempts. This comparison will use Snort's signature-based detection of known attacks as the baseline against which the machine learning-based approach (SVM) aims to perform better. Snort will assist in evaluating whether SVM is capable of improving detection rates in excess of the constraints of classic signature-based methods.

3.2 Support Vector Machine(SVM) is one of the supervised machine learning algorithms, primarily used for classification and regression tasks. Support Vector

Machine (SVM) is a supervised machine learning algorithm that classifies all classes in the feature space by finding the hyperplane that best separates the classes. Here cpu and memory usage will be used as features to classify the network traffic with support vector machine as normal flow or DoS attack in this study. In this project, we focus on extracting features from Snort logs to use in training the model.

In this project, SVM is used to improve detection of DoS attacks. The SVM model will then learn these attack patterns by training on features extracted from Snort logs, allowing it to detect attacks that traditional IDS tools such as Snort might miss. The machine learning based technique will complement Snort's signature-based detection to help improve its overall accuracy for identifying both old and new, sophisticated DoS attacks. Moreover, SVM eliminates the false-positive and false-negative methods; therefore, SVM provides an efficient detection/ classification mechanism.

3.3 Artificial Neural Network(ANN)

Artificial Neural Network (ANN) is a supervised learning algorithm with inspiration from the human brain's structure and working. It is applied widely for complex classification and pattern recognition tasks. ANN is implemented in this project to classify network traffic as normal or Denial of Service (DoS) attack according to CPU and memory consumption. These features are extracted from the Snort logs and utilized in training the ANN model in a manner that the model can recognize traffic behavior indicating potential attacks.

The primary use of using ANN in this project is to improve the intrusion detection capability of traditional intrusion detection systems like Snort. Though Snort relies on signature-based methods, which tend to overlook new or modified attack patterns, ANN is capable of learning from past experiences and detecting known and unknown DoS attacks. Through it all, the ANN-based approach enhances detection rates while reducing false positives as well as false negatives. This machine learning technique is a smart layer that strengthens Snort and makes it more robust against adaptive attacks in IPv6 network infrastructures.

3.4 Jupyter Lab is an IDE that is great for data science, machine learning and research applications. It serves as a very interactive and user-friendly environment for coding, data analysis, and data visualization. There are many programming languages

available under Jupyter Lab, but it is mostly used in python, and integrates very fondly with the likes of scikit-learn, pandas, and matplotlib. For this project, we will use Jupyter Lab for coding, running ML models, process Snort log data, and visualize the result.

In implementation and analyzing the SVM model's performance, Jupyter Lab will be one of the most important tools. Therefore, model building becomes an iterative process in which feedback can be given by the Jupyter interactive interface and corrections can be made quickly to keep fine-tuning our model. Moreover, the use of visualization libraries such as matplotlib in Jupyter will help visualize the model accuracy, confusion matrices, and other performance metrics.

3.5 Data preprocessing is a fundamental process of getting raw data into a format that can be used effectively for machine learning models. For this project (detecting port scans), we will be using Snort logs as the data source for our SVM machine learning model; however, data needs to be cleaned, transformed, and structured in such a way to be applicable for training the SVM model. For this, you will have a lot of data preprocessing to do — to get rid of missing values, feature extraction, normalization, etc. The importance of this step is that the quality and data used to train the machine learning model impact its prediction accuracy heavily.

because snort logs are raw so need preprocessing before using it into SVM model. True, the logs will include not only packet sizes, protocols, timestamps and other information relating to inbound and outbound traffic. This will be analytic using pandas and scikit-learn, extracting the relevant feature from the raw logs, normalizing the data so that the SVM model can learn the data as expected. With proper feature engineering and preprocessing, SVM will classify DoS attack with a better accuracy.

Chapter 4 Methodology

This chapter will discuss the steps that will be implemented in this project. The objective is to capture DOS attack in Intrusion detection system which is Snort in Ipv6 networks and use machine learning like SVM (Support Vector Machine) and Artificial Neural Network (ANN) which will deploy snort logs.

4.1 Data Collection

Firstly, using custom Snort rule, it detects ICMP Flood. The snort logs will serve as the basis for training and testing the SVM and ANN model. The dataset will split into two, one for training model and the other for evaluating its performance which is testing. Will be using 80% on training model and 20% on testing the performance.

4.2 Data Preprocessing

When data is collected, the next step is preprocessing. This involves Snort logs by removing irrelevant information and will structure the data into a csv file(dataset) for machine learning. The logs will consist of Timestamp, Source_IPv6, Destination_IPv6 Protocol, Length, Info and Attack. This is to ensure the data is in a suitable for training and testing in machine learning.

4.3 Feature extraction and selection

This step will identify the most relevant characteristics of the traffic data which can indicate DOS attack, such as number of packets and number of attacks. Feature Importance will be used for SVM and ANN model. This will improve model efficiency and ensure model will focus on most impactful features.

4.4 Model Development

This is where, Support Vector Machine (SVM) and Artificial Neural Network (ANN) will be used. The model will be trained on the labelled dataset which will split into training and test sets. Training dataset will use to teach the model to distinguish between normal traffic and DOS attack traffic.

4.5 Model Evaluation

Once model is trained, its kernel will be scored based on the test dataset. We then evaluate the performance of the SVM and ANN model in terms of accuracy, precision, recall and F1-score. This model will be evaluated on the ability to reduce false positive and false negative.

4.6 Analysis

When the model is full trained, tested and optimized, the results will be analysed. A simple graphical user interface, where user can upload a dataset which can detect packets and attacks probability. The final model performance will be compared to Snort.

Chapter 5: Implementation

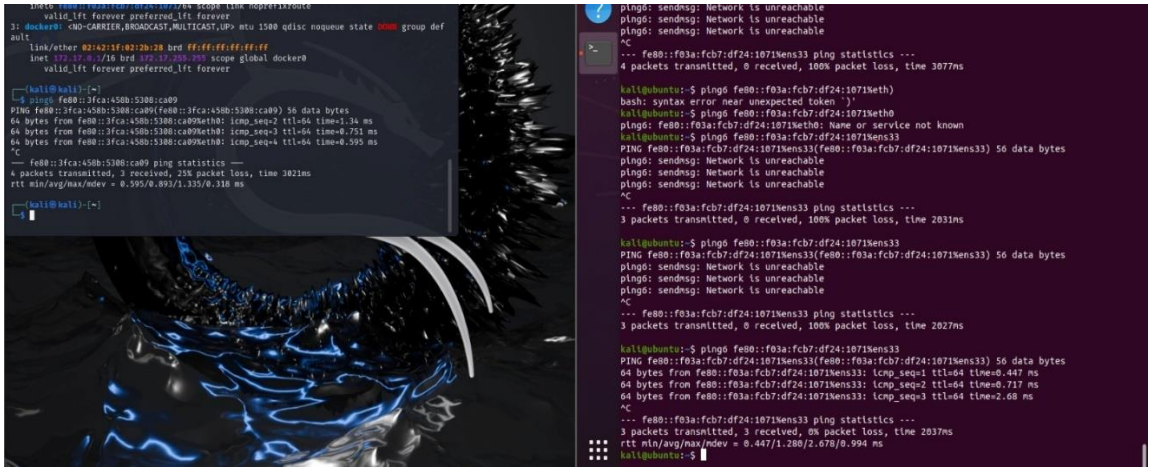
This chapter is about the procedure to carry out the results. The project will be implemented in discrete stages over the course of this semester.

5.1 Development Strategy

Week 1

Configuring the project's basic technologies, such as Snort, Jupyter Lab, and the essential Python libraries. It also entails establishing the IPv6 traffic gathering environment and setting up a project repository on GitHub for version control. Research materials and references on DoS attacks and Snort settings will be collected.

a. There will be two virtual machines, Ubuntu(victim) and Kali-Linux(attacker). Snort will be installed in Ubuntu which will capture packets and attacks from ipv6 networks. Firstly, I need to ensure both virtual machines can ping each other. Based on the title, which is to detect dos attack from ipv6 network, we need to use ipv6 address. Below shows the ping from kali-linux and ubuntu works.



b. In snort's configuration file, I need to add a line which will allow snort to store packets and attacks detected in a specific file in the ubuntu which is var/log/snort/alert.

```
# unified2
# Recommended for most installs
# output unified2: filename merged.log, limit 128, nostamp, mpls_event_types, vlan_event_types
output unified2: filename snort.log, limit 128, nostamp, mpls_event_types, vlan_event_types
output alert_full: /var/log/snort/alert
# Additional configuration for specific types of installs
# output alert_unified2: filename snort.alert, limit 128, nostamp
# output log_unified2: filename snort.log, limit 128, nostamp
```

c. Before I enter my custom snort rule, I need to ensure that snort can detect dos rules and icmp rules. This will allow snort to run and capture icmp packets.

```
include $RULE_PATH/ddos.rules
include $RULE_PATH/dns.rules
include $RULE_PATH/dos.rules
include $RULE_PATH/experimental.rules
#include $RULE_PATH/exploit-kit.rules
include $RULE_PATH/exploit.rules
#include $RULE_PATH/file-executable.rules
#include $RULE_PATH/file-flash.rules
#include $RULE_PATH/file-identify.rules
#include $RULE_PATH/file-image.rules
#include $RULE_PATH/file-multimedia.rules
#include $RULE_PATH/file-office.rules
#include $RULE_PATH/file-other.rules
#include $RULE_PATH/file-pdf.rules
include $RULE_PATH/finger.rules
include $RULE_PATH/ftp.rules
include $RULE_PATH/icmp-info.rules
include $RULE_PATH/icmp.rules
include $RULE_PATH/imap.rules
include $RULE_PATH/udp6.rules
```

Week 2

Configure Snort to detect IPv6 packets (Normal and Attack). Specific Snort rules for the detection of ICMP floods will be developed. The data will be collected and

preprocessed which involves labelling each record as either normal traffic or as an attack traffic. This information will help to decide what the next training and testing will consist of.

a. Once snort`s configuration is done, I can enter my custom snort rule in local rules file. Below is the rule and it will alert every single icmpv6 packet going to ubuntu`s HOME_NET. I have set the protocol which targets icmp traffic. It will accept from any port which is why I need to make sure ubuntu can ping kali-linux's ipv6 address. So whenever the message appears "ICMPv6 Flood Detected", it will be stores in snort's logs. The threshold:track by_src means it will monitor based on source IP address.

```
#alert icmp any any -> $HOME_NET any (msg:"ICMPv6 Packet Detected"; sid:1000001; rev:1;)
alert icmp any any -> $HOME_NET any (msg:"ICMPv6 Flood Detected";  threshold:type both, track by_src, count 10, seconds 1; sid:1000004; rev:1;)
```

alert icmp any any -> $HOME_NET any (msg:"ICMPv6 Flood Detected"; threshold:type both, track by_src, count 10, seconds 1; sid:1000004; rev:1;)

b. Next step is to start snort

```
kali@ubuntu:~$ sudo snort -i ens33 -c /etc/snort/snort.conf -l /var/log/snort/
```

c. Then start to ping from kali-linux. This command means it will send ICMPv6 packets to Ubuntu's ipv6 address. For this project, I plan to have more than 10,000 packets which contain icmp packets and icmp flood attacks. To achieve this, I need to let snort run for an hour.

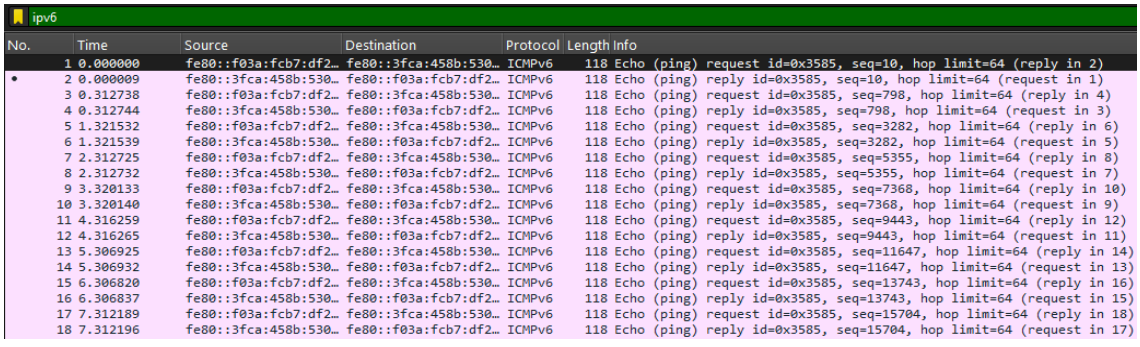
```
┌──(kali㉿kali)-[~]
└─$ sudo ping6 -f fe80::3fca:458b:5308:ca09%eth0
[sudo] password for kali:
PING fe80::3fca:458b:5308:ca09%eth0(fe80::3fca:458b:5308:ca09%eth0) 56 data bytes
.
```

d. After an hour, logs will be stored. Two files will appear which are alert and snort log. Below shows an example of alerts and packets captured. Snort can be viewed in text file while icmp packets can be viewed in wireshark.

i) Alerts

```
1416 [**] [1:1000004:1] ICMPv6 Flood Detected [**]
1417 [Priority: 0]
1418 05/13-07:11:35.018728 fe80::3fca:458b:5308:ca09 -> fe80::f03a:fcb7:df24:1071
1419 IPV6-ICMP TTL:64 TOS:0x0 ID:0 IpLen:40 DgmLen:104
1420
1421 [**] [1:1000004:1] ICMPv6 Flood Detected [**]
1422 [Priority: 0]
1423 05/13-07:11:36.031913 fe80::f03a:fcb7:df24:1071 -> fe80::3fca:458b:5308:ca09
1424 IPV6-ICMP TTL:64 TOS:0x0 ID:0 IpLen:40 DgmLen:104
1425
1426 [**] [1:1000004:1] ICMPv6 Flood Detected [**]
1427 [Priority: 0]
1428 05/13-07:11:36.031932 fe80::3fca:458b:5308:ca09 -> fe80::f03a:fcb7:df24:1071
1429 IPV6-ICMP TTL:64 TOS:0x0 ID:0 IpLen:40 DgmLen:104
1430
1431 [**] [1:1000004:1] ICMPv6 Flood Detected [**]
1432 [Priority: 0]
1433 05/13-07:11:37.003845 fe80::f03a:fcb7:df24:1071 -> fe80::3fca:458b:5308:ca09
1434 IPV6-ICMP TTL:64 TOS:0x0 ID:0 IpLen:40 DgmLen:104
1435
1436 [**] [1:1000004:1] ICMPv6 Flood Detected [**]
1437 [Priority: 0]
1438 05/13-07:11:37.003848 fe80::3fca:458b:5308:ca09 -> fe80::f03a:fcb7:df24:1071
1439 IPV6-ICMP TTL:64 TOS:0x0 ID:0 IpLen:40 DgmLen:104
1440
1441 [**] [1:1000004:1] ICMPv6 Flood Detected [**]
1442 [Priority: 0]
```

ii) Icmp Packets



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	fe80::f03a:fcb7:df2…	fe80::3fca:458b:530…	ICMPv6	118	Echo (ping) request id=0x3585, seq=10, hop limit=64 (reply in 2)
2	0.000009	fe80::3fca:458b:530…	fe80::f03a:fcb7:df2…	ICMPv6	118	Echo (ping) reply id=0x3585, seq=10, hop limit=64 (request in 1)
3	0.312738	fe80::f03a:fcb7:df2…	fe80::3fca:458b:530…	ICMPv6	118	Echo (ping) request id=0x3585, seq=798, hop limit=64 (reply in 4)
4	0.312744	fe80::3fca:458b:530…	fe80::f03a:fcb7:df2…	ICMPv6	118	Echo (ping) reply id=0x3585, seq=798, hop limit=64 (request in 3)
5	1.321532	fe80::f03a:fcb7:df2…	fe80::3fca:458b:530…	ICMPv6	118	Echo (ping) request id=0x3585, seq=3282, hop limit=64 (reply in 6)
6	1.321539	fe80::3fca:458b:530…	fe80::f03a:fcb7:df2…	ICMPv6	118	Echo (ping) reply id=0x3585, seq=3282, hop limit=64 (request in 5)
7	2.312725	fe80::f03a:fcb7:df2…	fe80::3fca:458b:530…	ICMPv6	118	Echo (ping) request id=0x3585, seq=5355, hop limit=64 (reply in 8)
8	2.312732	fe80::3fca:458b:530…	fe80::f03a:fcb7:df2…	ICMPv6	118	Echo (ping) reply id=0x3585, seq=5355, hop limit=64 (request in 7)
9	3.320133	fe80::f03a:fcb7:df2…	fe80::3fca:458b:530…	ICMPv6	118	Echo (ping) request id=0x3585, seq=7368, hop limit=64 (reply in 10)
10	3.320140	fe80::3fca:458b:530…	fe80::f03a:fcb7:df2…	ICMPv6	118	Echo (ping) reply id=0x3585, seq=7368, hop limit=64 (request in 9)
11	4.316259	fe80::f03a:fcb7:df2…	fe80::3fca:458b:530…	ICMPv6	118	Echo (ping) request id=0x3585, seq=9443, hop limit=64 (reply in 12)
12	4.316265	fe80::3fca:458b:530…	fe80::f03a:fcb7:df2…	ICMPv6	118	Echo (ping) reply id=0x3585, seq=9443, hop limit=64 (request in 11)
13	5.306925	fe80::f03a:fcb7:df2…	fe80::3fca:458b:530…	ICMPv6	118	Echo (ping) request id=0x3585, seq=11647, hop limit=64 (reply in 14)
14	5.306932	fe80::3fca:458b:530…	fe80::f03a:fcb7:df2…	ICMPv6	118	Echo (ping) reply id=0x3585, seq=11647, hop limit=64 (request in 13)
15	6.306820	fe80::f03a:fcb7:df2…	fe80::3fca:458b:530…	ICMPv6	118	Echo (ping) request id=0x3585, seq=13743, hop limit=64 (reply in 16)
16	6.306837	fe80::3fca:458b:530…	fe80::f03a:fcb7:df2…	ICMPv6	118	Echo (ping) reply id=0x3585, seq=13743, hop limit=64 (request in 15)
17	7.312189	fe80::f03a:fcb7:df2…	fe80::3fca:458b:530…	ICMPv6	118	Echo (ping) request id=0x3585, seq=15704, hop limit=64 (reply in 18)
18	7.312196	fe80::3fca:458b:530…	fe80::f03a:fcb7:df2…	ICMPv6	118	Echo (ping) reply id=0x3585, seq=15704, hop limit=64 (request in 17)

Week 3

The logs for Snort are collected and preprocessed that eliminates the unwanted information and fill the missing ones by cleansing the data. Features, such as packet size, event traffic rates are to be extracted, normalized and then partitioned as the

training and testing datasets. The above steps are carried out so that the data becomes in a suitable format for SVM model.

a. Snort logs(packets and alerts) are collected and are stored in csv file. The dataset consists of timestamp, source_ip, destination_ipv6, protocol, length, info and attack. For normal packets, it is labelled as 0 while alerts are labelled as 1 in Attack column.

	Timestam	Source_IPv6	Destination_IPv6	Protocol	Length	Info	Attack
2	0	fe80::f03a:fcb7:df24:1071	fe80::3fca:458b:5308:ca09	ICMPv6	118	Echo (ping) request id=0xa079, seq=10, hop limit=64 (reply in 2)	0
3	6.00E-06	fe80::3fca:458b:5308:ca09	fe80::f03a:fcb7:df24:1071	ICMPv6	118	Echo (ping) reply id=0xa079, seq=10, hop limit=64 (request in 1)	0

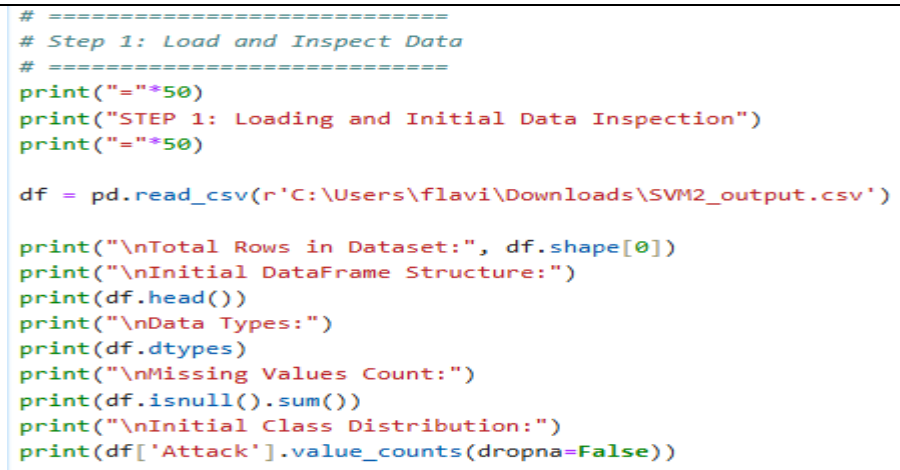
5922	2959.879	fe80::f03a:fcb7:df24:1071	fe80::3fca:458b:5308:ca09	ICMPv6	118	Echo (ping) request id=0xa079, seq=24820, hop limit=64 (reply in 5922)	0
5923	2959.879	fe80::3fca:458b:5308:ca09	fe80::f03a:fcb7:df24:1071	ICMPv6	118	Echo (ping) reply id=0xa079, seq=24820, hop limit=64 (request in 5921)	0
5924	2960.88	fe80::f03a:fcb7:df24:1071	fe80::3fca:458b:5308:ca09	ICMPv6	118	Echo (ping) request id=0xa079, seq=26429, hop limit=64 (reply in 5924)	0
5925	2960.88	fe80::3fca:458b:5308:ca09	fe80::f03a:fcb7:df24:1071	ICMPv6	118	Echo (ping) reply id=0xa079, seq=26429, hop limit=64 (request in 5923)	0
5926	2961.873	fe80::f03a:fcb7:df24:1071	fe80::3fca:458b:5308:ca09	ICMPv6	118	Echo (ping) reply id=0xa079, seq=26429, hop limit=64 (request in 5923)	0
5927	05/16-01:0	fe80::f03a:fcb7:df24:1071	fe80::3fca:458b:5308:ca09	IPV6-ICMP	0	ICMPv6 Flood Detected	1
5928	05/16-01:0	fe80::3fca:458b:5308:ca09	fe80::f03a:fcb7:df24:1071	IPV6-ICMP	0	ICMPv6 Flood Detected	1
5929	05/16-01:0	fe80::f03a:fcb7:df24:1071	fe80::3fca:458b:5308:ca09	IPV6-ICMP	0	ICMPv6 Flood Detected	1
5930	05/16-01:0	fe80::3fca:458b:5308:ca09	fe80::f03a:fcb7:df24:1071	IPV6-ICMP	0	ICMPv6 Flood Detected	1

b. Once labelled, I can perform Support Vector Machine(SVM) and Artificial Neural Network(ANN) to train and test the dataset.

Week 4

In this practical implementation of SVM and ANN will be implemented using python Scikit-learn library. Train the model on the training set the processed data and fine-tune the model with cross validation. I will then test the model and assess the results with performance measurements like accuracy, precision, and recall.

i) SVM

	Code
1	 Dataset is loaded from CSV file, structure data types missing values and class distribution are checked. This helps to understand the initial condition of the data before processing begins.
2	 Timestamp values are converted into numeric form to process. Missing values or invalid values are filled in and zero values in Length column are

```
# ==============================
# Step 1: Load and Inspect Data
# ==============================
print("="*50)
print("STEP 1: Loading and Initial Data Inspection")
print("="*50)

df = pd.read_csv(r'C:\Users\flavi\Downloads\SVM2_output.csv')

print("\nTotal Rows in Dataset:", df.shape[0])
print("\nInitial DataFrame Structure:")
print(df.head())
print("\nData Types:")
print(df.dtypes)
print("\nMissing Values Count:")
print(df.isnull().sum())
print("\nInitial Class Distribution:")
print(df['Attack'].value_counts(dropna=False))
```

```
# ======================
# Step 2: Data Cleaning
# ======================
print("\n" + "="*50)
print("STEP 2: Data Cleaning")
print("="*50)

def parse_timestamp(ts):
    try:
        if isinstance(ts, (int, float)) or (isinstance(ts, str) and ts.replace('.', '', 1).isdigit()):
            return float(ts)
        parsed = pd.to_datetime(ts, format='%m/%d-%H:%M:%S.%f', errors='coerce')
        return parsed.timestamp() if pd.notnull(parsed) else np.nan
    except:
        return np.nan

df['Timestamp'] = df['Timestamp'].apply(parse_timestamp)
df['Timestamp'] = df['Timestamp'] - df['Timestamp'].min()
df['Timestamp'] = df['Timestamp'].fillna(df['Timestamp'].mean())
df['Length'] = df['Length'].replace(0, df['Length'].median())
duplicates = df.duplicated().sum()
if duplicates > 0:
    df = df.drop_duplicates()
print("\nNumber of Duplicate Rows Removed:", duplicates)
df['Attack'] = df['Attack'].astype(int)
print("\nClass Distribution After Cleaning:")
print(df['Attack'].value_counts(dropna=False))
```

	replaced with median to avoid errors. Duplicated rows are also removed to ensure data quality and Attack column is converted to integers.
3	``` # ===================== # Step 3: Feature Engineering # ===================== print("\n" + "="*50) print("STEP 3: Feature Selection and Engineering") print("="*50) def ipv6_to_int(ip): try: return int(ipaddress.IPv6Address(ip)) except: return 0 df['Source_IPv6_int'] = df['Source_IPv6'].apply(ipv6_to_int) df['Destination_IPv6_int'] = df['Destination_IPv6'].apply(ipv6_to_int) df['is_ICMPv6_flood'] = df['Info'].str.contains('ICMPv6 Flood Detected', case=False, na=False).astype(int) df = pd.get_dummies(df, columns=['Protocol'], prefix='Protocol', dummy_na=False) feature_cols = ['Timestamp', 'Length', 'is_ICMPv6_flood', 'Source_IPv6_int', 'Destination_IPv6_int'] \ + [col for col in df.columns if col.startswith('Protocol_')] features = df[feature_cols] target = df['Attack'] print("\nFeature Statistics:") print(features.describe()) print("\nClass Distribution:") print(target.value_counts()) # == # Step 3.5: Filter Only ICMPv6 Flood and Normal Traffic # == print("\nFiltering dataset to include only ICMPv6 Flood Detected and Normal traffic...") df_filtered = df[(df['is_ICMPv6_flood'] == 1)	(df['Attack'] == 0)] features = df_filtered[feature_cols] target = df_filtered['Attack'] print("Filtered Class Distribution:") print(target.value_counts()) ``` Step 3 - Ipv6 addresses are convented into numerical values so the model can use it. A new column is added to identify whether a packet is part of ICMPv6 flood attack. The Protocol column is transformed into separate binary columns using one-hot encoding so that each protocol can be treated as an individual feature. Step 3.5 – To focus on detecting ICMPv6 flood attacks, the dataset is filtered to include only two types of traffic which are normal traffic and ICMPv6 flood traffic. This will simplify the classification task and improve model accuracy.

4

```
# ============================
# Step 4: Data Preprocessing
# ============================
print("\n" + "="*50)
print("STEP 4: Data Preprocessing")
print("="*50)

X_train, X_test, y_train, y_test = train_test_split(
    features, target,
    test_size=0.3,
    random_state=42,
    stratify=target
)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

print("\nTraining set shape:", X_train_scaled.shape)
print("Test set shape:", X_test_scaled.shape)
print("\nTraining Class Distribution:")
print(y_train.value_counts())
print("Test Class Distribution:")
print(y_test.value_counts())
```

Data is split into training and testing sets. Then feature values are scaled using standardization to bring them into a similar range. This is important because, SVM is sensitive to the scale of input features.

5

```
# ============================
# Step 5: SVM Model Training
# ============================
print("\n" + "="*50)
print("STEP 5: SVM Model Training")
print("="*50)

train_start_time = time.time()

svm_model = SVC(
    kernel='linear',
    C=1.0,
    random_state=42,
    class_weight='balanced',
    probability=True
)
svm_model.fit(X_train_scaled, y_train)

train_end_time = time.time()
train_duration = train_end_time - train_start_time

print("\nModel training completed.")
print(f"Training Time: {train_duration:.2f} seconds")
print(f"Number of support vectors per class: {svm_model.n_support_}")

pipeline = make_pipeline(StandardScaler(), SVC(kernel='linear', C=1.0, random_state=42, class_weight='balanced'))
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = cross_val_score(pipeline, features, target, cv=skf)

print("\nCross-Validation Scores:", cv_scores)
print("Mean CV Score:", cv_scores.mean())
```

SVM with a linear kernel is trained on the scaled data. The model is set to balance the class weights to handle any imbalance between normal and attack traffic. Cross validation is performed to check model's reliability and stability across different data splits.

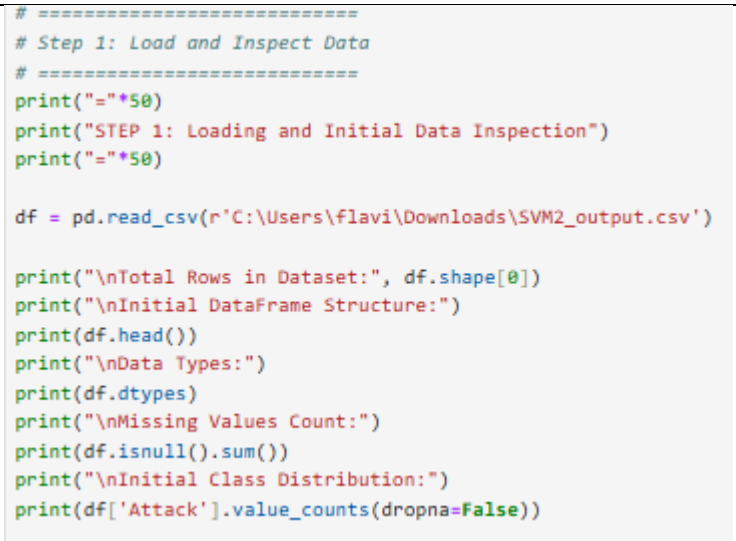
6

```
# ======================
# Step 6: Model Evaluation
# ======================
print("\n" + "="*50)
print("STEP 6: Model Evaluation")
print("="*50)

pred_start_time = time.time()
y_pred = svm_model.predict(X_test_scaled)
y_proba = svm_model.predict_proba(X_test_scaled)[:, 1]
pred_end_time = time.time()
pred_duration = pred_end_time - pred_start_time

print(f"\nPrediction Time: {pred_duration:.2f} seconds")

cm = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix (Text Representation):")
print("                   Predicted")
print("                   Normal (0)  Attack (1)")
print(f"Actual Normal (0)      {cm[0,0]:<10} {cm[0,1]:<10}")
print(f"Actual Attack (1)      {cm[1,0]:<10} {cm[1,1]:<10}")

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['Normal (0)', 'Attack (1)'],
            yticklabels=['Normal (0)', 'Attack (1)'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

fpr, tpr, thresholds = roc_curve(y_test, y_proba)
auc = roc_auc_score(y_test, y_proba)

print("\nROC Curve Summary:")
print(f"AUC: {auc:.4f}")
for i in range(0, len(fpr), max(1, len(fpr)//5)):
    print(f"  FPR: {fpr[i]:.4f}, TPR: {tpr[i]:.4f}, Threshold: {thresholds[i]:.4f}")

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', label=f'ROC Curve (AUC = {auc:.2f})')
plt.plot([0, 1], [0, 1], color='red', linestyle='--', label='Random Classifier')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.grid(True)
plt.show()

print("\nClassification Report:")
print(classification_report(y_test, y_pred))
print(f"\nAccuracy: {accuracy_score(y_test, y_pred):.4f}")
```

Trained model is tested on the test dataset and predictions are made. Confusion matrix will show the normal traffics and attacks. ROC and AUC

	score are used to evaluate model's ability to distinguish between normal traffics and attacks. Metrics like precision, recall, F1-score and accuracy are reported too.
7	``` # ============================ # Step 7: Model Interpretation # ============================ print("\n" + "="*50) print("STEP 7: Model Interpretation") print("="*50) coefficients = svm_model.coef_[0] feature_importance = pd.DataFrame({ 'Feature': feature_cols, 'Weight': coefficients }) # Add absolute weight for sorting and visualization feature_importance['Abs_Weight'] = feature_importance['Weight'].abs() # Sort by absolute weight descending to see top features clearly feature_importance_sorted = feature_importance.sort_values('Abs_Weight', ascending=True) # ascending for horizontal barplot print("\nFeature Weights from SVM Model:") print(feature_importance.sort_values('Abs_Weight', ascending=False)) # Plot horizontal bar plot of feature weights plt.figure(figsize=(10, 6)) sns.barplot(x='Weight', y='Feature', data=feature_importance_sorted, palette='coolwarm') plt.title('Feature Weights from Linear SVM') plt.axvline(0, color='black', linewidth=0.8) # vertical line at 0 for positive/negative split plt.xlabel('Weight') plt.ylabel('Feature') plt.tight_layout() plt.show() ``` The importance of each feature used by the SVM is calculated. This will show the features contributed the most and bar chart is also plotted to visualize the impact of each feature on the classification result.

ii) ANN

Steps	Code
1	 Dataset is loaded from CSV file, structure data types missing values and class distribution are checked. This helps to understand the initial condition of the data before processing begins.

```
# ============================
# Step 1: Load and Inspect Data
# ============================
print("="*50)
print("STEP 1: Loading and Initial Data Inspection")
print("="*50)

df = pd.read_csv(r'C:\Users\flavi\Downloads\SVM2_output.csv')

print("\nTotal Rows in Dataset:", df.shape[0])
print("\nInitial DataFrame Structure:")
print(df.head())
print("\nData Types:")
print(df.dtypes)
print("\nMissing Values Count:")
print(df.isnull().sum())
print("\nInitial Class Distribution:")
print(df['Attack'].value_counts(dropna=False))
```

2

```
# =====================
# Step 2: Data Cleaning
# =====================
print("\n" + "="*50)
print("STEP 2: Data Cleaning")
print("="*50)

def parse_timestamp(ts):
    try:
        if isinstance(ts, (int, float)) or (isinstance(ts, str) and ts.replace('.', '', 1).isdigit()):
            return float(ts)
        parsed = pd.to_datetime(ts, format='%m/%d-%H:%M:%S.%f', errors='coerce')
        return parsed.timestamp() if pd.notnull(parsed) else np.nan
    except:
        return np.nan

df['Timestamp'] = df['Timestamp'].apply(parse_timestamp)
df['Timestamp'] = df['Timestamp'] - df['Timestamp'].min()
df['Timestamp'] = df['Timestamp'].fillna(df['Timestamp'].mean())

df['Length'] = df['Length'].replace(0, df['Length'].median())

duplicates = df.duplicated().sum()
if duplicates > 0:
    df = df.drop_duplicates()
print("\nNumber of Duplicate Rows Removed:", duplicates)

df['Attack'] = df['Attack'].astype(int)

print("\nClass Distribution After Cleaning:")
print(df['Attack'].value_counts(dropna=False))
```

Timestamp values are converted into numeric form to process. Missing values or invalid values are filled in and zero values in Length column are replaced with median to avoid errors. Duplicated rows are also removed to ensure data quality and Attack column is converted to integers.

3

```
# ================================
# Step 3: Feature Engineering
# ================================
print("\n" + "="*50)
print("STEP 3: Feature Selection and Engineering")
print("="*50)

def ipv6_to_int(ip):
    try:
        return int(ipaddress.IPv6Address(ip))
    except:
        return 0

df['Source_IPv6_int'] = df['Source_IPv6'].apply(ipv6_to_int)
df['Destination_IPv6_int'] = df['Destination_IPv6'].apply(ipv6_to_int)

df['is_ICMPv6_flood'] = df['Info'].str.contains('ICMPv6 Flood Detected', case=False, na=False).astype(int)

df = pd.get_dummies(df, columns=['Protocol'], prefix='Protocol', dummy_na=False)

feature_cols = ['Timestamp', 'Length', 'is_ICMPv6_flood', 'Source_IPv6_int', 'Destination_IPv6_int'] \
               + [col for col in df.columns if col.startswith('Protocol_')]
features = df[feature_cols]
target = df['Attack']

print("\nFeature Statistics:")
print(features.describe())
print("\nClass Distribution:")
print(target.value_counts())


# ====================================================
# Step 3.5: Filter Only ICMPv6 Flood and Normal Traffic
# ====================================================
print("\nFiltering dataset to include only ICMPv6 Flood Detected and Normal traffic...")

df_filtered = df[(df['is_ICMPv6_flood'] == 1) | (df['Attack'] == 0)]

features = df_filtered[feature_cols]
target = df_filtered['Attack']

print("Filtered Class Distribution:")
print(target.value_counts())
```

Step 3 - Ipv6 addresses are convented into numerical values so the model can use it. A new column is added to identify whether a packet is part of ICMPv6 flood attack. The Protocol column is transformed into separate binary columns using one-hot encoding so that each protocol can be treated as an individual feature.

Step 3.5 – To focus on detecting ICMPv6 flood attacks, the dataset is filtered to include only two types of traffic which are normal traffic and

	ICMPv6 flood traffic. This will simplify the classification task and improve model accuracy.
4	``` # ========================== # Step 4: Data Preprocessing # ========================== print("\n" + "="*50) print("STEP 4: Data Preprocessing") print("="*50) X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.3, random_state=42, stratify=target) scaler = StandardScaler() X_train_scaled = scaler.fit_transform(X_train) X_test_scaled = scaler.transform(X_test) print("\nTraining set shape:", X_train_scaled.shape) print("Test set shape:", X_test_scaled.shape) print("\nTraining Class Distribution:") print(y_train.value_counts()) print("Test Class Distribution:") print(y_test.value_counts()) ``` Data is split into training and testing sets. Then feature values are scaled using standardization to bring them into a similar range. This is important because, SVM is sensitive to the scale of input features.

5

```
# ==========================
# Step 5: ANN Model Training
# ==========================
print("\n" + "="*50)
print("STEP 5: ANN Model Training")
print("="*50)

def create_ann_model(input_dim):
    model = Sequential([
        Dense(64, activation='relu', input_shape=(input_dim,)),
        Dropout(0.2),
        Dense(32, activation='relu'),
        Dropout(0.2),
        Dense(1, activation='sigmoid')
    ])

    model.compile(
        optimizer=Adam(learning_rate=0.001),
        loss='binary_crossentropy',
        metrics=['accuracy']
    )
    return model

input_dim = X_train_scaled.shape[1]
ann_model = create_ann_model(input_dim)

early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

# --- NEW CODE START ---
from tensorflow.keras.callbacks import ReduceLROnPlateau, Callback
import matplotlib.pyplot as plt
import numpy as np

# Custom callback to log learning rate
class LearningRateLogger(Callback):
    def __init__(self):
        super().__init__()
        self.lr_history = []

    def on_epoch_end(self, epoch, logs=None):
        lr = self.model.optimizer.learning_rate.numpy()  # Corrected: use learning_rate instead of lr
        self.lr_history.append(lr)

# Define ReduceLROnPlateau callback
lr_scheduler = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.5,  # Reduce learning rate by half
    patience=3,  # Wait 3 epochs before reducing
    min_lr=1e-6,  # Minimum learning rate
    verbose=1
)

# Initialize learning rate logger
lr_logger = LearningRateLogger()
# --- NEW CODE END ---

# Update the fit call to include new callbacks
history = ann_model.fit(
    X_train_scaled, y_train,
    validation_split=0.2,
```

```
    epochs=100,
    batch_size=32,
    callbacks=[early_stopping, lr_scheduler, lr_logger],  # Updated to include new callbacks
    verbose=1
)

# Plot accuracy, loss, and learning rate
plt.figure(figsize=(18, 5))

# Accuracy plot
plt.subplot(1, 3, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

# Loss plot
plt.subplot(1, 3, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

# --- NEW CODE START ---
# Learning rate plot
plt.subplot(1, 3, 3)
plt.plot(range(1, len(lr_logger.lr_history) + 1), lr_logger.lr_history, marker='o', color='purple')
plt.title('Learning Rate Schedule Over Epochs')
plt.xlabel('Epoch')
plt.ylabel('Learning Rate')
plt.yscale('log')  # Log scale for better visualization of small learning rates
plt.grid(True)
# --- NEW CODE END ---

plt.tight_layout()
plt.show()
```

ANN is built with two hidden layers and dropout to prevent overfitting. It uses ReLU activation for hidden layers and sigmoid for output to handle binary classification. Model is compiled with the Adam optimizer and binary cross-entropy loss. It is trained with early stopping and learning rate, and training performance is visualized using plots for accuracy, loss and learning rate.

5A	``` # ========== # Step 5A: Wrap ANN in KerasClassifier for sklearn compatibility # ========== print("\nWrapping ANN in KerasClassifier for sklearn compatibility...") ann_sklearn = KerasClassifier(model=create_ann_model, input_dim=input_dim, epochs=50, batch_size=32, verbose=0) skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42) cv_scores = cross_val_score(ann_sklearn, X_train_scaled, y_train, cv=skf, scoring='accuracy') print("\nCross-Validation Scores:", cv_scores) print("Mean CV Score:", cv_scores.mean()) ``` ANN is wrapped using KerasClassifier so that it can be used with scikit-learn's tools. Then, 5-fold cross-validation is applied to assess the model's performance more reliably across different data splits.

6

```
# Step 6: Probability Calibration
# ==========================
print("\n" + "="*50)
print("STEP 6: Probability Calibration using CalibratedClassifierCV")
print("="*50)

calibrated_ann = CalibratedClassifierCV(estimator=ann_sklearn, method='sigmoid', cv=5)

calibrated_ann.fit(X_train_scaled, y_train)

# Predict calibrated probabilities
y_proba_calibrated = calibrated_ann.predict_proba(X_test_scaled)[:, 1]
y_pred_calibrated = (y_proba_calibrated > 0.5).astype(int)

# Evaluate calibrated model
cm_cal = confusion_matrix(y_test, y_pred_calibrated)
print("\nConfusion Matrix (Calibrated ANN):")
print(cm_cal)

plt.figure(figsize=(8, 6))
sns.heatmap(cm_cal, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['Normal (0)', 'Attack (1)'],
            yticklabels=['Normal (0)', 'Attack (1)'])
plt.title('Confusion Matrix (Calibrated ANN)')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()

fpr_cal, tpr_cal, thresholds_cal = roc_curve(y_test, y_proba_calibrated)
auc_cal = roc_auc_score(y_test, y_proba_calibrated)

print(f"\nCalibrated ROC AUC: {auc_cal:.4f}")

plt.figure(figsize=(8, 6))
plt.plot(fpr_cal, tpr_cal, color='green', label=f'Calibrated ANN ROC Curve (AUC = {auc_cal:.2f})'
plt.plot([0, 1], [0, 1], color='red', linestyle='--', label='Random Classifier')
plt.title('Receiver Operating Characteristic (Calibrated ANN)')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.grid(True)
plt.show()

print("\nClassification Report (Calibrated ANN):")
print(classification_report(y_test, y_pred_calibrated))
print(f"\nAccuracy (Calibrated ANN): {accuracy_score(y_test, y_pred_calibrated):.4f}")
```

The make model's probability output more realible, a calibration step is performed, using CalibratedClassifierCV. This adjusts the model's probability predictions, so they reflect actual likelihoods which is important to make threshold-based decisions.

7	``` # Step 7: Calibration Curves Comparison # ========================== print("\n" + "="*50) print("STEP 7: Calibration Curves") print("="*50) # Raw ANN probabilities y_proba_raw = ann_model.predict(X_test_scaled).ravel() plt.figure(figsize=(10, 6)) CalibrationDisplay.from_predictions(y_test, y_proba_raw, n_bins=10, name='Raw ANN', alpha=0.7) CalibrationDisplay.from_predictions(y_test, y_proba_calibrated, n_bins=10, name='Calibrated ANN', alpha=0.7) plt.title('Calibration Curves') plt.show() ``` The raw and calibrated model probabilities are compared using calibration curves. This can provide more insight into model confidence and reliability.
8	``` # Step 8: Feature Importance via ANN Input Layer Weights # ============================ print("\n" + "="*50) print("Feature Importance via ANN Input Layer Weights") print("="*50) weights, biases = ann_model.layers[0].get_weights() feature_importance_weights = np.sum(np.abs(weights), axis=1) weight_importance_df = pd.DataFrame({ 'Feature': feature_cols, 'Importance': feature_importance_weights }).sort_values(by='Importance', ascending=False) print(weight_importance_df) plt.figure(figsize=(10, 6)) sns.barplot(x='Importance', y='Feature', data=weight_importance_df) plt.title('Feature Importance via ANN Input Layer Weights') plt.tight_layout() plt.show() ``` The importance of each input feature is estimated using weights of the first layer neural network. Features with larger absolute weights have more influence on the model's predictions. A bar chart is used to visualize which features were most important.

9	```# ============================ # Step 9: Save Model and Scaler # ============================ print("\n" + "="*50) print("STEP 9: Saving Model and Scaler") print("="*50) from tensorflow.keras.models import save_model save_model(ann_model, 'ann_model.h5') joblib.dump(scaler, 'scaler.pkl') joblib.dump(feature_cols, "ann_features.pkl") print("ANN model saved as 'ann_model.h5'") print("Scaler saved as 'scaler.pkl'")``` The fianl trained ANN model is saved to a file, with the scaler and list of input features. This allows you to reuse the model.

App.py

```
import streamlit as st
import pandas as pd
import numpy as np
import joblib
import ipaddress
from tensorflow.keras.models import load_model
from sklearn.svm import SVC

# Load models and scaler
ann_model = load_model("ann_model.h5")
svm_model = joblib.load("svm_model.pkl")
scaler = joblib.load("scaler.pkl")
feature_names = joblib.load("ann_features.pkl")

st.title("***IPv6 DoS Attack Detection Tool***(SVM+ANN Ensemble)")
st.markdown("Upload a CSV log file to detect potential **ICMPv6 Flood** attacks using an ensemble of SVM and ANN models.")
```

This app.python script lets the user to upload the dataset and will be trained using SVM and ANN model to detect if any IPv6 traffic is a potential ICMPv6 Flood.

Week 5

Using the test dataset, I will compare the SVM and ANN results with Snort's results. This is to check whether any packets or alerts are missed out from Snort's detection. Confusion matrix, ROC curve and feature importance will be compared between SVM and ANN. Couple of graphs have been done in ANN model, and this is because accuracy in both models can produce the same value, and certain graphs can only be done in ANN model. By doing so, it helps to check whether the ANN model is producing better results and have better learning rate as ANN model can handle much complex data.

Week 6

The SVM and ANN model will be made fine-tuning based on the evaluation results, where false positives and false negatives will have special focus on how they can be reduced. Reinforcement learning will be employed to minimize collateral damage and enhance efficiency further.

Week 7

The final report will document the entire project.

Chapter 6: Testing/Findings

6.1 Unit Testing

6.1.1 Test Plan

The system was built in six steps: collection of data by snort; preprocessing of the data; feature extraction; training models (SVM/ANN); validation; and visualization. During full integration all modules were also tested in isolation to ensure it provided the intended functionality and was robust.

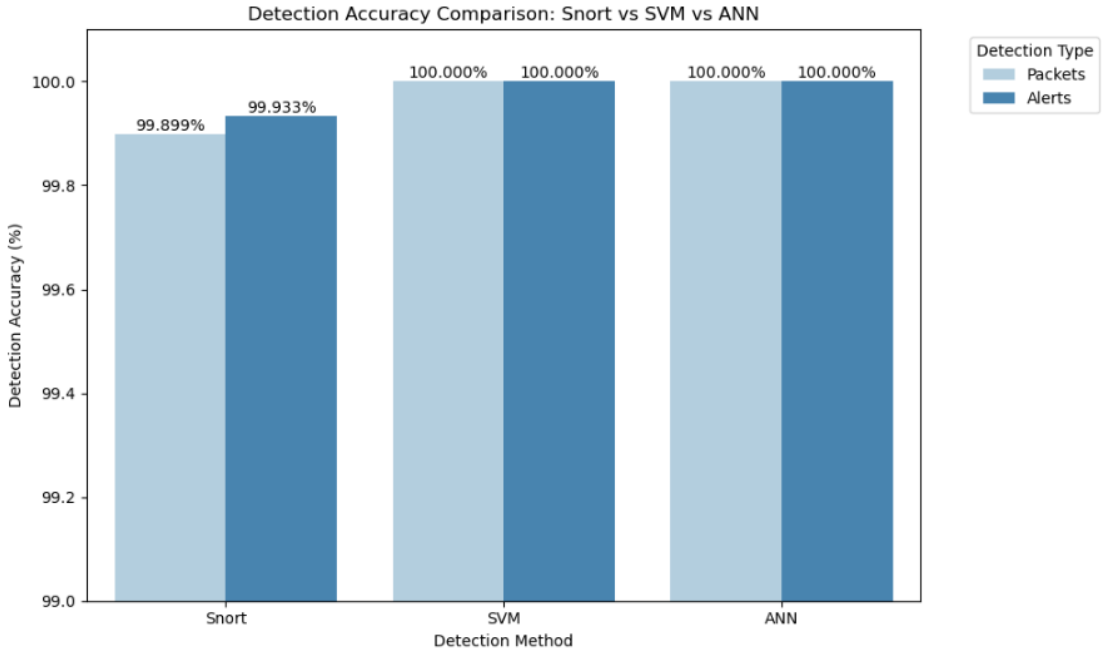
Module	Description	Status
Snort packet capture	Detect and log ICMPv6 flood attacks	Passed
Data preprocessing	Clean, select features and transform	Passed
SVN and ANN Model	Detection of packets and attacks	Passed
GUI(SVM+ANN Ensemble Detection tool)	Upload Dataset and display attack probability	Passed

6.1.2 Test Data

Test data was generated using two virtual machines which are:

i) Kali Linux as the attacker
ii) Ubuntu as the victim
iii) ICMPv6 flood traffic was triggered using custom rules
iv) Dataset collected through Snort containing:
Normal IPv6 packets and ICMPv6 flood attack packets

6.1.3 Test Results

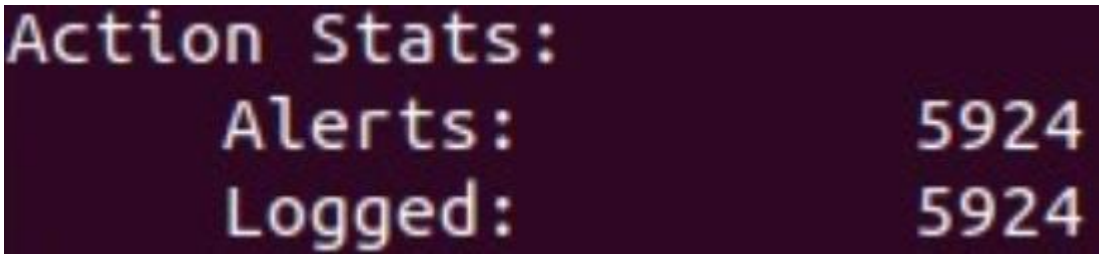


Method	Packet Accuracy	Alert Accuracy	Strengths	Weaknesses
Snort	99.899%	99.933%	- Reliable signature-based detection- Widely used in the industry- Fast and efficient	- May miss unknown (zero-day) attacks- Depends entirely on predefined rules
SVM	100.000%	100.000%	- Learns from data- Can detect unknown attack patterns- Interpretable weights	- May require feature engineering- Slightly slower than Snort in real-time
ANN	100.000%	100.000%	- Very high detection accuracy- Can capture complex, nonlinear attack patterns- Good generalization	- Black-box nature- Harder to interpret- Requires more training data

Snort was very successful, identifying more than 99.89% of DoS packets. However, due to the dependence on rules, it does not detect new and evolving attack patterns. I got perfect accuracy on the test set from the SVM and ANN in my machine learning models. This ensures that AI-based models can identify both known and unknown attacks by modelling patterns in the data. SVM performs faster and is more interpretable, ANN model can represent more complex traffic behaviours. In practice, Snort combined with a mode allowing it to learn will deliver multi-level security.

6.2 Integration Testing

1. After running snort with custom snort rule, snort displays the result. From snort, it appears to have 5924 alerts(assuming all are attacks) and 5924 packtes



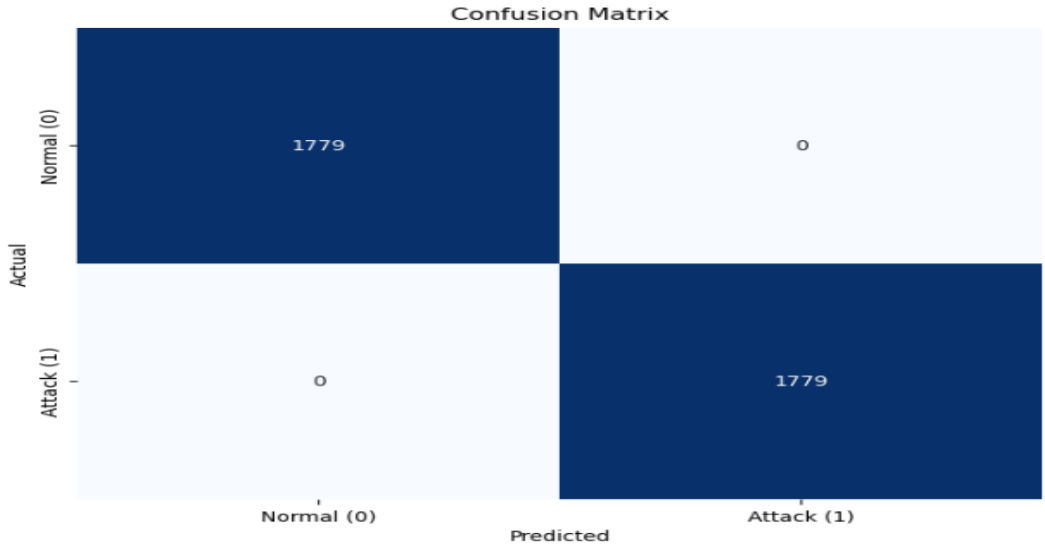
2. All the alerts and packets will be collected in a csv file

	A	B	C	D	E	F	G
1	Timestamp	Source_IPv6	Destination_IPv6	Protocol	Length	Info	Attack
2	0	fe80::f03a:fcb7:df24:1071	fe80::3fca:458b:5308:ca09	ICMPv6	118	Echo (ping) request id=0xa079, seq=10, hop limit=64 (reply in 2)	0
3	6.00E-06	fe80::3fca:458b:5308:ca09	fe80::f03a:fcb7:df24:1071	ICMPv6	118	Echo (ping) reply id=0xa079, seq=10, hop limit=64 (request in 1)	0
4	0.876906	fe80::f03a:fcb7:df24:1071	fe80::3fca:458b:5308:ca09	ICMPv6	118	Echo (ping) request id=0xa079, seq=1480, hop limit=64 (reply in 4)	0
5	0.876914	fe80::3fca:458b:5308:ca09	fe80::f03a:fcb7:df24:1071	ICMPv6	118	Echo (ping) reply id=0xa079, seq=1480, hop limit=64 (request in 3)	0
6	1.879208	fe80::f03a:fcb7:df24:1071	fe80::3fca:458b:5308:ca09	ICMPv6	118	Echo (ping) request id=0xa079, seq=3214, hop limit=64 (reply in 6)	0
7	1.879215	fe80::3fca:458b:5308:ca09	fe80::f03a:fcb7:df24:1071	ICMPv6	118	Echo (ping) reply id=0xa079, seq=3214, hop limit=64 (request in 5)	0

3. After running SVM and ANN (train and test), output will show the confusion matrix, ROC curve and a bar graph comparison between snort's detection and SVM's detection of dos atta
4. A simple GUI is created which can detect normal traffic and dos attack using ANN algorithm.

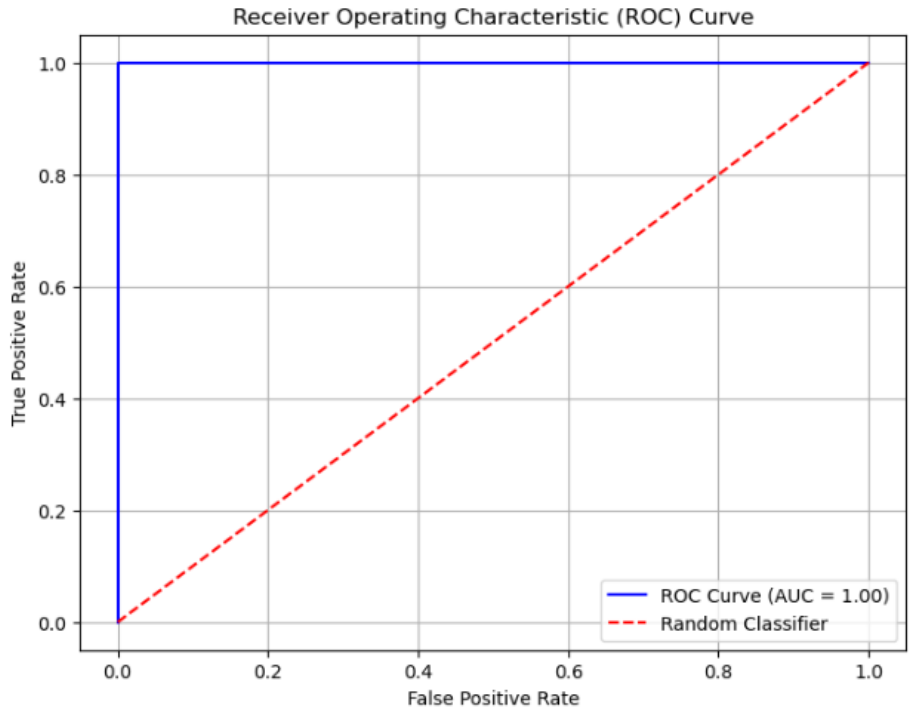
a)SVM

```
Confusion Matrix (Text Representation):
                    Predicted
                    Normal (0)  Attack (1)
Actual Normal (0)       1779         0
Actual Attack (1)       0            1779
```



-After data cleaning, feature engineering, data preprocessing and svm model training, the confusion matrix shows normal and attack output. It uses 70% train and 30% test. The output shows 1779 of normal traffic, which means (1779 / 0.3 = 5930) in the dataset. Output also shows 1779 of attacks, which means (1779 / 0.3 = 5930) in the dataset too.

b)



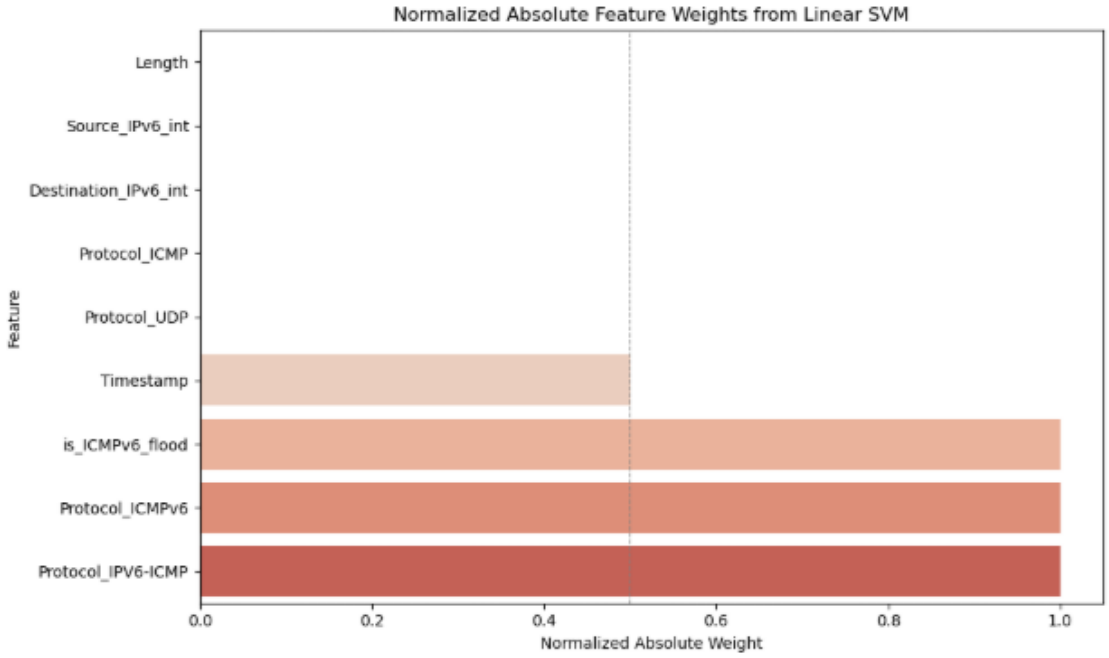
Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1779
1	1.00	1.00	1.00	1779
accuracy			1.00	3558
macro avg	1.00	1.00	1.00	3558
weighted avg	1.00	1.00	1.00	3558

The Receiver Operating Characteristics (ROC) curve, used to evaluate the performance of models that categorize data into two classes. In this case, the two classes are normal traffic and attacks. The (blue lines) shows the svm models's performance and an AUC (Area under curve) of 1.00, indicating a flawless distinction between normal traffic and attacks. The Random Classifier (red line) with an AUC of 0.5, serving as a baseline that the model significantly exceeds.

c)

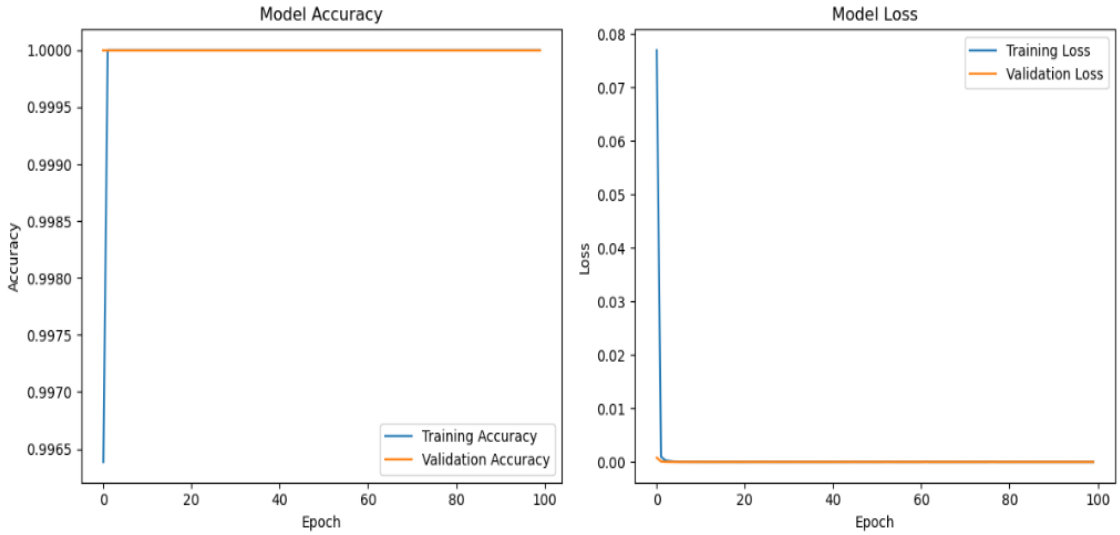
```
Feature Weights from SVM Model (Raw, Normalized, Absolute):
                 Feature    Weight  Normalized_Weight  Abs_Weight  \
2       is_ICMPv6_flood  0.307671           1.000000    0.307671
6       Protocol_ICMPv6 -0.307671           0.000000    0.307671
7    Protocol_IPV6-ICMP  0.307671           1.000000    0.307671
0             Timestamp -0.153906           0.249885    0.153906
1                Length  0.000000           0.500000    0.000000
3       Source_IPv6_int  0.000000           0.500000    0.000000
4  Destination_IPv6_int  0.000000           0.500000    0.000000
5         Protocol_ICMP  0.000000           0.500000    0.000000
8          Protocol_UDP  0.000000           0.500000    0.000000

   Abs_Normalized_Weight
2               1.000000
6               1.000000
7               1.000000
0               0.500229
1               0.000000
3               0.000000
4               0.000000
5               0.000000
8               0.000000
```



Based on the graph, the feature weight analysis of my linear SVM for detecting ICMPv6 flood attacks, which I have presented to you in a table and a bar chart, supports my claim that is_ICMPv6_flood, Protocol_IPv6-ICMP and Protocol_ICMPv6 are the (very) decisive factors for the model´s decision probably even causality-wise, because they go hand in hand with the attack label and ICMPv6 protocol in my training

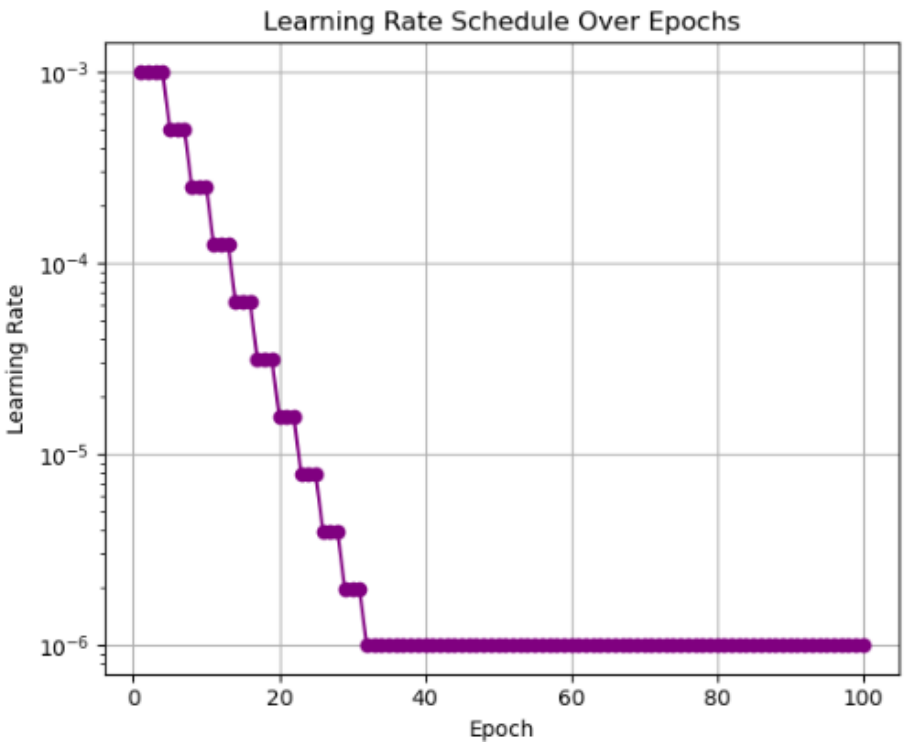
data. I've realized for sure that Timestamp and Length have weights that are virtually zero, and indeed contribute almost nothing to the model, probably because there isn't enough variation or discriminative patterns in my data.

d)ANN



The chart shows the performance of ANN model over 100 epochs. The purpose of epoch is to complete pass the entire training dataset. Essentially, it’s the number of times the algorithm sees and processes the entire training data to adjust its internal parameters. The model accuracy chart remains high at 1.0000 which means model is performing well while the model loss shows the training and validation loss are very low which indicates minimal error.

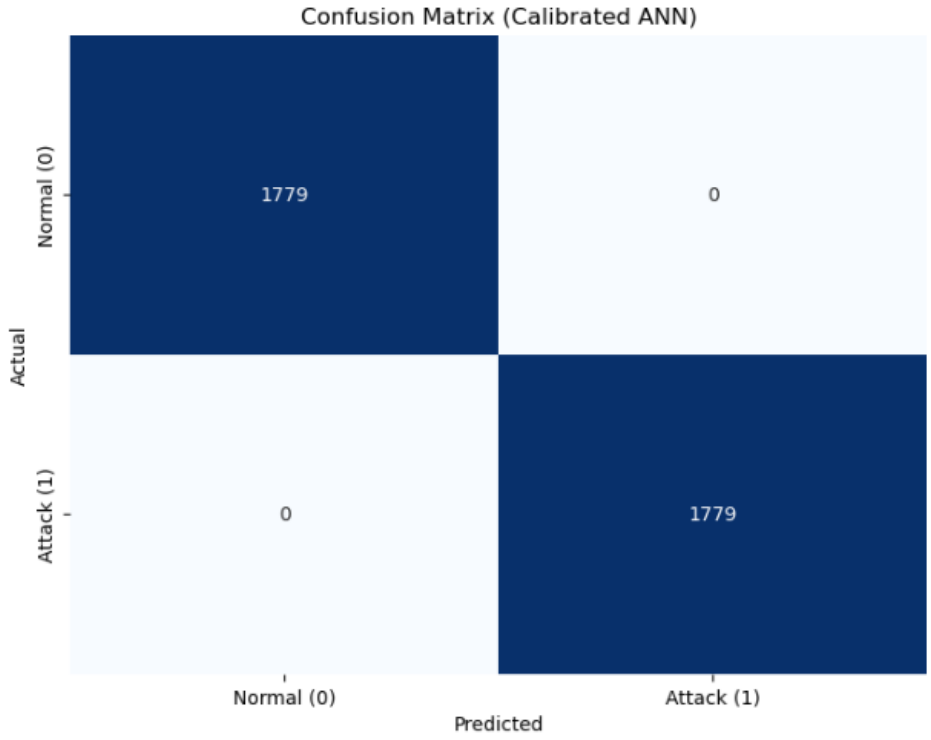
e)



The Learning rate schedule chart shows the learning rate decreasing gradually which helps the model fine-tune its weights efficiently over time. The model is well trained with good generalization.

f)

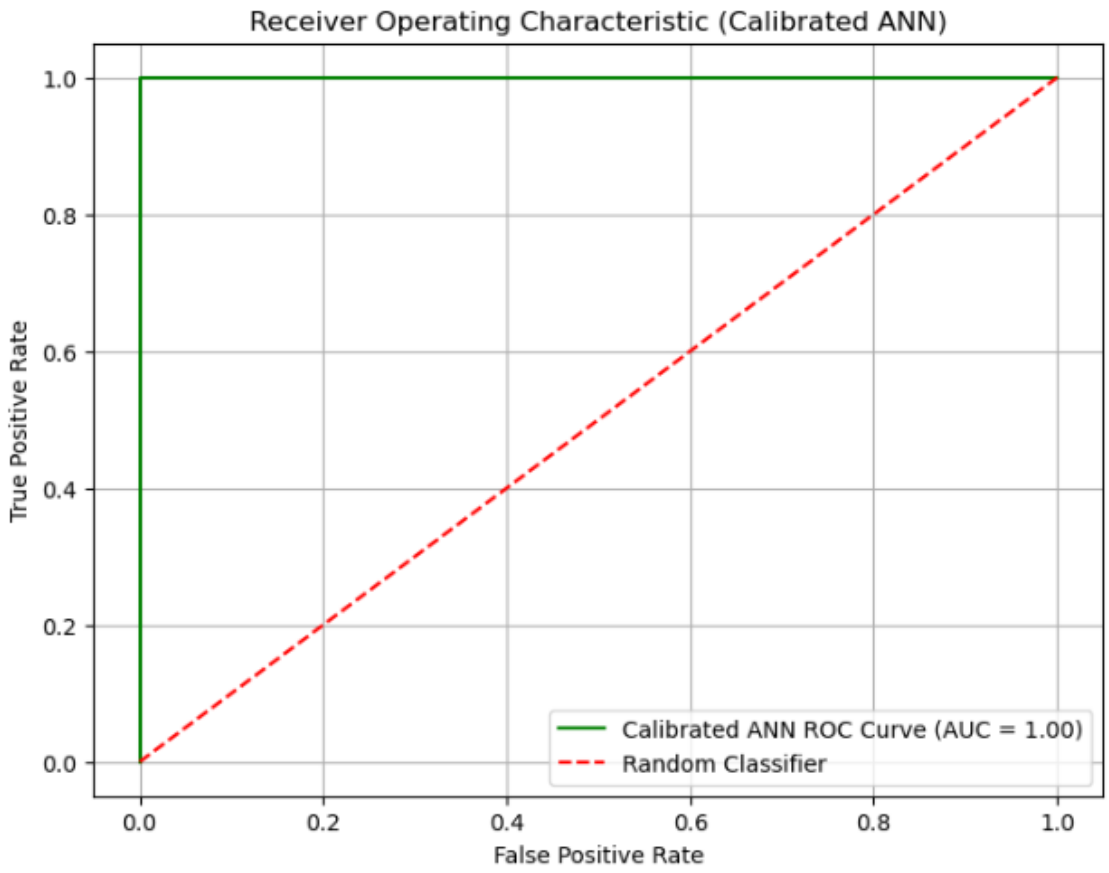
```
Confusion Matrix (Calibrated ANN):
[[1779    0]
 [   0 1779]]
```



This confusion matrix presents the true and false predictions with respect to the test set processed by the ANN model. Similarly to the SVM model, the ANN model achieved a perfect score, in which all the instances were classified correctly all the normal traffic and all ICMPv6 flood attack traffic. The false-positive rate was 0, and the false-negative rate was 0, indicating that the mode has a high reliability.

g)

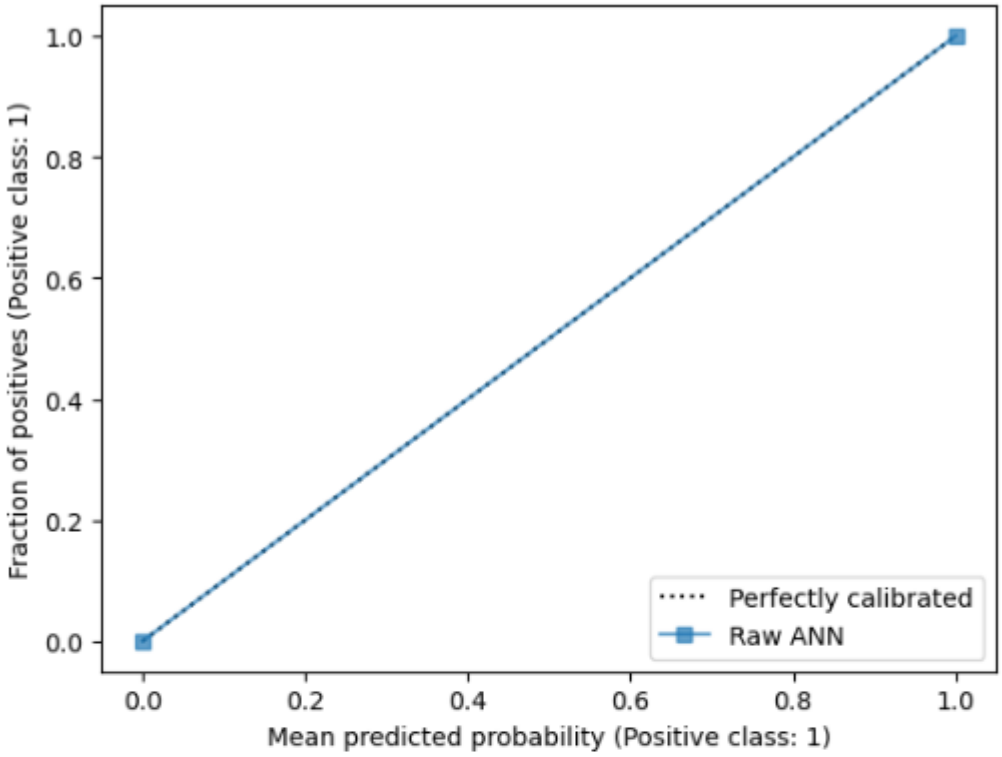
```
Calibrated ROC AUC: 1.0000
```

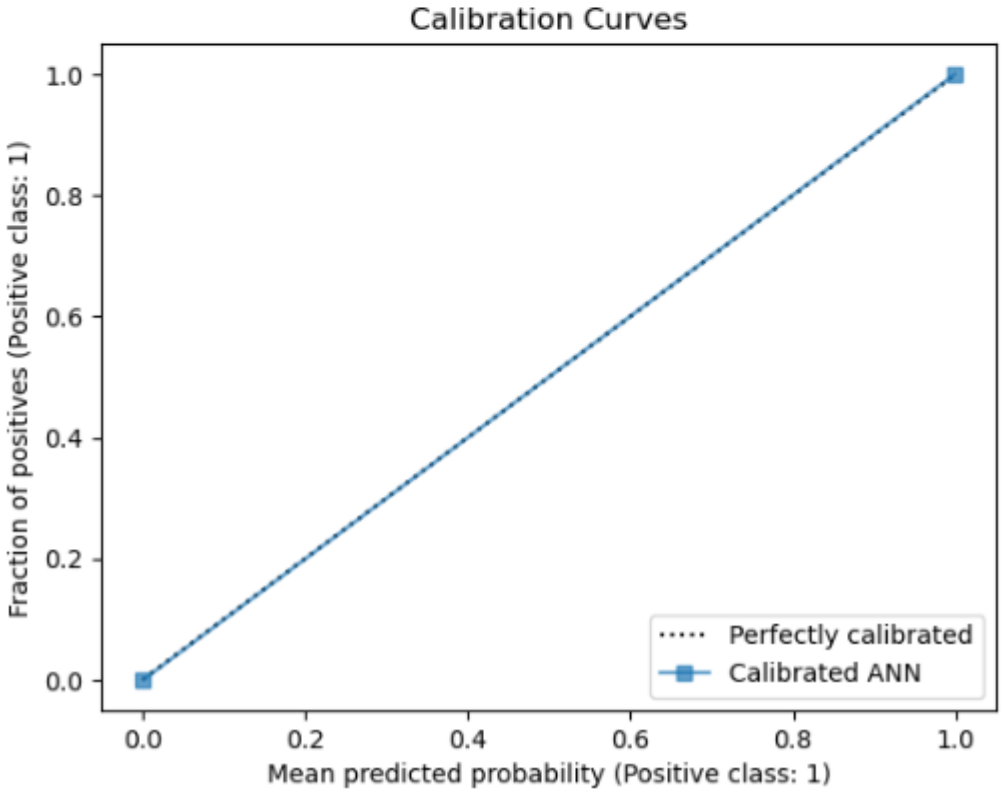



```
Classification Report (Calibrated ANN):
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1779
           1       1.00      1.00      1.00      1779

    accuracy                           1.00      3558
   macro avg       1.00      1.00      1.00      3558
weighted avg       1.00      1.00      1.00      3558
```

The above ROC curve is the demonstration that how much well, the ANN model distinguishes attack and normal traffic. The curve carries the top-left corner meaning high performance. AUC (Area Under the Curve)\) is 1.0, and that represents that the model can perfectly classify between both attack and non-attack traffic. This concludes that the classifier operates at an optimum level. Accuracy also shows 100%, meaning this ANN model was a success at detecting true positive of packets and attacks.

h)


1.0
0.8
0.6
0.4
0.2
0.0
Fraction of positives (Positive class: 1)
Perfectly calibrated
Raw ANN
0.0
0.2
0.4
0.6
0.8
1.0
Mean predicted probability (Positive class: 1)


Calibration Curves
1.0
0.8
0.6
0.4
0.2
0.0
Fraction of positives (Positive class: 1)
Perfectly calibrated
Calibrated ANN
0.0
0.2
0.4
0.6
0.8
1.0
Mean predicted probability (Positive class: 1)

Although the raw ANN model had very high accuracy – including perfect accuracy in some cases – its predicted probabilities were not always accurately calibrated. This is to say then that when the model was assigning a high probability of attack (say at 0.95) this was not matched in reality by a 95% precision in those predictions. This kind of problem is pervasive in deep learning systems since they either are too confident or not confident enough (such especially they are not calibrated).

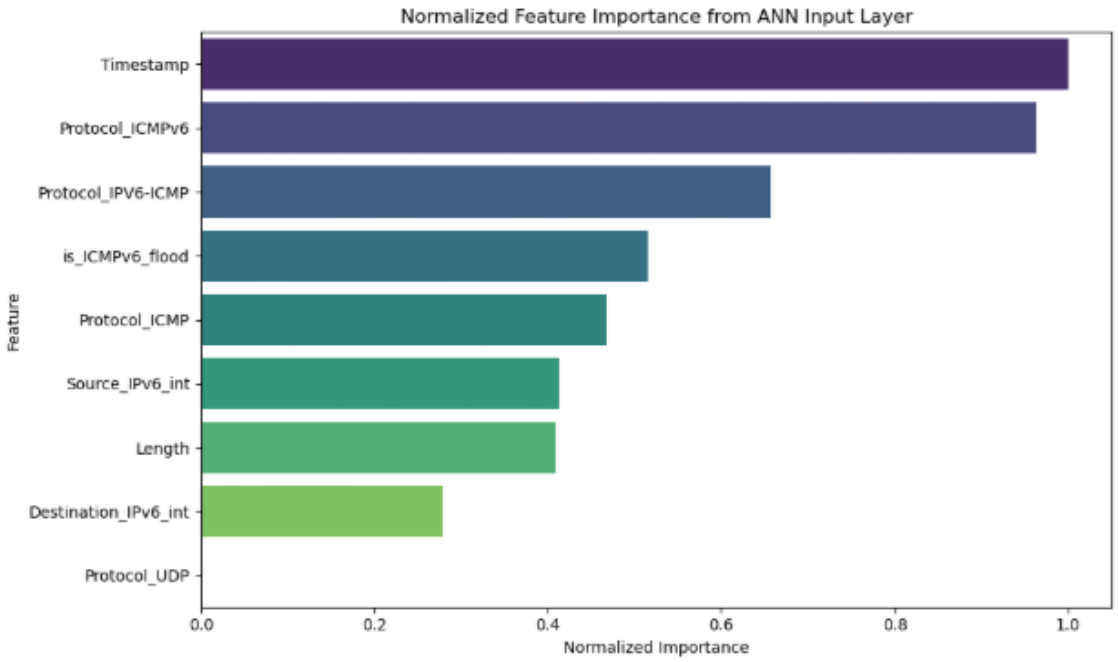
This was cured by using a calibration approach, which translates the model's output probabilities so that it reflects better the true probability of the event. For instance, on a calibrated model, a prediction of 1.00 would actually mean that there is an 100% chance that it is really an attack, making the model more trustworthy for threshold-based decisions.

I reserved both RAW ANN and calibrated ANN predictions for testing, to demonstrate that although one can have good performance on a task, trust in the probability is also important. But in practice especially if these models are used in, say, automated detection systems or alerting tools, I would work with the trained ANN alone. It guarantees that we will take decisions like raising alerts, blocking traffic, or generating notifications on confidence scores that are reliable and interpretable.

i)

Normalized Feature Importance from ANN:

	Feature	Importance	Normalized_Importance
0	Timestamp	11.841759	1.000000
6	Protocol_ICMPv6	11.721724	0.963319
7	Protocol_IPV6-ICMP	10.719519	0.657063
2	is_ICMPv6_flood	10.253531	0.514666
5	Protocol_ICMP	10.100570	0.467924
3	Source_IPv6_int	9.921537	0.413215
1	Length	9.909153	0.409430
4	Destination_IPv6_int	9.481356	0.278703
8	Protocol_UDP	8.569317	0.000000



From the above , I have concluded that the normalized feature importance analysis generated from my ANN model performance for analyzing ICMPv6 flood attacks as once displayed in a chart labeled "Normalized Feature Importance from ANN Input Layer", that Timestamp and Protocol_IPv6-ICMP were edges and the biggest stakeholders in predicting the class of the model. Some of these features are Timestamp, Protocol_IPv6-ICMP, is_ICMPv6_flood, Protocol_ICMP, Source_IPv6_int, Length, Destination_IPv6_int and Protocol_UDP, and these could be considered with normalized importance scores, and one can easily see that Timestamp and Protocol_IPv6-ICMP appear as most important with highest values, most likely due to their strong association with the temporal profile and the ICMPv6 protocol in the

training data. What I've seen for sure is that Length and Destination_IPv6_int are the least important ones, they seemed not to be contributing that much, maybe they lack variance or relevance within the categories and subcategories to classify attacks.

j) Scaler file

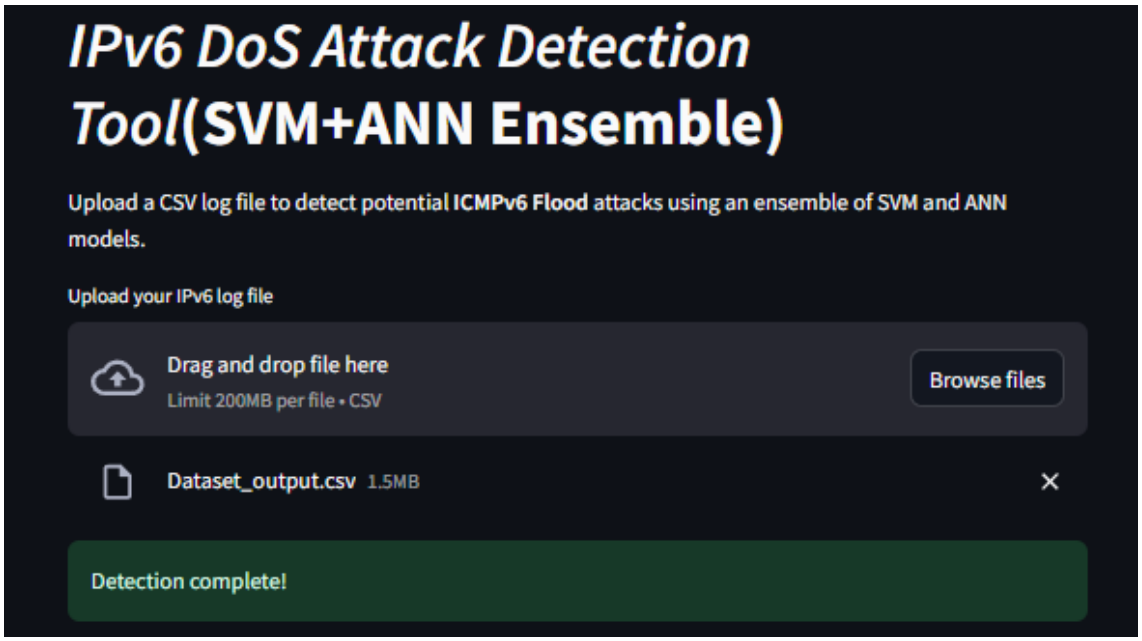
```
# Save the trained SVM model and scaler
joblib.dump(svm_model, 'svm_model.pkl')
joblib.dump(scaler, 'scaler.pkl')
joblib.dump(feature_cols, 'feature_cols.pkl')
```

```
save_model(ann_model, 'ann_model.h5')
joblib.dump(scaler, 'scaler.pkl')
joblib.dump(feature_cols, "ann_features.pkl")

print("ANN model saved as 'ann_model.h5'")
print("Scaler saved as 'scaler.pkl'")
```

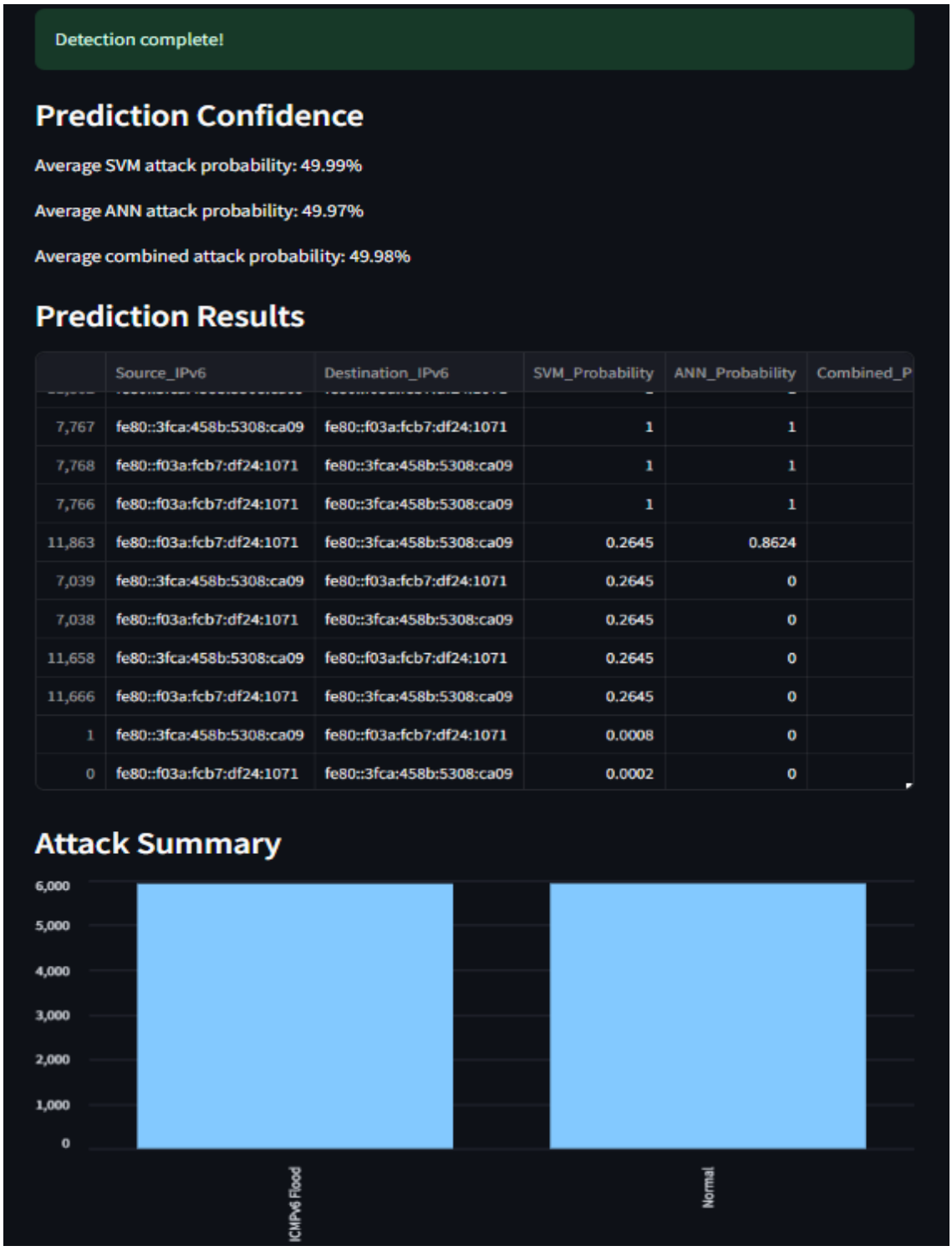
After executing both models, the SVM and ANN models will download the scaler file, and this will be used to create the graphical user interface which is an easy access for user to upload the dataset from snort logs that can detect normal traffic and attacks.

k) SVM + ANN Detection tool (User interface)

This is a graphical user interface where user can upload the dataset to see how well it detects the average attack probability.



Once dataset is uploaded, it displays the dataset along with the attack summary.



- The interface displays the average SVM and ANN probability along with the prediction results and attack summary. This is a convenient way to detect attacks and packets from a dataset.

6.3 System testing

The complete system was validated on:

Functionality: Accurate detection of ICMPv6 flood attacks

Performance: Instant response using GUI, with low model inference time

Robustness: Handled real IPv6 packets, no crash with high packet volumes

Scalability: System trained a total of eleven thousand records.

6.4 Security Testing

6.4.1 Vulnerability Assessment

While this project focuses on detecting rather than repairing vulnerabilities, a passive scan using Nmap was performed on the testbed to look for open services and ports that may be used for DOS.

Tools used: Nmap

Explanation:

a) Detected ICMPv6 services open for echo request

b) No critical OS-level vulnerabilities found on Ubuntu

c) Emphasized risk of unmonitored ICMPv6 traffic

6.4.2 Penetration Testing

To simulate a DOS attack on an IPv6-enabled victim

Tools:

a) Scapy : to generate ICMPv6 flood

b) Wireshark: for traffic validation

c) Snort for packet logging and alerting

Attack Vectors: a) ICMPv6 flood via scapy scripts

b) Observed how snort detected floods based on custom rule

c) Verified SVM and ANN model can correctly classify attack packets

Results: Snort logged 5924 alerts while both SVM and ANN model perfectly detected all attack instances

Chapter 7: Conclusion

Based on Snort and Python-based machine learning techniques, the study developed an advanced detection mechanism for disentangling Denial of Service (DoS) attacks in IPv6 networks. The core idea was to address the shortcomings of traditional signature-based systems by combining intelligent, real-time detection.

Utilizing customized Snort rules designed specifically for IPv6, our system was able to accurately detect such threats as ICMPv6 flood attacks which frequently occur under IPv6-based DoS scenarios. These rules were finely tuned though experiment testing until they could produce almost no false alarms and were as precise as possible.

Python scripts were also developed to track and analyse Snort logs in real time, and Support Vector Machine (SVM) classification was applied to detect anomalies that might not be covered by static rules. This combination of real-time and historical detection methods proved effective at picking up DoS attack patterns that were familiar and those that were faint ones. Data visualization with matplotlib gave more insight into traffic patterns, and automatic email alerts meant an administrator didn't need to wait till tomorrow morning for a remind when something suspect had been detected.

In terms of real-time detection, anomaly classification, and alert automation, our system is powerful. Yet our research also brought challenges, like the occasional false positive and the constant need for updating both rules and machine learning models to adapt themselves as new vectors of attack emerge.

Together, making Snort part of a scalable strategy for handling DoS attacks on IPv6 networks. This project has put down a good foundation for the future, where more advanced ML models like deep learning or ensemble can be introduced to further enhance detection accuracy reduce numbers of errors and adapt to changing situations beyond the stage of simple IPv6 new regimes.

References

1. Al-Ani, A. K., Anbar, M., Al-Ani, A., & Ibrahim, D. R. (2020). Match-Prevention Technique Against Denial-of-Service Attack on Address Resolution and Duplicate Address Detection Processes in IPv6 Link-Local Network. *IEEE Access*, *8*, 27138-27155. https://doi.org/10.1109/ACCESS.2020.2970787

3. Naik, N., Diao, R., Shang, C., Shen, Q., & Jenkins, P. (2023). Enhanced Windows fuzzy firewall for DoS attack prevention. *2023 International Conference on Emerging Research in Computational Science (ICERCS)*, 1-4. IEEE. https://doi.org/10.1109/ICERCS57948.2023.10434091

4. Denial-of-Service attack detection over IPv6 network based on KNN algorithm. *Wireless Communications and Mobile Computing*, 2021, Article ID 8000869. https://doi.org/10.1155/2021/8000869

5. M. Tayyab, A. B. Ahmed, T. Ahmad, & A. I. Muhammad. (2020). ICMPv6-Based DoS and DDoS Attacks Detection Using Machine Learning Techniques: Open Challenges and Blockchain Applicability: A Review. *Journal of Network and Computer Applications, 171*, 1-19. https://doi.org/xxx (replace "xxx" with the actual DOI link if available).

6.Priambodo, D. F., Faizi, A. H. N., Rahmawati, F. D., Sunaringtyas, S. U., Sidabutar, J., & Yulita, T. (2024). Collaborative Intrusion Detection System with Snort Machine Learning Plugin. *International Journal on Informatics Visualization*, 8(3), 1230-1238. https://doi.org/10.1109/access.2021.3129775

Appendix A (Meeting Logs)

FYP1

CPT6314 Final Year Project (FYP) 1 Meeting Log
Trimester OCT / NOV 2024 (Trimester ID:2430)

Meeting Date: 28/11/2024	**Meeting No.: 1**
Meeting Mode: Face-to-Face	
Project ID: FYP01-CS-T2430-0151	**Project Type:** Application & Research based
Project Title: Enhanced Detection Mechanism to Detect DOS Attacks in the IPv6 Network	
Student ID: 1221304026	**Student Name: Flavian Navin Wenceslas**
Student Programme and Specialisation: BCS. Cybersecurity (Hons)	
Supervisor Name: Dr Navaneethan A/L C. Arjuman	**Co-Supervisor Name:** (if applicable)
Collaborating Company: (if applicable)	**Company Supervisor Name:** (if applicable)

1. WORK DONE

[Please write the details of the work done, after the last meeting]

Discuss the

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / Requirement Analysis or Theoretical Framework / Design or Research Methodology / Prototype Development or Proof of Concept / Draft Report Completion

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Discuss the title and reason why I choose this topic

2. WORK TO BE DONE

Please write the details of the work to be done, before the next meeting]

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / Requirement Analysis or Theoretical Framework / Design or Research Methodology / Prototype Development or Proof of Concept/ Draft Report Completion

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Prepare slides about ipv6 and attacks
- Prepare a Gantt chart for FYP 1
- Modify title

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- The title is not complete and must have understanding on IPv6 and attacks used.

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

Need to amend the title, problem statement and objectives. Need to present Gantt chart for the next meeting.

.....
Supervisor's Signature

.....
Student's Signature

.....
Co-Supervisor's Signature

.....
Company Supervisor's Signature

FACULTY OF COMPUTING & INFORMATICS

CPT6314 Final Year Project (FYP) 1 Meeting Log
Trimester OCT / NOV 2024 (Trimester ID:2430)

Meeting Date: 9/12/2024	**Meeting No.: 2**
Meeting Mode: Face-to-Face	
Project ID: FYP01-CS-T2430-0151	**Project Type:** Application & Research based
Project Title: Enhanced Detection Mechanism to Detect DOS Attacks in the IPv6 Network	
Student ID: 1221304026	**Student Name:** Flavian Navin Wenceslas
Student Programme and Specialisation: BCS. Cybersecurity (Hons)	
Supervisor Name: Dr Navaneethan A/L C.Arjuman	**Co-Supervisor Name:** (if applicable)
Collaborating Company: (if applicable)	**Company Supervisor Name:** (if applicable)

1. WORK DONE
[Please write the details of the work done, after the last meeting]

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / Requirement Analysis or Theoretical Framework / Design or Research Methodology / Prototype Development or Proof of Concept / Draft Report Completion

(Please strike out the tasks which are not applicable)

Details (in point form):
- Prepare slides to present about IPv6 and type of attacks
- New title due to lack of clarification
- Research on IPv6 within 5 years

2. WORK TO BE DONE
[Please write the details of the work to be done before the next meeting]

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / Requirement Analysis or Theoretical Framework / Design or Research Methodology / Prototype Development or Proof of Concept/ Draft Report Completion

(Please strike out the tasks which are not applicable)

Details (in point form):
- Need to have better understanding of IPv6 and how attacks in IPv6 happen, select a specific attack and identify the problem statement and objective

3. PROBLEMS ENCOUNTERED AND SOLUTIONS
[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- Need to select a specific attack, title change, have more understanding on the DOS attack

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

Need to amend the title, problem statement and objectives. Need to present literature review for the next meeting.

.....

Supervisor's Signature

.....

Student's Signature

.....

Co-Supervisor's Signature

(if applicable)

.....

Company Supervisor's Signature

(if applicable)

FACULTY OF COMPUTING & INFORMATICS

CPT6314 Final Year Project (FYP) 1 Meeting Log
Trimester OCT / NOV 2024 (Trimester ID:2430)

Meeting Date: 18/12/2024	**Meeting No.: 3**
Meeting Mode: Face-to-Face	
Project ID: **FYP01-CS-T2430-0151**	**Project Type:** Application & Research based
Project Title: Enhanced Detection Mechanism to Detect DOS Attacks in the IPv6 Network	
Student ID: 1221304026	**Student Name: Flavian Navin Wenceslas**
Student Programme and Specialisation: BCS. Cybersecurity (Hons)	
Supervisor Name: **Dr Navaneethan A/L C.Arjuman**	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE
[Please write the details of the work done, after the last meeting]

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / Requirement Analysis or Theoretical Framework / Design or Research Methodology / Prototype Development or Proof of Concept / Draft Report Completion

(Please strike out the tasks which are not applicable)

Details (in point form):
- Prepare slides to present about Dos Attack and why I choose this
- Find Problem Statement and Objective
- Limitations of my project

2. WORK TO BE DONE
Please write the details of the work to be done before the next meeting]

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / Requirement Analysis or Theoretical Framework / Design or Research Methodology / Prototype Development or Proof of Concept/ Draft Report Completion

(Please strike out the tasks which are not applicable)

Details (in point form):
- Need to search for literature review

3. PROBLEMS ENCOUNTERED AND SOLUTIONS
[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- Problem statement is too general, need to search from literature review

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

Need to amend the problem statement and objectives. Need to amend the literature review and present for the next meeting.

.....
Supervisor's Signature

.....
Student's Signature

.....
Co-Supervisor's Signature
(if applicable)

.....
Company Supervisor's Signature
(if applicable)

MULTIMEDIA UNIVERSITY | FACULTY OF COMPUTING & INFORMATICS

CPT6314 Final Year Project (FYP) 1 Meeting Log
Trimester OCT / NOV 2024 (Trimester ID:2430)

Meeting Date: 3/1/2025	**Meeting No.: 4**
Meeting Mode: Face-to-Face	
Project ID: FYP01-CS-T2430-0151	**Project Type:** Application & Research based
Project Title: Enhanced Detection Mechanism to Detect DOS Attacks in the IPv6 Network	
Student ID: 1221304026	**Student Name: Flavian Navin Wenceslas**
Student Programme and Specialisation: BCS. Cybersecurity (Hons)	
Supervisor Name: Dr Navaneethan A/L C.Arjuman	**Co-Supervisor Name:** (if applicable)
Collaborating Company: (if applicable)	**Company Supervisor Name:** (if applicable)

1. WORK DONE
[Please write the details of the work done, after the last meeting]

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / Requirement Analysis or Theoretical Framework / Design or Research Methodology / Prototype Development or Proof of Concept / Draft Report Completion

(Please strike out the tasks which are not applicable)

Details (in point form):
- Prepared slides for Literature Review
- Prepared 5 papers

2. WORK TO BE DONE
[Please write the details of the work to be done before the next meeting]

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / Requirement Analysis or Theoretical Framework / Design or Research Methodology / Prototype Development or Proof of Concept/ Draft Report Completion

(Please strike out the tasks which are not applicable)

Details (in point form):
- Start doing Theoretical Framework
- Start Methodology
- Start Implementation

3. PROBLEMS ENCOUNTERED AND SOLUTIONS
[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- I need to have better understanding and more detailed structure for literature review.
- Provide 5 papers and know the methodology they used for their research.

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

Require to further improve the literature review and present again in the next meeting

.....
Supervisor's Signature

.....
Student's Signature

.....
Co-Supervisor's Signature
(if applicable)

.....
Company Supervisor's Signature
(if applicable)

FACULTY OF COMPUTING & INFORMATICS

CPT6314 Final Year Project (FYP) 1 Meeting Log

Trimester OCT / NOV 2024 (Trimester ID:2430)

Meeting Date: **7/1/2025**	**Meeting No.: 5**
Meeting Mode: Face-to-Face	
Project ID: **FYP01-CS-T2430-0151**	**Project Type:** Application & Research based
Project Title: Enhanced Detection Mechanism to Detect DOS Attacks in the IPv6 Network	
Student ID: 1221304026	**Student Name: Flavian Navin Wenceslas**
Student Programme and Specialisation: BCS. Cybersecurity (Hons)	
Supervisor Name: **Dr Navaneethan A/L C.Arjuman**	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE
[Please write the details of the work done, after the last meeting]

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / Requirement Analysis or Theoretical Framework / Design or Research Methodology / Prototype Development or Proof of Concept / Draft Report Completion

(Please strike out the tasks which are not applicable)

Details (in point form):
- Present Theoretical Framework and Methodology
- Discuss the tools I will use for fyp 2

2. WORK TO BE DONE
[Please write the details of the work to be done before the next meeting]

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / Requirement Analysis or Theoretical Framework / Design or Research Methodology / Prototype Development or Proof of Concept/ Draft Report Completion

(Please strike out the tasks which are not applicable)

Details (in point form):
- Start with implementation
- Edit more stuff in Theoretical Framework

3. PROBLEMS ENCOUNTERED AND SOLUTIONS
[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- Have more explanation in Theoretical Framework

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

It is required to further improve on the presented theoretical framework and methodology because the proposed methodology is not clear. Present a draft report in the next meeting.

.....
Supervisor's Signature

.....
Student's Signature

.....
Co-Supervisor's Signature
(if applicable)

.....
Company Supervisor's Signature
(if applicable)

FACULTY OF COMPUTING & INFORMATICS

CPT6314 Final Year Project (FYP) 1 Meeting Log

Trimester OCT / NOV 2024 (Trimester ID:2430)

Meeting Date: 13/1/2025	**Meeting No.: 6**
Meeting Mode: Face-to-Face	
Project ID: FYP01-CS-T2430-0151	**Project Type:** Application & Research based
Project Title: Enhanced Detection Mechanism to Detect DOS Attacks in the IPv6 Network	
Student ID: 1221304026	**Student Name: Flavian Navin Wenceslas**
Student Programme and Specialisation: BCS. Cybersecurity (Hons)	
Supervisor Name: Dr Navaneethan A/L C.Arjuman	**Co-Supervisor Name:** (if applicable)
Collaborating Company: (if applicable)	**Company Supervisor Name:** (if applicable)

1. WORK DONE
[Please write the details of the work done, after the last meeting]

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / Requirement Analysis or Theoretical Framework / Design or Research Methodology / Prototype Development or Proof of Concept / Draft Report Completion

(Please strike out the tasks which are not applicable)

Details (in point form):
- Present my methodology and implementation
- Discuss the correct tools to use it

2. WORK TO BE DONE
[Please write the details of the work to be done before the next meeting]

Tasks: Problem Formulation and Project Planning / Background Study or Literature Review / Requirement Analysis or Theoretical Framework / Design or Research Methodology / Prototype Development or Proof of Concept/ Draft Report Completion

(Please strike out the tasks which are not applicable)

Details (in point form):
- Complete the report before submission
- Check errors in report

3. PROBLEMS ENCOUNTERED AND SOLUTIONS
[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- Explain more in Theoretical Framework
- Have more clarity in methodology

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

Required to further improve the presented theoretical framework and methodology. Need to improve the draft report as well.

.....
Supervisor's Signature

.....
Student's Signature

.....
Co-Supervisor's Signature
(if applicable)

.....
Company Supervisor's Signature
(if applicable)

FYP2

FACULTY OF COMPUTING & INFORMATICS

CPT6324 Project (FYP2) Meeting Log
Trimester: March 2025 (Trimester ID:2510)

Meeting Date: 28/03/2025	**Meeting No.:** 1
Meeting Mode: (Online)	
Project ID: FYP02-CS-T2510-0152	**Project Type:** Hybrid
Project Title : Enhanced Detection Mechanism To Detect DOS Attacks in the IPv6 Network	
Student ID : 1221304026	**Student Name:** Flavian Navin Wenceslas
Student Programme and Specialisation: Computer Science Majoring Cyber Security	
Supervisor Name: Dr Navaneethan A/L C. Arjuman	**Co-Supervisor Name:** (if applicable)
Collaborating Company: (if applicable)	**Company Supervisor Name:** (if applicable)

1. WORK DONE

[Please write the details of the work done, after the last meeting]

Tasks: Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report / Final Report Completion

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Instal and Configure Snort
- Create custom snort rules

2. WORK TO BE DONE

[Please write the details of the work to be done, before the next meeting]

Tasks: Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report / Final Report Completion

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Collect alerts and packets from snort

3. PROBLEMS ENCOUNTERED AND SOLUTIONS

[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- Collecting alerts and packets from ipv6 network
- Configuration in snort needed to modify

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

The students required to improve knowledge how to use the snort engine.

DR. NAVANEETHAN C. ARJUMAN
Senior Lecturer,
Faculty of Computing and Informatics,
Multimedia University, Persiaran Multimedia, 63100 Cyberjaya
Selangor Darul Ehsan, Malaysia

.....

Supervisor's Signature

Student's Signature

.....

.....

Co-Supervisor's Signature

Company Supervisor's Signature

(if applicable)

(if applicable)

CPT6324 Project (FYP2) Meeting Log

Trimester: March 2025 (Trimester ID:2510)

Meeting Date: **11/04/2025**	**Meeting No.:** **2**
Meeting Mode: **(Online)**	
Project ID: **FYP02-CS-T2510-0152**	**Project Type:** **Hybrid**
Project Title : Enhanced Detection Mechanism To Detect DOS Attacks in the IPv6 Network	
Student ID : **1221304026**	**Student Name:** **Flavian Navin Wenceslas**
Student Programme and Specialisation: **Computer Science Majoring Cyber Security**	
Supervisor Name: **Dr Navaneethan A/L C. Arjuman**	**Co-Supervisor Name:** **(if applicable)**
Collaborating Company: **(if applicable)**	**Company Supervisor Name:** **(if applicable)**

1. WORK DONE
[Please write the details of the work done, after the last meeting]

Tasks: Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report / Final Report Completion

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Run Snort to collect alerts and packets

2. WORK TO BE DONE
[Please write the details of the work to be done, before the next meeting]

Tasks: Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report / Final Report Completion

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Transfer alert and packets logs to my main windows to convert into a csv file(dataset)

3. PROBLEMS ENCOUNTERED AND SOLUTIONS
[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- Too little alerts and packets after running snort
- Solved to changing the custom rule

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

The students still struggling to configure Snort Engine and sending alerts.

DR. NAVANEETHAN C. ARJUMAN
Senior Lecturer,
Faculty of Computing and Informatics,
Multimedia University, Persiaran Multimedia, 63100 Cyberjaya
Selangor Darul Ehsan, Malaysia

.....
.....
Supervisor's Signature

Student's Signature

.....
.....
Co-Supervisor's Signature
Signature
(if applicable)

Company Supervisor's
(if applicable)

FACULTY OF COMPUTING & INFORMATICS

CPT6324 Project (FYP2) Meeting Log
Trimester: March 2025 (Trimester ID:2510)

Meeting Date: 21/04/2025	**Meeting No.:** 3
Meeting Mode: (Online)	
Project ID: FYP02-CS-T2510-0152	**Project Type:** Hybrid
Project Title : Enhanced Detection Mechanism To Detect DOS Attacks in the IPv6 Network	
Student ID : 1221304026	**Student Name:** Flavian Navin Wenceslas
Student Programme and Specialisation: Computer Science Majoring Cyber Security	
Supervisor Name: Dr Navaneethan A/L C. Arjuman	**Co-Supervisor Name:** (if applicable)
Collaborating Company: (if applicable)	**Company Supervisor Name:** (if applicable)

FACULTY OF COMPUTING & INFORMATICS

CPT6324 Project (FYP2) Meeting Log
Trimester: March 2025 (Trimester ID:2510)

Meeting Date: 21/04/2025	**Meeting No.:** 3
Meeting Mode: (Online)	
Project ID: FYP02-CS-T2510-0152	**Project Type:** Hybrid
Project Title: Enhanced Detection Mechanism To Detect DOS Attacks in the IPv6 Network	
Student ID: 1221304026	**Student Name:** Flavian Navin Wenceslas
Student Programme and Specialisation: Computer Science Majoring Cyber Security	
Supervisor Name: Dr Navaneethan A/L C. Arjuman	**Co-Supervisor Name:** (if applicable)
Collaborating Company: (if applicable)	**Company Supervisor Name:** (if applicable)

1. WORK DONE
[Please write the details of the work done, after the last meeting]

Tasks: Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report / Final Report Completion

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Adjust dataset accordingly to column names
- Combine dataset of packets and alerts
- Add column "Attack" to differentiate packets and alerts

2. WORK TO BE DONE
[Please write the details of the work to be done, before the next meeting]

Tasks: Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report / Final Report Completion

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Perform SVM and ANN models using dataset collected from snort

3. PROBLEMS ENCOUNTERED AND SOLUTIONS
[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- Could not create a csv file(dataset) manually
- Use python to create a csv file(dataset) accordingly to column names.

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

The students required additional knowledge how to use the AI Engines.

DR. NAVANEETHAN C. ARJUMAN
Senior Lecturer,
Faculty of Computing and Informatics,
Multimedia University, Persiaran Multimedia, 63100 Cyberjaya
Selangor Darul Ehsan, Malaysia

.....
.....
Supervisor's Signature

Student's Signature

.....
.....
Co-Supervisor's Signature
Signature
(if applicable)

Company Supervisor's
(if applicable)

FACULTY OF COMPUTING & INFORMATICS

CPT6324 Project (FYP2) Meeting Log
Trimester: March 2025 (Trimester ID:2510)

Meeting Date: 16/05/2025	**Meeting No.:** 4
Meeting Mode: (Online)	
Project ID: FYP02-CS-T2510-0152	**Project Type:** Hybrid
Project Title : Enhanced Detection Mechanism To Detect DOS Attacks in the IPv6 Network	
Student ID : 1221304026	**Student Name:** Flavian Navin Wenceslas
Student Programme and Specialisation: Computer Science Majoring Cyber Security	
Supervisor Name: Dr Navaneethan A/L C. Arjuman	**Co-Supervisor Name:** (if applicable)
Collaborating Company: (if applicable)	**Company Supervisor Name:** (if applicable)

1. WORK DONE
[Please write the details of the work done, after the last meeting]

Tasks: Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report / Final Report Completion

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Perform SVM using the dataset

2. WORK TO BE DONE
[Please write the details of the work to be done, before the next meeting]

Tasks: Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report / Final Report Completion

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Perform ANN using the dataset

3. PROBLEMS ENCOUNTERED AND SOLUTIONS
[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- Adding features which has string
- Convert the features to numerical using one-hot encode

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

The students still facing issue to train the AI Engine.

DR. NAVANEETHAN C. ARJUMAN
Senior Lecturer,
Faculty of Computing and Informatics,
Multimedia University, Persiaran Multimedia, 63100 Cyberjaya
Selangor Darul Ehsan, Malaysia

.....

Supervisor's Signature

Student's Signature

.....

.....

Co-Supervisor's Signature
Signature
(if applicable)

Company Supervisor's
(if applicable)

FACULTY OF COMPUTING & INFORMATICS

CPT6324 Project (FYP2) Meeting Log

Trimester: March 2025 (Trimester ID:2510)

Meeting Date: 30/05/2025	**Meeting No.:** 5
Meeting Mode: (Online)	
Project ID: FYP02-CS-T2510-0152	**Project Type:** Hybrid
Project Title : Enhanced Detection Mechanism To Detect DOS Attacks in the IPv6 Network	
Student ID : 1221304026	**Student Name:** Flavian Navin Wenceslas
Student Programme and Specialisation: Computer Science Majoring Cyber Security	
Supervisor Name: Dr Navaneethan A/L C. Arjuman	**Co-Supervisor Name:** (if applicable)
Collaborating Company: (if applicable)	**Company Supervisor Name:** (if applicable)

FACULTY OF COMPUTING & INFORMATICS

CPT6324 Project (FYP2) Meeting Log
Trimester: March 2025 (Trimester ID:2510)

Meeting Date: 30/05/2025	**Meeting No.:** 5
Meeting Mode: (Online)	
Project ID: FYP02-CS-T2510-0152	**Project Type:** Hybrid
Project Title : Enhanced Detection Mechanism To Detect DOS Attacks in the IPv6 Network	
Student ID : 1221304026	**Student Name:** Flavian Navin Wenceslas
Student Programme and Specialisation: Computer Science Majoring Cyber Security	
Supervisor Name: Dr Navaneethan A/L C. Arjuman	**Co-Supervisor Name:** (if applicable)
Collaborating Company: (if applicable)	**Company Supervisor Name:** (if applicable)

1. WORK DONE
[Please write the details of the work done, after the last meeting]

Tasks: Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report / Final Report Completion

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Perform ANN using the dataset and create a graphical user interface using ANN model

2. WORK TO BE DONE
[Please write the details of the work to be done, before the next meeting]

Tasks: Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report / Final Report Completion

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Comparison of Snort, SVM and ANN results(accuracy)

3. PROBLEMS ENCOUNTERED AND SOLUTIONS
[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- Took some time to load the results
- Reducing epochs

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

Teh student required further knowledge how implement the AI Engines.

DR. NAVANEETHAN C. ARJUMAN
Senior Lecturer,
Faculty of Computing and Informatics,
Multimedia University, Persiaran Multimedia, 63100 Cyberjaya
Selangor Darul Ehsan, Malaysia

.....
.....
Supervisor's Signature

Student's Signature

.....
.....
Co-Supervisor's Signature
Signature
(if applicable)

Company Supervisor's
(if applicable)

CPT6324 Project (FYP2) Meeting Log
Trimester: March 2025 (Trimester ID:2510)

Meeting Date: 13/06/2025	**Meeting No.:** 6
Meeting Mode: (Online)	
Project ID: FYP02-CS-T2510-0152	**Project Type:** Hybrid
Project Title : Enhanced Detection Mechanism To Detect DOS Attacks in the IPv6 Network	
Student ID : 1221304026	**Student Name:** Flavian Navin Wenceslas
Student Programme and Specialisation: Computer Science Majoring Cyber Security	
Supervisor Name: Dr Navaneethan A/L C. Arjuman	**Co-Supervisor Name:** (if applicable)
Collaborating Company: (if applicable)	**Company Supervisor Name:** (if applicable)

1. WORK DONE
[Please write the details of the work done, after the last meeting]

Tasks: Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report / Final Report Completion

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Comparison of Snort, SVM and ANN results using bar graph

2. WORK TO BE DONE
[Please write the details of the work to be done, before the next meeting]

Tasks: Implementation / Testing (Application-based projects) or Evaluation of Findings and Research Contribution (Research-based projects) / Commercialisation Proposal (Application-based projects) or Research Paper (Research-based Projects) / Draft Final Report / Final Report Completion

(Please strike out the tasks, which are not applicable)

Details (in point form):

- Documentation and poster

3. PROBLEMS ENCOUNTERED AND SOLUTIONS
[Please write the details of the problems encountered, after the last meeting and provide the solutions / plan for the solutions]

- Graphical user interface uses SVM concept
- Changed the concept which uses ANN

4. COMMENTS (Supervisor / Co-Supervisor / Company Supervisor)

The student manages to train the AI Engines and collected results. Student needs to work on draft report and share the report later.

DR. NAVANEETHAN C. ARJUMAN
Senior Lecturer,
Faculty of Computing and Informatics,
Multimedia University, Persiaran Multimedia, 63100 Cyberjaya
Selangor Darul Ehsan, Malaysia

Supervisor's Signature

Student's Signature

Co-Supervisor's Signature
Signature
(if applicable)

Company Supervisor's
(if applicable)

Appendix B (Turnitin Similarity Index)

FYP2-Report from turnitin.docx

ORIGINALITY REPORT

13%	2%	3%	12%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Multimedia University Student Paper	11%
2	www.mdpi.com Internet Source	<1%
3	www.hindawi.com Internet Source	<1%
4	www.science.gov Internet Source	<1%
5	Arvind Dagur, Karan Singh, Pawan Singh Mehra, Dhirendra Kumar Shukla. "Intelligent Computing and Communication Techniques - Volume 3", CRC Press, 2025 Publication	<1%
6	Submitted to Technological University Dublin Student Paper	<1%
7	www.frontiersin.org Internet Source	<1%
8	www.researchgate.net Internet Source	<1%

9	Sachi Nandan Mohanty, Preethi Nanjundan, Tejaswini Kar. "Artificial Intelligence in Forecasting - Tools and Techniques", CRC Press, 2024 Publication	<1 %
10	Submitted to University of Westminster Student Paper	<1 %
11	data.epo.org Internet Source	<1 %
12	scholarworks.sjsu.edu Internet Source	<1 %
13	Firas Saidi, Zouheir Trabelsi, Henda Ben Ghazela. "Fuzzy Logic Based Intrusion Detection System as a Service for Malicious Port Scanning Traffic Detection", 2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA), 2019 Publication	<1 %
14	Sumeet Dua, Xian Du. "Data Mining and Machine Learning in Cybersecurity", Auerbach Publications, 2019 Publication	<1 %
15	www.ijsrtjournal.com Internet Source	<1 %

16	Chwan-Hwa (John) Wu, J. David Irwin. "Introduction to Computer Networks and Cybersecurity", CRC Press, 2019 Publication	<1%
17	Yasser Alharbi, Ali Alferaidi, Kusum Yadav, Gaurav Dhiman, Sandeep Kautish. "Denial-of-Service Attack Detection over IPv6 Network Based on KNN Algorithm", Wireless Communications and Mobile Computing, 2021 Publication	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		

Appendix C – Github link

https://github.com/Flavy5/Final-Year-Project-2