NETWORK INTRUSION PREVENTION SYSTEM USING NETFLOW DATA AND MACHINE LEARNING

Final Year Dissertation

Author:
Jacinth Daniel Moses

Supervisor:
Professor. Kayvan Karim

A thesis submitted in fulfillment of the requirements for the degree of BSc (Hons) Computer Science (Cyber Security)

in the

School of Mathematical and Computer Sciences

Heriot-Watt university
Dubai

April 2024

Student Declaration of Authorship

Course code and name:	F20PA: Research Methods & Requirements Engineering
Type of assessment:	Individual
Coursework Title:	Final Year Dissertation
Student Name:	Jacinth Daniel Moses
Student ID Number:	H00379506

Declaration of authorship. By signing this form:

- **I declare** that the work I have submitted for individual assessment OR the work I have contributed to a group assessment, is entirely my own. I have NOT taken the ideas, writings or inventions of another person and used these as if they were my own. My submission or my contribution to a group submission is expressed in my own words. Any uses made within this work of the ideas, writings or inventions of others, or of any existing sources of information (books, journals, websites, etc.) are properly acknowledged and listed in the references and/or acknowledgements section.

- I confirm that I have read, understood and followed the University's Regulations on plagiarism as published on the University's website, and that I am aware of the penalties that I will face should I not adhere to the University Regulations.

- I confirm that I have read, understood and avoided the different types of plagiarism explained in the University guidance on Academic Integrity and Plagiarism

Student Signature: *Jacinth Daniel Moses*

Date: *03/04/2024*

Copy this page and insert it into your coursework file in front of your title page.
For group assessment each group member must sign a separate form and all forms must be included with the group submission.

Your work will not be marked if a signed copy of this form is not included with your submission.

Acknowledgements

I would like to start by thanking my supervisor Professor Kayvan Karim for his constant communication, guidance and gentle nudges to show me the right direction that this dissertation should take.

And the team at nTop for providing a free copy od nProbe Enterprise as part of their free software for academia initiative, Especially Maria Teresa Allegro for quick correspondence and support.

And Finally the open Source community, namely Anthony Lapenna for providing the containerized ELK Stack and thomaslaurenson for providing a containerized website for attack simulation.

Contents

Acknowledgements **ii**

Contents **iii**

List of Figures **v**

List of Tables **vi**

Abbreviations **vii**

Abstract **viii**

1 Introduction **1**
1.1 Motivation . . . 2
1.2 Aim . . . 3
1.3 Primary Objectives . . . 3
1.4 Document Structure . . . 3

2 Background **4**
2.1 Intrusion Detection Techniques . . . 4
2.1.1 Signature-Based detection . . . 4
2.1.2 Anomaly-Based Detection . . . 4
2.2 Network Intrusion Prevention Systems . . . 4
2.3 NetFlow . . . 5
2.4 Related Works . . . 6
2.5 Critiques . . . 10
2.6 Related Works Results . . . 12
2.7 Proposed Feature Sets . . . 13

3 Experimentation & Implementation Methodology **15**
3.1 Machine Learning & Deep Learning Experimentation . . . 15
3.1.1 Data Preprocessing . . . 15
3.1.2 Dataset Feature Selection . . . 17
3.1.3 Models Experimentation . . . 18
3.1.3.1 Cross-Validation . . . 19
3.1.3.2 Evaluation Metrics . . . 20
3.2 NIPS System Implementation . . . 20
3.3 Network Setup & Attack Simulation . . . 24
3.3.1 GNS3 . . . 24
3.3.2 VMWare . . . 25
3.4 NIPS Testing Methodology . . . 25
3.4.1 SQL Injection . . . 26
3.4.2 Reconnaissance/Scanning . . . 26
3.4.3 Denial of Service attack (DoS) . . . 26

3.4.4 Distributed Denial of Service attack (DDoS) 27
3.4.5 nProbe Setup 27
3.5 Project Management 27

4 Feature Subset & NIPS Evaluation 28
4.1 Machine Learning & Deep Learning Evaluation 28
4.1.1 Evaluation Metrics 28
4.1.2 Binary Classification Results 30
4.1.3 Multi-Class Classification Results 32
4.1.4 Feature Subset Evaluation 34
4.2 NIPS Evaluation 34
4.2.1 SQL Injection 35
4.2.2 Denial of Service attack (DoS) 35
4.2.3 Distributed Denial of Service attack (DDoS) 36
4.2.4 Reconnaissance/Scanning 36
4.2.5 Benign Flows 38
4.2.6 Network Speeds 38

5 Conclusion and Future Work 39
5.1 Conclusions 39
5.2 Future Work 39

Apendix Structure 40

A PLES 44
A.1 Professional and Legal Considerations 44
A.2 Ethical and Social Considerations 44

B Project Management 45
B.1 Project Plan 45
B.2 Risk Analysis 47

C Supplementary 48
C.1 Chapter 1 48
C.2 Chapter 3 50
C.2.1 Use Case Scenario 51
C.2.2 User Requirements 52
C.3 Chapter 4 53
C.3.1 Confusion Matrices for Binary Classifications 55
C.3.2 Confusion Matrices for Multi Classifications 59

List of Figures

2.1 A network diagram with an NIDS installed 5
2.2 A network diagram with an NIPS installed 5
2.3 NetFlow components 5

3.1 The process used for testing 15
3.2 Before Under-Sampling 16
3.3 After Under-Sampling 16
3.4 Cross-Validation Method Used 19
3.5 Flow of Execution 22
3.6 Kibana Sankey Diagram Visualization 22
3.7 Kibana Elastiflow's Exported Logs 23
3.8 closer look at flow record in Kibana 23
3.9 Kibana NIPS Logs 24
3.10 Routes configured on windows 10 24
3.11 GNS3 Testing Setup used for developing NIPS 25

4.1 Confusion Matrices the of top Binary Classification subset from each Model 31
4.2 Confusion Matrices of the top Multi-Classification subset from each Model 33
4.3 Snort initialization 34
4.4 Kali server hosting thomaslaurenson's vulnerable container and kali attacker injecting a payload 35
4.5 Snorts SQL Classification 35
4.6 NIPS SQL Classification 35
4.7 Snorts DoS Classification 35
4.8 Snorts DDoS Classification 36
4.9 Snorts Scanning Classification 36
4.10 NIPS Scanning Classification During a TCP Scan 37
4.11 NIPS Scanning Classification During a UDP Scan 37
4.12 NIPS Scanning Classification During a NULL Scan 37
4.13 NIPS Scanning Classification During a XMAS Scan 38
4.14 NIDS Network Speeds 38
4.15 Snorts Network Speeds 38

B.1 Deliverable 1 Gantt chart 45
B.2 Deliverable 2 Gantt chart 46

C.1 Before Under-Sampling 50
C.2 After Under-Sampling 50
C.3 Proposed use case for NIPS 51
C.4 Wireshark screenshot of network during DoS attack 61
C.5 Wireshark screenshot of network during DDoS attack 62
C.6 The docker compose stack running 63

List of Tables

2.1 Summary of results from related works 12

3.1 Count of Each Label 16
3.2 Parameter values for RF and ANN 18
3.3 ANN Layer Structure 18

4.1 Classification Metrics and Formulas 28
4.2 Naming Conventions 29
4.3 Binary Classification Cross-Validation Resulst 30
4.4 Summarized Binary Prediction Results 30
4.5 Summarized Multi-Classification Prediction Results 32
4.6 Multi-Class Cross-Validation Results 33

C.1 NetFlow v2 Features List from [1] 48
C.2 Elastiflow to Dataset name mappings 49
C.3 GNS3 VM Nodes configurations 50
C.6 MoSCoW Abbreviations 53
C.7 Feature Importance based on 5 subsets from previous works. 54
C.8 All Binary Prediction Results 56
C.9 Multi-Classification Prediction Results 57

Abbreviations

IOC	**I**ndicators **O**f **C**ompromise	**SOC**	**S**ecurity **O**perations **C**enter
NIDS	**N**etwork **I**ntrusion **D**etection **S**ystem	**TTP**	**T**actics **T**echniques **P**rocedures
NIPS	**N**etwork **I**ntrusion **P**revention **S**ystem	**IoT**	**I**nternet **O**f **T**hings
SIEM	**S**ecurity **I**ncident and **E**vent **M**anagement	**IR**	**I**ncident **R**esponse
FS	**F**eature **S**election	**FE**	**F**eature **E**xtraction
DT	**D**ecision **T**ree	**RF**	**R**andom **F**orest
ANN	**A**rtificial **N**eural **N**etwork	**CNN**	**C**onvolutional **N**eural **N**etwork
AUC	**A**rea **U**nder **C**urve	**DR**	**D**etection **R**ate
FAR	**F**alse **A**larm **R**ate	**ROC**	**R**onvolutional **O**eural **C**etwork
IP	**I**nternet **P**rotocol	**OS**	**O**perating **S**ystem
NAT	**N**etwork **A**ddress **T**ranslation	**CLI**	**C**ommand **L**ine **I**nterface
GNS3	**G**rapical **N**etwork **S**imulater **3**		

Abstract

As the world becomes increasingly connected every day, the amount of traffic that flows through networks all around the world increases at an exponential rate [2], similarly the amount of bad actors who look to exploit flaws and gain access to computer systems or networks is also increasing. One solution to guarding a personal network is through the use of a network intrusion prevention system (NIPS), which uses various detection methods to detect traffic with malicious intent, by analyzing every network packet that flows through the system and blocks them if it is classified as an attack, this however is not that efficient, as every single packet that flows through the system needs to be analyzed, this can quickly become a bottleneck for even a medium-sized network. The solution this dissertation proposes is a new NIPS that uses an aggregated conversation summary called NetFlow put through a machine learning (ML) algorithm to detect malicious activity while the packets are free to pass through at full speed, and if malicious activity is detected, the origin IP's subsequent packets are dropped.

Keywords: *Network Intrusion Detection Systems, Network Intrusion Prevention Systems, NIDS, NIPS, NetFlow, Machine Learning*

Chapter 1

Introduction

Technology is advancing at an exponential rate, and so is cybercrime [3], with releases of new software, hardware, and operating systems (OS) come more potential for vulnerabilities to be discovered and exploited as they are designed and implemented by humans, and humans tend to make mistakes or overlook an aspect of their code or system that can be used as a vulnerability. An example of this could be a web developer not checking the inputs a user can give in a website text field allowing them to run SQL code via SQL Injection [4], A few years ago in 2019, Fortnite had an SQL injection vulnerability where a malicious actor could access any user's account [5]. And more recently in 12th of April 2024, a command injection vulnerability was discovered in the GlobalProtect feature of PAN OS software. When the attack is executer the attacker can run any malicious code on the firewall with root/admin Privileges [6].

One solution to this problem is to deploy a system a Network Intrusion Detection Systems (NIDS) which can analyze traffic using two main detection methods, statistical-anomaly-Based and signature-based methods to flag suspicious behavior and generate an alert to a security incident and event management (SIEM) tool.

When an alert pops up in a SIEM tool, a security operations center (SOC) analyst will then respond to the matter by referring to a playbook which is a step-by-step guide designed to help them navigate and manage the alert, this is called Incident Response (IR). And while security orchestration Automation and Response (SOAR) tools and even some SIEM tools can have some built-in or pre-defined actions\workflows set up by a person that executes when a specific type of alert occurs, they are often limited. For example, if a denial of service alert is received, the workflow will automatically drop packets coming from that IP and ban it. However, SOARs are notoriously difficult to integrate into existing infrastructure and very difficult to set up as they are very complex [7].

A Network Intrusion Prevention System (NIPS) is very similar to an IDS and is often referred to as the "same thing" and merged into terms such as NIDPS as It also uses similar detection methods as listed above and generates alerts to an SIEM or SOAR tool. with the one key difference being, an IPS will analyze the traffic in real-time while an IDS does it later, and the other being that an IDS just reports what happens in the network and expects a SOC analyst to take action with the information it provides, while the IPS takes action and actively does the incident response without human intervention

necessary. One might think that SOAR and NIPS are essentially the same, but thats not true as IPS works in real-time.

However, as more devices connect to our networks and use the internet, the number of packets that run through a network, and thus an NIPS, increase greatly, as each packet needs to be examined by the NIPS. This requires a lot of computational power and processing time. Additionally, the NIPS may not be able to efficiently find patterns that indicate an attack as individual packets don't have many Indicators of compromise(IOC's), and if the NIPS keeps track of all conversations in the network itself, it is only as robust as the RAM allocated to it, as a malicious actor can generate fake traffic that the NIPS must analyze and keep track of, which will either push malicious traffic out of the scope of analysis of the NIPS or crash the NIPS, bringing the network down. But by looking at an aggregated view of the traffic from an endpoint, which is the main purpose of NetFlow, we could potentially detect any malicious intent more accurately and classify it faster while not impacting Network traffic speeds, as the packets themselves are not being analyzed. and so analyzing an aggregated format like NetFlow is beneficial as many cisco routers support the protocol and cisco routers are very popular [8], The distributing the load of traffic and is more easily scalable as buying new hardware is easier then allocating more RAM to a system as there are physical and software limitations to doing so.

This dissertation focuses on proposing/building a Network-Based Intrusion Prevention System which uses a Machine Learning algorithm based Decision-Tree classifier along with a subset of netFlow features for classification which were decided on using the evaluation metrics that are shown in chapter 5. Typically Machine learning is reserved for use in NIDS as they can be slow [9] and thats not an issue for an NIDS as they don't work in real-time, however, an IPS does and so the ML Model can become a bottleneck and so rather than analyzing every packet that flows through the network, This IPS uses NetFlow flows to analyze malicious flows and send any Internet Protocol Addresses (IP)s it deems to be malicious to the firewall (IP Tables) which will drop packets accordingly. Thus allowing packets to flow through the network at their normal speeds.

1.1 Motivation

Current NIPS solutions such as Trend Micro TippingPoint and Cisco Secure Firewall while reliable, are outdated, slow and clunky [10]. I am setting out to propose/make this an easy-to-use system utilizing machine learning to analyze NetFlow flows in realtime without causing a significant hit to network performance.

1.2 Aim

I aim to find an optimal subset of NetFlow features and pair it with an Machine Learning(ML) or Deep Learning(DL) model to close the gap between prediction accuracy and classification speed. Additionally, I intend to present an easy-to-use system, that can be deployed by running a docker container and some simple configuration. The primary users will be small to medium sized businesses and people who host web services from home.

1.3 Primary Objectives

1. Find a suitable set of features to minimize complexity and maintain accuracy of detection.
2. Find a suitable machine learning or deep learning model based on prior works to find a faster model for real-time detection for both binary and multi-class classification.
3. Build a self contained docker application that can be deployed on any linux computer to act as the NIPS in a low complexity network.
4. Present an industry standard dashboard to the network admin/SOC analysts with summaries of both the network and NIPS.
5. The system should not in any way affect the throughput or speed of the network

1.4 Document Structure

- Chapter 1: Is an Introduction to the project containing my aims, objectives, problem space and motivations behind this dissertation
- Chapter 2: Contains prior research relevant to the application of machine learning and deep learning to other NIDS and NIPS systems, Feature selection methods for faster processing, etc. . . and a critique of each research paper.
- Chapter 3: Contains details about the implementation used during feature subset testing and implementation of the proposed system.
- Chapter 4: Discusses the results of feature subset testing and model evaluation and also goes into details about evaluation and the effectiveness of the proposed system.
- Chapter 5: Discusses final conclusions and proposes future areas of work.

Chapter 2

Background

2.1 Intrusion Detection Techniques

2.1.1 Signature-Based detection

One detection method that NIDS uses is called Signature-Based detection [11] which looks at packets and compares their signature *(which is a unique pattern or identifier for that specific packet)* [12] to a database of known malicious packets and flags the packet or stream as malicious and generates an alert to a dashboard. But this detection method is vulnerable to variant attacks. This database is an accumulation of known malicious packets that are typically created by researchers or network defenders.

2.1.2 Anomaly-Based Detection

Another detection method and NIDS can use is Statistical-Anomaly-Based Detection [11] which when installed sets up a baseline that it dictated to be normal network traffic behavior. When the network exhibits behavior that deviates from the baseline like for instance a large transfer of data from a server to another an outside client or a specific IP is constantly requesting a web page from the server *(Denial of Service Attack)* This will be flagged as malicious activity and an alert in the chosen dashboard will pop up. This detection method however is susceptible to many false positives, As our approach for detection is similar we will be looking for a model to lower the amount of false positives we encounter

2.2 Network Intrusion Prevention Systems

An IPS can be split into 2 main types, Network-Based IPS which installed in such a way that it can monitor all traffic that enters and exits the network while a Host-Based IPS is installed on individual host devices or endpoints such as desktops, laptops, raspberry pi's, etc.. . . and analyzes the logs created by the operating system to look for threats.

The main difference being an NIDS (depicted in Figure 2.1) will simply report any malicious activity in the network most likely after the attack has been carried out, while

an NIPS will actively to stop the security threat as intrusion detection happens in real time as network packets flow through the NIPS as depicted in Figure 2.2. and this can be dropped.

FIGURE 2.1: A network diagram with an NIDS installed

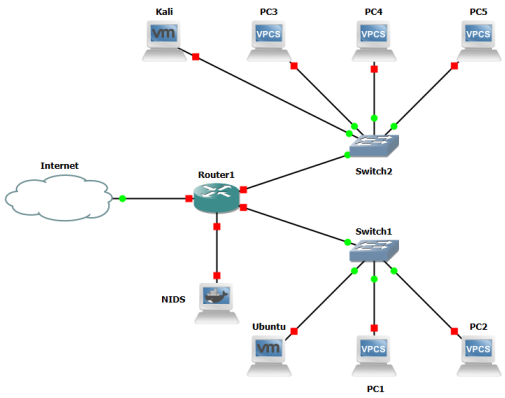
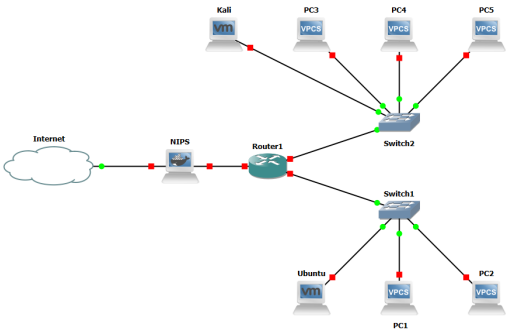
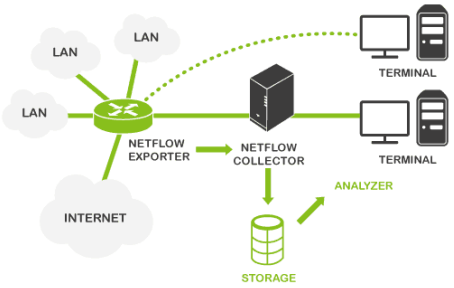


FIGURE 2.2: A network diagram with an NIPS installed



2.3 NetFlow

FIGURE 2.3: NetFlow components



Recognized as an Internet Engineering Task Force (IETF) standard, NetFlow, is a network protocol developed by Cisco Systems and is widely used to collect statistics about the IP traffic flowing across network devices such as routers, switches, and hosts. It manages to do this by monitoring entire conversations (flows) rather than individual packets, which is better for our use case as it is more efficient to analyze a flow as opposed to every packet that flows in the network. [13]

A typical flow monitoring setup consists of:

- Observation point: Usually a router, switch, or firewall interface through which traffic passes. A set of observation points in a domain is called an Observation domain.
- NetFlow Exporter: A service that runs on an observation point that exports flows in the form of flow records by aggregating data packets and periodically exports NetFlow records via User Datagram Protocol (UDP) to NetFlow collectors. A flow is recognized by analyzing packets for similar attributes to other packets, such as the source IP address, destination address, source port, destination port number, Layer-3 protocol field, or type of service. Finally, a flow is sent to the collector

when no new conversations for a particular flow occur within a period of time or when a TCP flag indicates that the flow is over (FIN or RST). [14]

- NetFlow Collector: A central service or server that receives flow records from multiple exporters and parses and stores the data in a database or somewhere it can be analyzed later. It can be either a software or hardware-based tool. [15]
- NetFlow Analyzer: A tool that analyzes the data stored by the collector and reports no network activity by creating alerts that provide insight into bandwith usage, traffic patterns, etc. This process helps us identify malicious activity occurring in the network, which is part of what this dissertation aims to produce by implementing ML for intrusion detection. [16]

One similar protocol, known as IP Flow Information Export (IPFIX), which is informally known as NetFlow v10, is based on NetFlow v9 and follows NetFlow v9's wide adoption rate [17]; however, the two protocols are not interoperable.

2.4 Related Works

As network intrusion detection using ML is a popular topic for many researchers as many papers have come out with various models and feature selectin techniques claiming to be better than others. However, it is difficult to compare the performance of ML models against each other as different datasets that were used for training and testing have different features, as the authors of the datasets had different ideas of what features would be helpful for classification. The paper by [18] aims to deliver a solution to this by providing five datasets that have been converted from raw packet capture files to the NetFlow format.

The benefit is the standardization of feature sets and their relevance, as NetFlow-enabled routers are prevalent in not only the workplace, but homes as well. The datasets used by M. Sarhan et al. are UNSW-NB15, BoT-IoT, ToN-IoT, and CSE-CIC-IDS2018. Flows from the PCAP files of the 4 original datasets were extracted by the nProbe tool, which converts the flows into relevant NetFlow V9 identifiers before saving them in CSV files. The binary labels are then created by identifying the flow using the five flow identifiers, which are source/destination IPs, port numbers, and protocols. When a flow is identified as an attack it is labelled as 1 and 0 for benign flow. The result of doing these operations on the four datasets is the creation of NF-UNSW-NB15, NF-BoT-IoT, NF-ToN-IoT, and NF-CSE-CIC-IDS2018. Lastly, the 5th dataset NF-UQ-NIDS is created by merging all 4 datasets, creating a "universal NIDS dataset". For evaluation, the new datasets were tested against the old datasets by putting each through an Extra Trees ensemble classifier, and the results showed that for both binary and multi-classification,

a slight hit to accuracy (1-2%) was observed but prediction times dropped consistently, which is always a plus in a high-speed network.

Feature selection (FS) is an important part of making the model both efficient and accurate as discarding the features with the little to no correlation to a classification (noise reduction) can prevent overfitting, which is when the model looks for features in the unseen data that have no correlation to the classification and thus leads to worse real-world classifications (worse generalization). FS methods which can be categorised as wrapper methods which generates subsets of the data and uses an evaluation function to evaluate the subset, filter methods which work by selecting features-based on their corelation to the classification outcome, embedded methods which is when FS is integrated into the classifier algorithm and hybrid methods which is a combination of 2 or 3 methods. Another benefit is increased computational efficiency as models with fewer features are faster to train and less resource intensive which leads to faster classifications both for the training dataset and unseen data which is important for our use case as we want detection to happen in real time.

One way to select features is proposed by this paper by Fraihat et al. [19] which deals with selecting a subset of features that will both reduce dimensionality and increase detection accuracy. The author accomplishes this by using a modified Arithmetic Optimization Algorithm (AOA) and selects 7 attributes from the NF-ToN-IoT-v2 dataset. The way the modified AOA works is by using the 4 basic arithmetic operations (+,-,*,/) and utilizing a wrapper method and evaluating them via an evaluation function which in this case is the error rate of a decision tree classifier (Lower error rate is better). The author then proceeds to test the average, standard deviation, and worst/best values for other optimization algorithms such as Binary Bat algorithm (B)(BAT), White Shark Optimizer B(WSO), and Grey Wolf Optimizer B(GWO) along with the proposed AOA B(AOA).

The datasets produced by each FS algorithm was tested using the following 4 classifiers: Decision tree (DT), random forest (RF), XGBoost and Naïve Bayes (NB). BWGO attained the best average, STD with 8 features; however, the paper mentions that BAOA was the runner-up, and in testing, the results between BAOA and BWGO would fluctuate, as one would do better in one round and worse in the other. Following this the error rate and accuracy of the 4 algorithms were tested, and they found that with 7 attributes, BAOA achieved the best error rate while BGWO attained better accuracy with a small margin of 0.002 with 8 attributes given to the algorithms. Finally the proposed features for binary and multiclassification are presented in table 2.7

In the next paper [20] we see the paper claim to create an optimal subset of NetFlow features using the datasets created by [19]. It achieves this by looking into different FS methods which can be categorised as wrapper methods which generates subsets of the data and uses an evaluation function to evaluate the subset, filter methods which

work by selecting features-based on their corelation to the classification outcome, embedded methods which is when FS is integrated into the classifier algorithm and hybrid methods which is a combination of 2 or 3 methods. The first technique they used is LASSO regularization which is an embedded method that works by adding penalties to coefficients of the features that are less important and finds a set of features with the smallest absolute value of coefficients. This approach is great as the regularization prevent overfitting; however, if a features penalty is too high it could lead to feature being dropped wrongly.

The next technique they use is chi2 which is a filter method works by first formulating a null hypothesis that the feature and classification are independent variables, constructing a contingency table which displays the frequency distribution of the features, calculate the expected frequencies under the null hypothesis assumption and then calculate the chi2 statistic which is compared with a critical value, if the calculated value is greater the null hypothesis is rejected suggesting there is a correlation. And lastly random forest importance which is an embedded method works by evaluating the importance of a feature by counting how many times the feature comes up in the ensemble of decision trees (DT) that are produced by the random forest (RF) classifier, the more a feature contributes to a classification the more important it is. After all these tests were run an optimal subset of 10 features was produced as seen in table 2.7 and to ensure that this feature set could actually yield good results it was tested on 4 classifiers RF, AdaBoost, Naïve (NB) and ANN which yielded very good results with F1 scores regularly greater than 94%.

Another method to select features is seen in [20] we see the paper claim to create an optimal subset of NetFlow features using the datasets created by [19]. It achieves this by looking into 3 techniques. The first technique they used is LASSO regularization which is an embedded method that works by adding penalties to coefficients of the features that are less important and finds a set of features with the smallest absolute value of coefficients. This approach is great as the regularization prevent overfitting; however, if a features penalty is too high it could lead to feature being dropped wrongly. The next technique they use is chi2 which is a filter method works by first formulating a null hypothesis that the feature and classification are independent variables, constructing a contingency table which displays the frequency distribution of the features, calculate the expected frequencies under the null hypothesis assumption and then calculate the chi2 statistic which is compared with a critical value, if the calculated value is greater the null hypothesis is rejected suggesting there is a correlation.

And lastly random forest importance which is an embedded method works by evaluating the importance of a feature by counting how many times the feature comes up in the ensemble of DT that are produced by the RF classifier, the more a feature contributes to a classification the more important it is. After all these tests were run an optimal subset of 10 features was produced as seen in table 2.7 and to ensure that this feature set could

actually yield good results it was tested on 4 classifiers RF, AdaBoost, NB and Artificial neural network (ANN) which yielded very good results with F1 scores regularly greater than 94%.

The paper by Awad, M.; Fraihat, S.; Salameh, K.; Al Redhaei, A. [21] looks to find a balance between accuracy and performance by using feature correlation and feature importance. Taking the NF-ToN-IoT-v2 dataset created by [18] which has 43 features, they first perform data cleaning (removing incomplete rows), data transformation which is turning the class labels into numbers 0-9 for multi-classification and 0-1 for binary classification. The Pearson correlation coefficient is then calculated between feature pairs and pairs that have a high correlation (close to 1 or -1) are removed. Then the random forest classifier is used to give all remaining features in the dataset importance scores based on its contribution to a classification and subsets based on different threshold values (Mean, median, etc.) are created and tested using DT, RF, XGB, and NB. After testing they found the best threshold value to be the mean = 0.023 and this produced 13 features for binary classification and 17 for multi classification, The results can be seen in the table below 2.1

Deep learning (DL) has some advantages over Machine Learning (ML) such as handling lage and complex datasets, improved performance on things like computer vision, can handle missing values and is quite scalable and generalizable to other datasets. Its application is explored in the paper by [22] aims to compare the performance metrics of ML and DL models by using different subsets of the latest CIDDS dataset and balanced the dataset by under sampling meaning an equal distribution for all classes. For ML classifiers the author uses K-Nearest Neighbour (KNN) which is a parametric algorithm that classifies new datapoints based on its distance to K neighbours, XGBoost (XGB) which is an optimized gradient decent algorithm that builds many decision trees to improve prediction accuracy and speed and finally Classification and regression tree (CART) which constructs a binary tree by recursively splitting the data based on the beast features and thresholds.

For DL the author uses a Convolutional Neural Network (CNN) which is an algorithm that uses multiple layers of filters to extract features from the input features and performs classification and Long Short-Term Memory (LSTM) which is a type of recurrent Neural Network (RNN) with memory cells to process long term dependencies. The author aims to combine predictions from the best performing ML and DL models to create an ensemble model which is a combination of different models. The results showed that the CART model performed well as it had an accuracy of 99.8% and the lowest training time, while DL classifiers such as CNN+LSTM were found to be accurate (after embedding) but took a long time to train and did not have any major performance gains.

Another paper that uses a similar approach is [23] where their proposed model for multiclassification is a CNN+LSTM+SVM, an SVM performs final classification with the input from the LSTM model in the place of the SoftMax function. For feature

reduction the author normalized the data followed by Principal Component Analysis (PCA); which was used to reduce the remaining 77 features to 30. Other models tested were KNN (K=5), RF,CNN,CNN-LSTM, and DNN(5 Layers) and the results showed that the proposed model had the best accuracy (97.29%) - why no times shown, and why no look into binary classification

Finally we look at a two-level classifier which is a combination of 2 different classification algorithms where the first level is usually a quick classification of the data (usually binary) and level 2 is a longer but more detailed multi-classification algorithm. The article written by Seo, W, and Pak, W [9] looks to implement a real-time network intrusion prevention system (NIPS) using a hybrid machine learning approach. The paper states that conventional machine-learning methods are too slow to be used in real-time scenarios as the model's processing time becomes a bottleneck in a network where throughput is greater than 100Gbps. The paper proposes a two-level classifier where level 1 is a packet-based classifier, which classifies packets as they come in as either an attack or benign, but with lower accuracy. If level 1 cannot accurately classify a packet, it gets sent to level 2, which is a flow-based classifier, which is a slower but more accurate classification algorithm.

This approach has a minimal effect on network throughput while also maintaining a higher accuracy rate, and although By the time level 2, is done with classifying the packet, the flow will most likely be complete; it is more context-aware as opposed to analysing all bits of the packet individually. The results presented in the paper support the claims made as classification times for the UNSW-NB15 and CICIDS2017 datasets using the proposed decision tree algorithm had the shortest classification time, and the proposed random forest algorithm has the highest precession, recall, and F1-Score. Other algorithms, such as HAST-I, normal DT, and normal RF, were tested.

2.5 Critiques

Some general critiques for many of these papers are that it does not talk much about generalizability as the training, testing and validation are usually done using 1 dataset, some exceptions are papers by [19] and [22].

Another major critique for some papers such as [20] [23] [19] and [9] is that they don't report the classification time or training time taken or don't consider them to be important in their evaluation metrics. Conventionally, training time is not that important of a metric, however, This project aims to allow users to compile different models in place of the chosen models, it would have been helpful to gain that insight.

The paper by M.Sarhan Et al. [18] only tests a decision tree ensemble classifier but fails to test any other algorithms such as CNN,ANN, XGBoost etc. This dissertation experiments with multiple models taking inspiration from the paper by M.Komisarek Et

al. [20]. However this paper is also not without critiques, as it only looks at features that would be able to do both binary and multi-classification at the same time and doesn't take into consideration that certain features may be better for binary classification and likewise for multi-classification.

Another critique found regarding the paper by S.Fraihat Et al. [19] is that BGWO should have been explored further as the tables in chapter 4 suggested that with 8 attributes the error rate only lagged behind BAOA which used 7 attributes by 0.01 but yielded slightly greater accuracy of 0.002. Another critique is that the time for training hasn't been listed nor has the prediction times been listed or taken into account for all the different classifiers tested.

Regarding the paper by N.Thapa Et al. [22], While the authors considered training time they did not consider classification times for the models tested. In this project both classification time and training time were be taken into consideration as this system is intended to be deployed in real-time.

In the paper by I.Shivhare Et al. [23] This paper does not talk about how well the tested deep learning models generalize to new data nor does it mention any other metrics besides accuracy. In this project multiple data sets and model testing such as Cross Validation(CV) were done to ensure generalizability.Similarly this paper by W.Seo Et al. [9] does not calculate numerical summaries about how the proposed DF and RF work together in classification nor does it provide any numerical data about the false positive and false negative rates. To mitigate these oversights, important numerical metrics such as accuracy, F1-Scores, Confusion Matricies, . . . was recorded.

And finally the proposed feature sets by M.Awad Et al. [21] includes source and destination IP addresses for binary classification and destination IP Address for multi-classification, This may lead to biases and false negatives, if the model is trained on this feature set and deployed in a live network with similar IP Address traffic, it may incorrectly flag the IP as malicious. Another minor thing is that the paper says it proposes 17 features and then provides a list of 18. for this reason, all features that will produce a bias such as source and destination IPs.

2.6 Related Works Results

- Most abbreviations can be found in the Abbreviations page
- The results displayed are an extensive summary, only the performance metrics in **bold** and *italics* are listed in the results section

Table 2.1: Summary of results from related works

Paper	**FS Algorithm**	**Classifier(s) Used**	**Dataset Used**	**Performance Metric(s)**	**Result**
[18]	-	Extra Trees ensemble classifier	NF-UNSW-NB15 NF-ToN-IoT NF-BoT-IoT NF-CSE-CIC-IDS2018 NF-UQ-NIDS	***Prediction time (µs)***, Accuracy, AUC, F1, DR, FAR	7.79 5.37 6.05 17.04 14.35
[20]	Chi2, Random forest, importance, Lasso L1	RF, AdaBoost(AB), NB and ANN	NF-UNSW-NB15, NF-BoT-IoT, NF-CSE-CIC-IDS2018, NF-UQ-NIDS, NF-ToN-IoT.	***F1-Score***, accuracy, Precision, Recall, F1, (BCC), (MCC) AUC_ROC	RF 0.98 AB 0.94 NB 0.64 ANN 0.97
[21]	Feature correlation, Feature importance	DT, RF, XGB, and NB	NF-ToN-IoT-v2	***Accuracy***, PR, RC, F1-Score	DT,RF =1 XGB =0.98 NB =0.62
[19]	BAOA, BBAT, BWSO and BGWO	DT, RF, XGB and NB	NF-ToN-IoT-v2, NF-UNSW-NB15-v2, NF-BoT-IoT-v2, NF-CSE-CIC-IDS2018-v2	***Accuracy, precision, DR, PR, F1-score***	All Metrics greater than 0.995

Continued on next page

Table 2.1: Summary of results from related works (Continued)

[23]	SVM, PCA	KNN, RF, CNN, CNN-LSTM, DNN, CNN-LSTM-SVM	CICIDS2017	***Accuracy***	0.901, 0.8848, 0.9165, 0.9361, 0.9561, 0.9729
[22]	Feature importance	KNN, XGB, CART, CNN, LSTM	CIDDS-001, CIDDS-002, CIC-IDS2017	***Mean Accuracy***, Training Time	All models scored greater than 0.97253
[9]	Pearson Correlation	2-Level-classifier, DT, RF, HAST-I(100/300)	UNSW-NB15, CICIDS2017	***F1-Score***, accuracy, precision, recall	0.98 (DT), 0.982 (RF)

2.7 Proposed Feature Sets

Features proposed by: [20] *both Binary and Multi-Classification (10 Features):*

1. FLOW_DURATION_MILLISECONDS
2. TCP_WIN_MAX_IN
3. DURATION_OUT
4. MAX_TTL
5. L7_PROTO
6. SRC_TO_DST_AVG_THROUGHPUT
7. SHORTEST_FLOW_PKT
8. MIN_IP_PKT_LEN
9. TCP_WIN_MAX_OUT
10. OUT_BYTES

Features proposed by: [21] *for Binary Classification (13 Features):*

1. SRC_TO_DST_SECOND_BYTES
2. LONGEST_FLOW_PKT
3. L4_DST_PORT
4. TCP_WIN_MAX_IN
5. IN_BYTES
6. L7_PROTO
7. IPV4_SRC_ADDR
8. IPV4_DST_ADDR
9. MAX_IPvPKT_LEN
10. SRC_TO_DST_AVG_THROUGHPUT
11. L4_SRC_PORT
12. FLOW _DURATION_MILLISECONDS
13. SHORTEST_FLOW_PKT

Features proposed by: [21] *for Multi-Classification (18 Features):*

1. LONGEST_FLOW_PKT
2. IN_BYTES
3. MAX_IP_PKT_LEN
4. SRC_TO_DST_SECOND_BYTES
5. L4_DST_PORT
6. SRC_TO_DST_AVG_THROUGHPUT
7. L7_PROTO
8. TCP_WIN_MAX_IN
9. IPV4_DST_ADDR
10. DST_TO_SRC_SECOND_BYTES
11. SHORTEST_FLOW_PKT
12. OUT_BYTES
13. MIN_IP_PKT_LEN
14. FLOW_DURATION_MILLISECONDS
15. DST_TO_SRC_AVG_THROUGHPUT
16. CLIENT_TCP_FLAGS
17. L4_SRC_PORT
18. TCP_WIN_MAX_O

Features proposed by: [19] *for Binary Classification (12 Features):*

1. PROTOCOL
2. IN_PKTS
3. TCP_FLAGS
4. SERVER_TCP_FLAGS
5. DURATION_OUT
6. MAX_TTL
7. MAX_IP_PKT_LEN
8. RETRANSMITTED_OUT_PKTS
9. DST_TO_SRC_AVG_THROUGHPUT
10. NUM_PKTS_1024_TO_1514_BYTES
11. TCP_WIN_MAX_IN
12. DNS_QUERY_ID

Features proposed by: [19] *for Multi-Classification (14 Features):*

1. PROTOCOL
2. IN_PKTS
3. OUT_BYTES
4. TCP_FLAGS
5. CLIENT_TCP_FLAGS
6. SERVER_TCP_FLAGS
7. SHORTEST_FLOW_PKT
8. MAX_IP_PKT_LEN
9. RETRANSMITTED_OUT_PKTS
10. NUM_PKTS_1024_TO_1514_BYTES
11. TCP_WIN_MAX_IN
12. TCP_WIN_MAX_OUT
13. ICMP_TYPE
14. DNS_QUERY_ID

Chapter 3

Experimentation & Implementation Methodology

3.1 Machine Learning & Deep Learning Experimentation

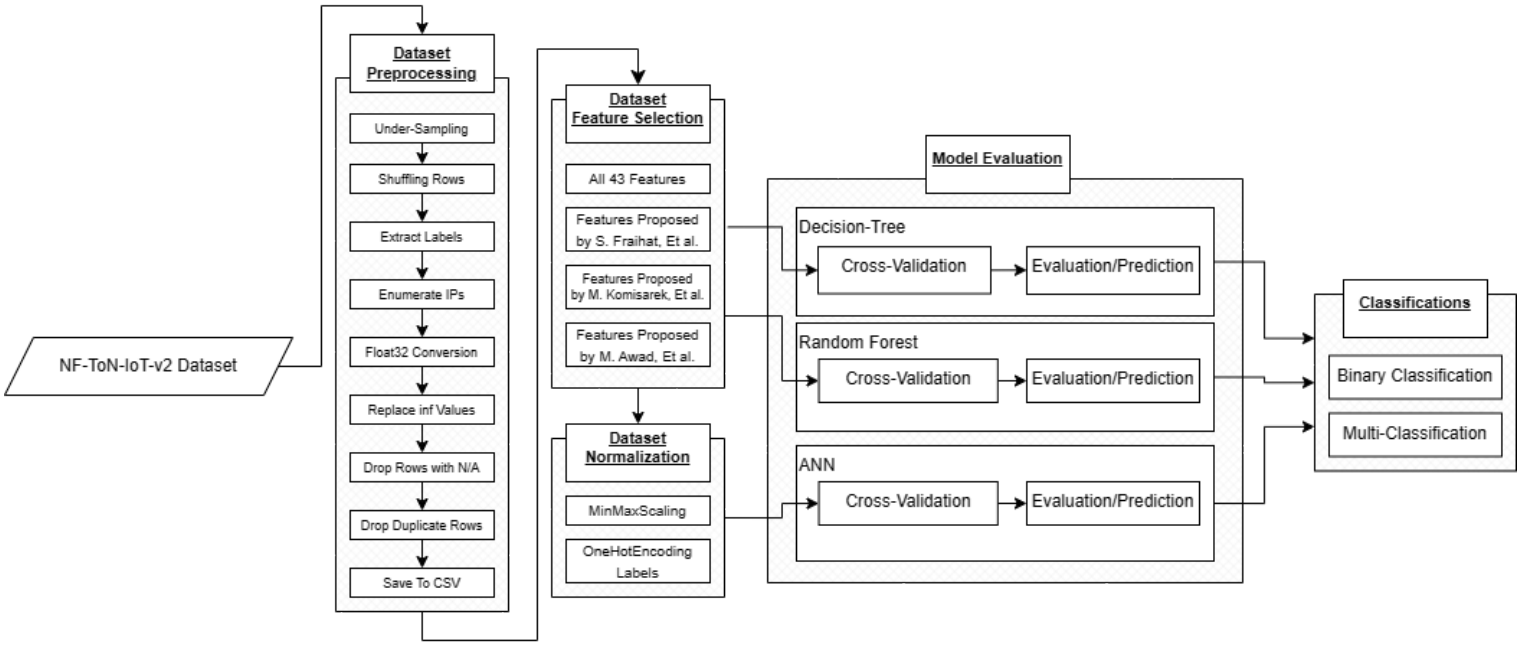


FIGURE 3.1: The process used for testing

3.1.1 Data Preprocessing

The NF-ToN-IoT-v2 [1] dataset was used for training, as it was the common link utilized by all previous works that provide this dissertation with the feature subsets used to train the models.

Under-Sampling: The NF-ToN-IoT-v2 dataset presented a challenge as it was simply too large to train models on, as it would take too long to train and thus was infeasible, and so only 40% of the majority classes ("Benign", "scanning", "xss", "ddos", "password", "dos", "injection") were kept while keeping the minority classes ("backdoor","mitm","ransomware") intact as theit conts were too few as it stood. This approach manages to keep a similar proportion to the initial dataset, mimicking real world

Multi-Classification Labels					
Class	**Before**	**After**	**Class**	**Before**	**After**
Benign	6099469	2439787	**dos**	712609	285043
scanning	3781419	1512567	**injection**	684465	273786
xss	2455020	982008	**backdoor**	16809	16809
ddos	2026234	810493	**mitm**	7723	7723
password	1153323	461329	**ransomware**	3425	3425
Binary Classification Labels					
Benign	6099469	2439787	**Attack**	10841027	4353183

TABLE 3.1: Count of Each Label

traits while reducing the computational requirements to train the models. The new proportions of the dataset can be seen in figures (3.3 & 3.4) as well as table 3.1 below, For a look at the proportion of binary classes, see Figures C.1.

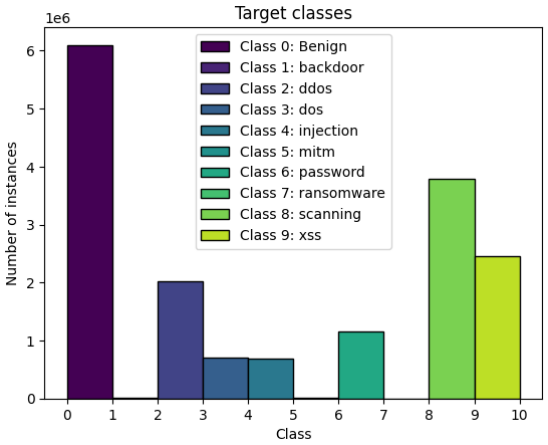


FIGURE 3.2: Before Under-Sampling

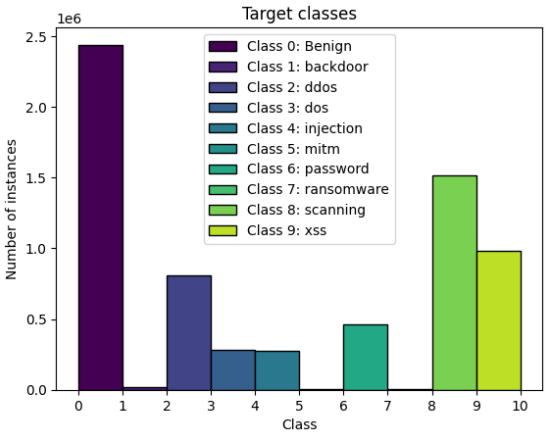


FIGURE 3.3: After Under-Sampling

Shuffling Dataset: The under-sampled dataset is then shuffled, a common practice used to avoid any of the models from overfitting, as if it is presented with ordered/sequential data it may pick up patterns that don't reflect what happens in the real world [24].

Modification of Data in Dataset Values:

- **Extracting Labels:** In this step the column "Attack" and "Label" are removed, making the total number of features 43.
- **Enumerating IP Addresses:** As ML and DL models work best with numerical values the IP Addresses are enumerated into an integer format using pythons ipaddress package
- **Float32 Conversion:** Two specific columns caused this step to be necessary, namely: SRC_TO_DST_SECOND_BYTES and DST_TO_SRC_SECOND_BYTES.

These columns are stored by default using the float64 datatype. Problems arose when passing these fields into a model as the remaining features were stored in int32 or float32, and so the fields were converted to float32 and when value couldn't be reprented in 32 bits it was replaced with 'inf'.

- **Replacing 'inf' values:** The previous step left inf values in the dataset, these were quickly replaced by the maximum float32 value(3.4028235e+38) to preserve the "spirit/impact" of the large value.
- **Dropping rows with N/A values:** Removing any rows with N/A values is important as this can cause the classifier/Model to see the lack of data as a correlation with a specific label, This is bad as in real-world applications there may be missing values for certain features which can lead to an incorrect classification.
- **Dropping rows with duplicate rows:** Similarly duplicate records can cause the classifier to assume an irrelevant characteristic of the duplicated record to have a correlation with the class type.
- **Saving new dataset:** As all the processes mentioned above are time intensive, we save the dataset to forgo having to do these steps over and aver again.

Dataset Normalization: Only used for the ANN as without this step the value ranges of the features vary greatly, causing the ANN's loss function to go to NaN as the loss values dramatically changes as every new row is introduced, eventually turning into NaN, and an ANN with a Nan value leads to an unstable model that produces incorrect predictions.

The following techniques were utilized:

- **Min-Max Scaling:** The formula $x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$ is applied to all elements within a column, where x_{min} is the lowest value in the column, x_{max} is the largest value, x is the current value and x_{scaled} is the new normalized value.
- **One-Hot Encoding:** Transforms categorical values from the Attack column into binary representations that match the output of an ANN. The transformation was done using a OneHotEncoder()

3.1.2 Dataset Feature Selection

Rather than doing feature selection ourselves on the features of the dataset, which is listed listed at: C.1, We perform A comparative analysis of the feature sets found by the papers discussed in the related works section that can be found in table 2.7 was performed.

Namely Feature sets proposed by:

- Salem Fraihat Et al.[19], Which consists of 2 subsets for binary and multi-classification containing 12 and 14 features respectively.
- Mikołaj Komisarek Et al. [20], Which consists of a single subset for binary and multi-classification containing 10 features.
- Mohammed Awad Et al. [21], Which consists of 2 subsets for binary and multi-classification containing 13 and 18 features respectively.
- And finally all 43 features of the dataset as a baseline. Totaling 6 subsets tested.

3.1.3 Models Experimentation

The 3 models that were trained using these feature sets and the original 43 features, were Decision Tree (DT), Random Forest (RF), and Artificial Neural Network (ANN) for binary classification and multi-classification.

For the decision tree classifier, the standard model is used from sklearn.tree package without any modifications.
The Random forest classifier is from the sklearn.ensble package and the ANN classifier comes courtesy of the tensorflow package. with parameters/layer definitions asper the findings of : [20], initially the paper only defines the loss_function as catagorical_crossentropy, however, binary_crossentropy was also utilized for binary classification.

Model	Parameter	Value
RF	n_estimators	200
	max_features	auto
	max_depth	8
	criterion	entropy
ANN	epochs	16
	batch_size	20
	loss_funcrion	catagorical_crossentropy/ binary_crossentropy

TABLE 3.2: Parameter values for RF and ANN

Table 3.3: ANN Layer Structure

Layer (type)	**Output Shape**	**Activation Function**
Input	X	-
Dense	(None, 64)	relu
Dense	(None, 32)	relu
Dense	(None, Y)	sigmoid/softmax

The table to the right 3.3 depicts the layers of the ANN that wast utilized, The value X depends on the amount of features in every subset, and Y depends on the type of classification, binary = 1, multiclass = 10.

Additionally, while doing Cross-Validation and fitting, eatly stopping was used to not waste computational resources and time, with the metric checked being the loss value and the patience being 3.

3.1.3.1 Cross-Validation

Cross-Validation(CV) is a robust measure against overfitting [25], which is when a model performs well during testing, as it is evaluated based on information it was trained with, but doesn't generalize well to unseen data, leading to bad predictions. It works by passing the reduced dataset to the stratified K-fold algorithm, which splits the dataset into k folds while maintaining the proportion of the classes. K-1 folds are then used to train a new model and the remaining fold is used for evaluation. This process is also known as the hold 1 out method. Additionally, the data is shuffled again to prevent the model from learning irrelevant patterns.

The K value chosen for this experiment is 5, meaning that the model will be trained and evaluated 5 for 5 iterations. For cross validation of the DT and RF classification models, the cross_validate function from sklearn.model_selection was used, while for the ANN a custom solution that essentially achieved the same goal was created. For each iteration, the evaluation fold and results are saved and aggregated at the end to give an overall summary of the model.

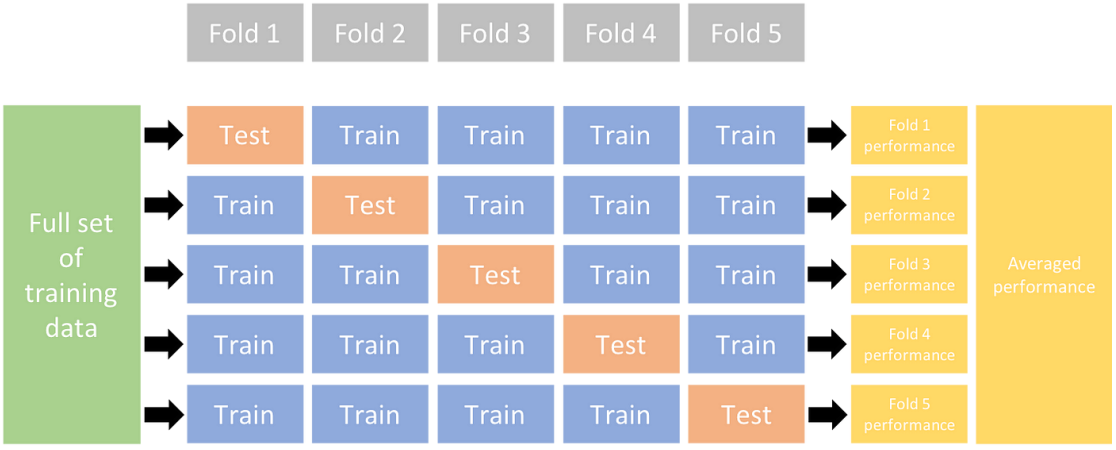


FIGURE 3.4: Cross-Validation Method Used

Prediction/Evaluation: Once it was verified that each model combined with each of the 6 subsets was not overfitting, each model was then trained with 70% of each subset separately and evaluated using the remaining 30% of the corresponding subset for both binary and multi-class prediction. The split was done using train_test_split with stratification. Better describing the process is the pseudocode below, namely, pseudocode 1:

Algorithm 1: ML/DL Model and Subset Evaluation Pseudocode

Input: Cleaned-NF-ToN-IoT-v2.csv
Output: CV Scores, EvaluationScores, Confusion Matrices

```
models ← [DTClassifier(), RFClassifier(), ANNClassifier()]
proposedBinarySubsets ← [allFeatures, sFraihatBinSubset, mKomisarekSubset,
  mAwadBinSubset]
proposedMultiSubsets ← [allFeatures, sFraihatMultiSubset, mKomisarekSubset,
  mAwadMultiSubset]
allDatasets ← [proposedBinarySubsets, proposedMultiSubsets];
foreach model in models do
    foreach classificationTypeDataset in allDatasets do
        foreach subset in classificationTypeDataset do
            cross_validate(model,subset);
            train(model,subset);
            evaluate(model,subset);
        end
    end
end
```

3.1.3.2 Evaluation Metrics

Metrics analyzed for DT and RF during CV were: 'accuracy', 'precision_macro', F1_macro and 'recall_macro. While for ANN the metrics analyzed was the 'accuracy' and 'loss'.

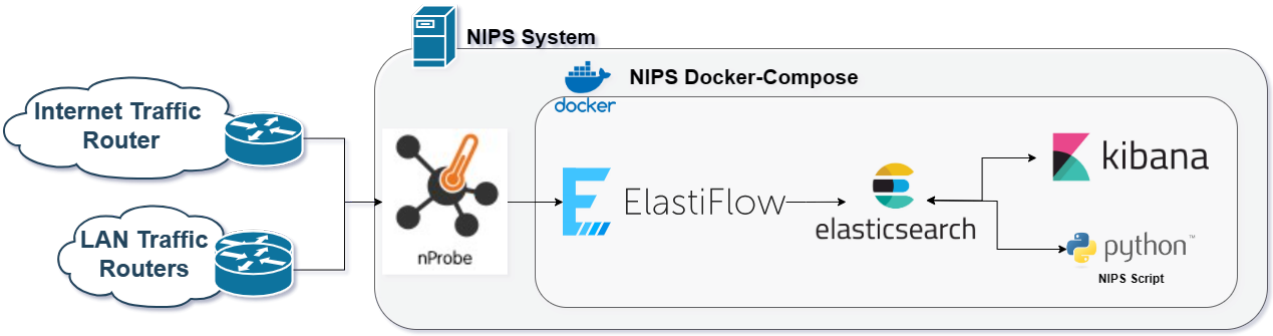
During Evaluation using train-test-split, Accuracy, precision, recall, F1 scores and time taken for fitting and classification were all be recorded along with confusion matrices.

3.2 NIPS System Implementation

It is important to note that when the models were deployed in the manner described below, they were be trained on the whole data set to learn intricate and minute differences of different flows within the dame classification, Failure to do so would have resulted in misclassifications and reduced predictive performance in real-world scenarios. The remaining steps in data preprocessing such as shuffling, extracting labels, etc... were done

When discussing the overall architecture of this project it can get confusing and hence why the terms in the next page will be used for clarity and are NOT interchangeable:

- **NIPS System:** The Windows 10 virtual machine depicted in figure 3.11 that runs docker, manages static routing and has windows firewall installed.



- **nProbe:** Is an extensible NetFlow probe/collector and proxy. For our use case it is a probe that collects, analyzes and exports network traffic in the NetFlow v9 format to ElastiFlow which will then be parsed, stored in elasticsearch and later be used by both kibana and the Level 1 and 2 classifiers in the NIPS Script.

- **NIPS Docker-Compose:** Which is a collection of individual docker containers running on the NIPS System consisting of:

  - **ElastiFlow:** The flow collector which listens on UDP port 2055 on the NIPS system where nProbe sends CFLOW packets which is the packet format in which netflow data is sent. Once it recieves a packet it is parsed before it is exported to Elasticsearch in the form of a DataStream to be stored
  - **Elasticsearch:** The centralized database with powerful querying and indexing capabilities where the flows that ElastiFlow sends adr stored/
  - **Kibana:** Kibana is a powerful visualization tool that displays all flow information in easy-to-read and comprehensible dashboards that are filled with various pie charts, Sankey diagrams, etc. This dashboard web gui used by SOC analyst to get overview of network.
  - **NIPS Script:** Contains a 2-Level Classifier inspired by [9] where Level 1 is a fast and accurate binary classifier, and level 2 is a more accurate multi-classification classifier. If both levels detect an attack, an alert is sent to kibana with the attack type, and a new IPTables rule is added blocking packets from that IP.

- **Firewall:** The firewall runs on the NIPS system and allows traffic to flow from one subnet to the other, not to mention communication to the internet. If an IP is blocked, packets that are intercepted by the system are dropped. As docker can run on windows, linux and more, it is important to note that the firewalls used by each are OS are different, and so a command for both IPTables [26] and windows firewall have been implemented.

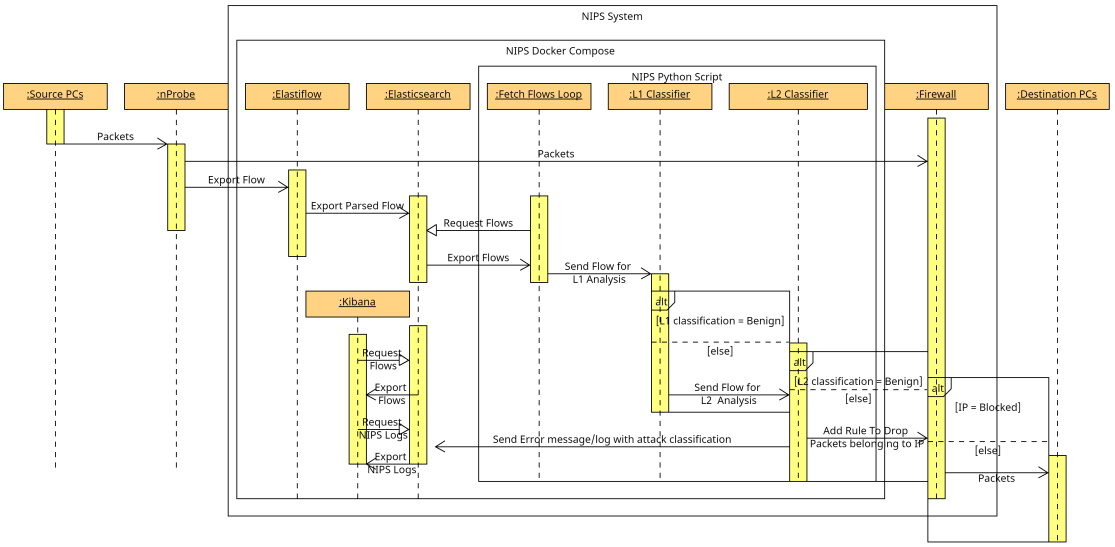


FIGURE 3.5: Flow of Execution

The way the proposed system works as depicted by the sequence diagram 3.5 is:

When network traffic flows through the routers it points traffic traveling to different subnets or the internet to the NIPS System which has nProbe running on all internal interfaces. nProbe creates an aggregated summary of the conversation between the source and destination, This aggregated flow is then exported every 60 seconds (whether the conversation is over or not) to the NIPS Systems own to UDP port 2055, Where the ElastiFlow container is listening. When a flow/record is sent it is then parsed by ElastiFlow before it is exported to Elasticsearch in the form of a DataStream to be stored.

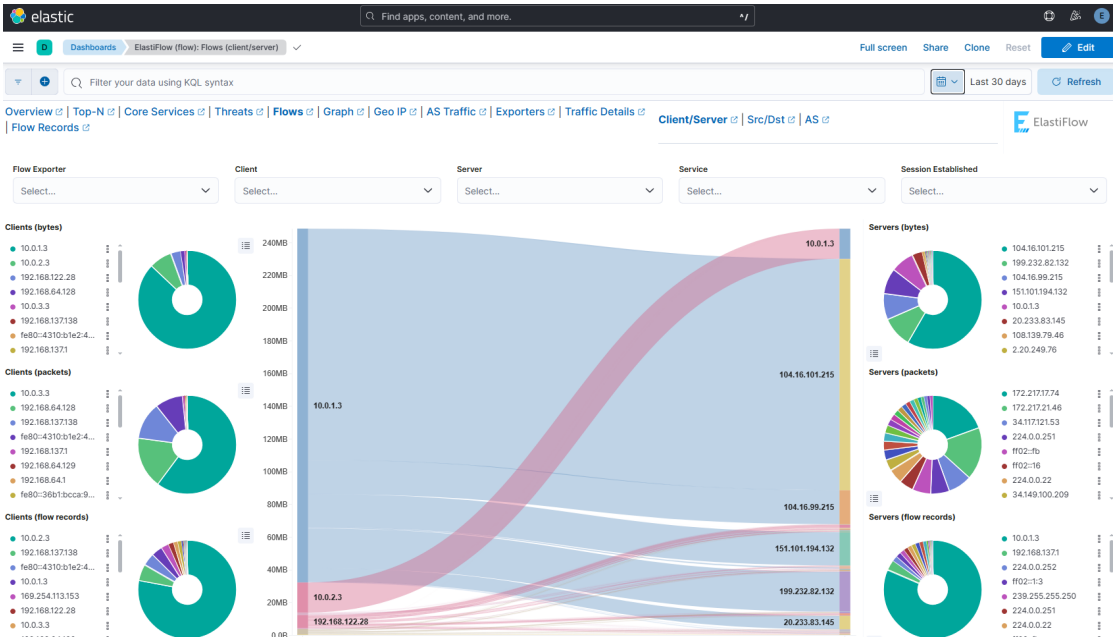


FIGURE 3.6: Kibana Sankey Diagram Visualization

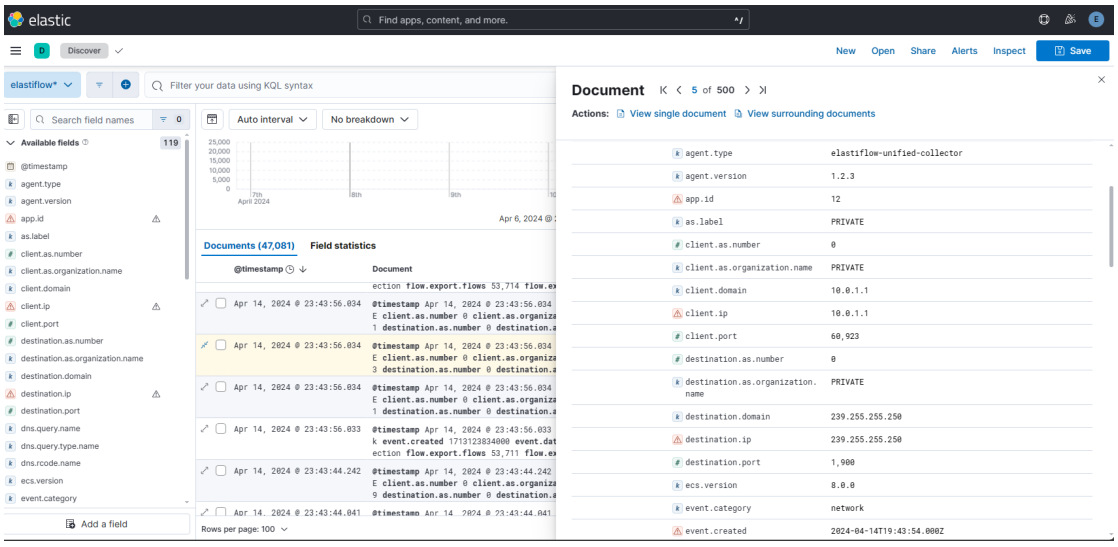


FIGURE 3.7: Kibana Elastiflow's Exported Logs

From this central database, both Kibana (as seen above in figs. 3.6& 3.7) and the NIPS Python script can acquire the flows. The Main loop in the Python script will query Elasticsearch every second from new records (flows), If any are found it, the flow is immediately mapped from ElastiFlow's parsed feature names back to the feature names used in the datasets and while training, mapping can be found in Table: C.2. The record is then passed into the Level 1 binary Classifier and if the prediction is equal to 1 (meaning attack), the flow is then sent to the Level 2 multi-class classifier for further classification. The event is then logged to kibana via the dedicated NIPS Logs datastream (as seen in figure 3.9) to elasticsearch, and the terminal where the script is running, with details about the flow regardless if it is classified as benign or any of the other attacks depending on the log level chosen by the user. Finally, after logging, if the user has turned on NIPS mode by changing the global variable "NIPS_MODE" defined in the .env file at root, or within the python of Dockerfile. An example record can be seen belos.

FIGURE 3.8: closer look at flow record in Kibana

As this NIPS Docker-Compose is not platform specific, meaning it can run on Linux, Windows and even Mac, each system has its own firewall implementations. And so commands to block an IP for windows and linux have been implemented and are passed to the host OS by making an SSH connection from the container to the host, it is up to the user to setup SSH and pass it to the container through various ways described in the user guide. The code repository for the docker-compose stack can be found at my github [27], along with the setup/user guide.

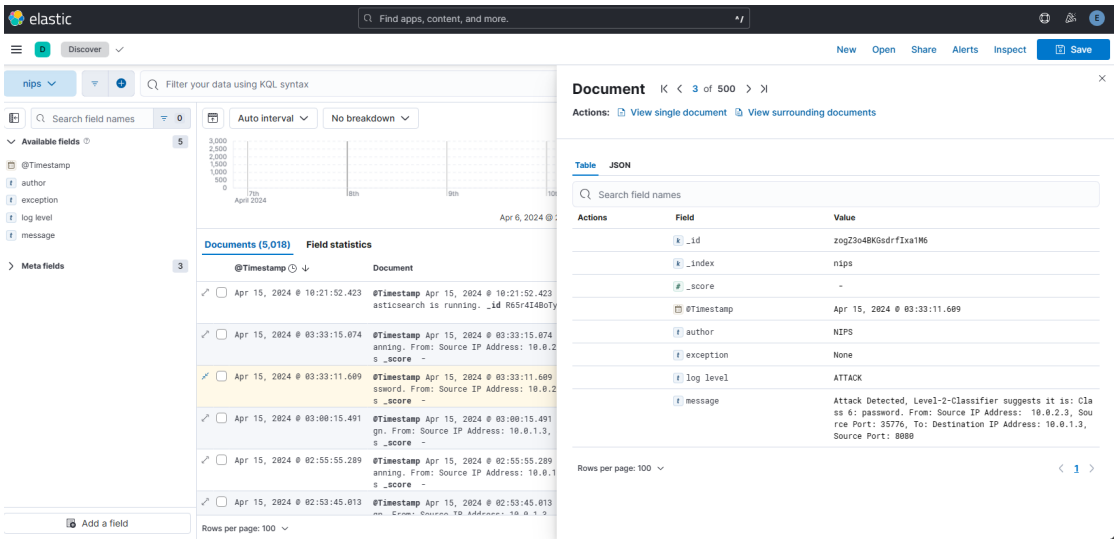


FIGURE 3.9: Kibana NIPS Logs

3.3 Network Setup & Attack Simulation

3.3.1 GNS3

The testing and development was carried out in a virtualized environment using Graphical Network (GNS3) which is a powerful network emulator that is used for testing network configurations without the use of additional hardware [28]. In addition, many appliances (Emulations of real hardware) are available such as cisco routers, jupyter servers, switches, docker containers etc.... As per recommendation of GNS3 all appliances (switches, docker containers and VPCs) were run using the GNS3-VM running on VMWare.

The network setup is simple with each interface from the windows 10 machine, and the machines connected to them, being in a different subset, IP Addresses of each machines interface can be seen in 3.11and all machines are capable of connecting to the internet as all requests end up in the windows 10 machine and it

has a default route pointing the ubuntu-router container which has 1 DHCP interface connected to the NAT node which uses the host machines WI-FI card for access to the internet, and another NIC connecter to

```
Persistent Routes:
  Network Address          Netmask  Gateway Address  Metric
        10.0.0.0    255.255.255.0         10.0.0.1       5
        10.0.1.0    255.255.255.0         10.0.1.1       5
        10.0.2.0    255.255.255.0         10.0.2.1       5
         0.0.0.0          0.0.0.0         10.0.0.2  Default
```

FIGURE 3.10: Routes configured on windows 10

the windows on startup the following command is run, which enables network address translation on eth0 and ads an IPTables rule to forward all packets destined to a local ip back to windowws 10:

```
ip route add 10.0.0.0/16 via 10.0.0.01 &&
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

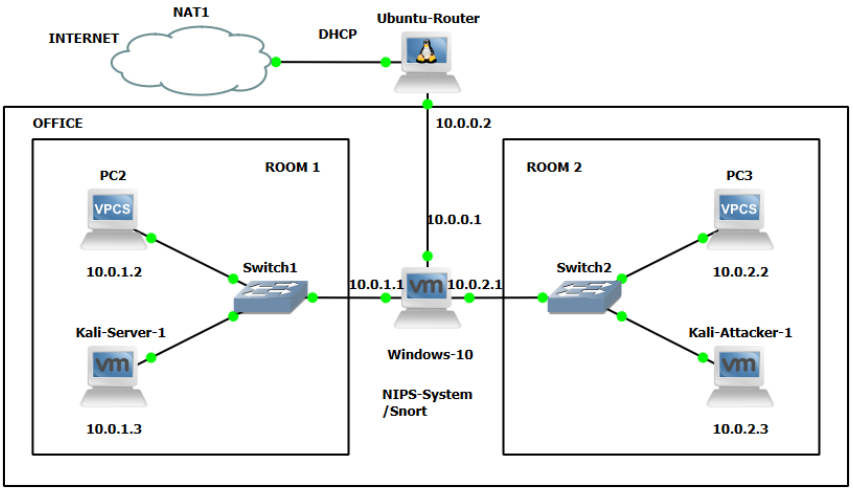


FIGURE 3.11: GNS3 Testing Setup used for developing NIPS

3.3.2 VMWare

VMWare [29] is a powerful desktop virtualization software (also known as a type 2 hypervisor), that integrates well with GNS3 and enables VMs to be placed within the virtual networking infrastructure made in GNS3 allowing communication between VMWare VMs, docker containers, VPCs, switches and more.

Many VMs were deployed and the configuration of each is listed in Table C.3, Additionally the Ubuntu-Router was running on the GNS3 utilizing its allocated resources.

3.4 NIPS Testing Methodology

The proposed NIPS was tested against a well-known, open-source NIDS called snort which is a packet based detection system and prevention system (NIDS/NIPS). After installing snort on a separate windows 10 VM with the same configurations (cores, memory, ...), the attacks listed below were performed twice on the topology once for each system. Additionally Network speeds, while each systems was deployed was recorded using speedtest.net.

For evaluating the NIPS, the correctness of the classifications of both Level 1 and level 2 classifiers log were taken into account, while for snort, the predicted attack along with the priority level that is provided in snort's was used for evaluation. The scale snort uses is defined as high (1), medium (2), low (3), and very low (4). Additionally, Snort 2.9.20 was evaluated using its community tier rules.

3.4.1 SQL Injection

A code injection technique that uses a text field on a website to send a malicious SQL command to a webserver to manipulate the back-end database by either changing values, viewing unauthorized information, or wiping the database. This attack uses a containarized website running on docker containers made by thomaslaurenson [30] who also provided example payloads as seen in his github.

```
# All payloads below were a separate test.
# Dump username and salary of all users:
    username: ' OR 1=1#
    password: anythingyouwant
# Dump MySQL version:
    username: ' UNION SELECT null,@@version#
    password: anythingyouwant
# Dump all users passwords:
username:' UNION SELECT username,password FROM users#
    password: anythingyouwant
```

3.4.2 Reconnaissance/Scanning

The practice of collecting information on a computer network or system illegally, by utilizing scanning methods to find vulnerabilities which can be exploited. This attack was performed using a popular Command-Line Tool called nmap [31] and more specifically, the following commands were used to perform many types of scans.

```
nmap -O 10.0.1.3  # OS fingerprinting is used to deduce what OS the host is using.
nmap -sT 10.0.1.3 # Does full TCP handshake with each port in host to determine open
                  # TCP ports.
nmap -sU 10.0.1.3 # Does UDP scan of each port in host to determine open UDP ports.
nmap -sS 10.0.1.3 # SYN Scan only does steps 1\&2 of a TCP handshake, neglecting to
                  # send the final RST packet as many firewalls only log full
                  # handshakes and so this is considered a stealthy scan.
nmap -sX 10.0.1.3 # Also known as an XMAS scan that identifies open TCP ports by
                  # sending a packet with URG, PUSH, and FIN flags set.
nmap -sN 10.0.1.3 # The null scan sends a packet with no flags to determine if TCP
                  # ports are open.
```

3.4.3 Denial of Service attack (DoS)

An attempt to overload a computer by making illegitimate requests so that genuine requests either time out or are never processed. We utilize the same webserver from the SQL injection server running on port 8080.

```
hping3 10.0.1.3 --flood -q -d 120 -p 8080
```

```
# --flood: sends as many TCP requests as posable disregarding replies
# -q: enables a less verbose output to the terminal.
# -d: defines the payload size
# -p: defines the port to be attacked.
```

3.4.4 Distributed Denial of Service attack (DDoS)

Similar to a DoS attack but with different/random IPs making illegitimate requests, making it harder to stop the attack.

```
hping3 10.0.1.3 --flood -q -d 120 -p 8080 --rand-source
# --rand-source: the source IP Address is different every time.
```

3.4.5 nProbe Setup

Thanks to the ntop team for providing a free copy of nProbe Enterprise as part of their free for education initiative, nprobe is a critical component as it is what produces the Information Elements that the classifiers use for prediction, It is to be noted that as the NIPS Docker-compose was run, at the same time, nProbe was running and sending NetFlow flows to kibana. During testing the following command was used:

```
.\nprobe /c -i 'Intel(R) PRO/1000 MT Network Connection #3' -V 9 -b 1 -t 60 -o
1 --export-template -T " %IPV4_SRC_ADDR %L4_SRC_PORT %IPV4_DST_ADDR
%L4_DST_PORT %PROTOCOL %FLOW_DURATION_MILLISECONDS %TCP_WIN_MAX_IN
%DURATION_OUT %MAX_TTL %L7_PROTO %SRC_TO_DST_AVG_THROUGHPUT %SHORTEST_FLOW_PKT
%MIN_IP_PKT_LEN %TCP_WIN_MAX_OUT %OUT_BYTES %FIRST_SWITCHED %LAST_SWITCHED "
```

-V, specifies the netflow format used, -t is the timeout for active conversations and -T specifies the features we want to be sent in the flow.

3.5 Project Management

- Usecase Scenarios along with NIPS system requirements are listed in the Appendix C.3.
- The risks associated with this project can be found in the Appendix B.2
- The project followed the agile methodology for building the NIPS and details for the project timelines can be found in Appendix B.1

Chapter 4

Feature Subset & NIPS Evaluation

4.1 Machine Learning & Deep Learning Evaluation

4.1.1 Evaluation Metrics

The following statistics will be used to determine which model works the best with which subset.

Table 4.1: Classification Metrics and Formulas

Metric	**Description**	**Formula**
Accuracy	The proportion of correct classifications with all classifications.	$\frac{TP+TN}{TP+TN+FN+FP}$
Precision	The proportion of correctly classified flows of a type to the total predicted flows of that type.	$\frac{TP}{TP+FP}$
Recall	Also known as True Positive Rate or Detection Rate, it is the proportion of true positive instances to all actual positive instances.	$\frac{TP}{TP+FN}$
F1-Score	The harmonic mean of precision and recall.	$\frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$
Specificity	Measures how sensitive the classifier is to negative cases. A high rate can mean legitimate requests are more likely to get blocked.	$\frac{TN}{TN+FP} \times 100$
True Negative Rate	The proportion of incorrectly classified Attack/Benign flows.	$\frac{TN}{TN+FP}$

Continued on next page

Table 4.1 – *Continued from previous page*

Metric	**Description**	**Formula**
False Positive Rate	Also known as the False Acceptance Rate.	$\frac{FP}{TN + FP}$
Training Time	The amount of time required for the model to be trained.	N/A
Classification Time	The time taken for either multi-classification or binary classification, this will be a very important metric as the proposed system will run in real time so the lower this value the better.	N/A
Confusion Matrices	Useful for visualizing performance of a model by plotting the amount True Positives and the False Positives in one grid.	N/A

Performance metrics used by other papers can be found in this table 2.1.

Precision, Recall and the F1-Score all use the macro averages of each of the metrics, meaning, that all classes equally contribute to the final result, whereas a in weighted averages, each classes contribution is proportional to the size of the class. Since our classes have vastly different sizes as seen earlier, these metrics take into account the performance of the model on every the minority classes big or small.

During testing, naming conventions for each of the subsets was used and are used for ease, below are the conventions that will be found in the result tables and plots

Table 4.2: Naming Conventions

Author	**Naming Convention**
Salem Fraihat Et al.[19]	paper18
Mikołaj Komisarek Et al.[20]	paper19
Mohammed Awad Et al.[21]	paper20

Finally, Subsets that performed the best in their classifier category are highlighted in green in tables: 4.4 and 4.5

Tables 4.3 and 4.6, are the cross-validation results of the process described in the Implementation section 3.1.3, While tables 4.4 and 4.5, are the results of the evaluation process described in 3.1.3.1, with mode detailed results in Appendix C tables C.8 and C.9

4.1.2 Binary Classification Results

The Cross-Validation results for binary classification clearly show that Decision-Trees performed well in all 4 subset tests, achieving 99% accuracy and excellent precision, recall and F1-scores. This leads to the conclusion that for binary classification is a robust classifier along with all the other classifiers who all have have accuracies ¿ 94%.

Classifier Model	Subset	Accuracy	Precision	Recall	F1	Loss
DT	All 43 Features	0.9958	0.9954	0.9955	0.9954	-
	Paper 18	0.9476	0.9372	0.9544	0.9443	-
	Paper 19	0.9960	0.9960	0.9952	0.9956	-
	Paper 20	0.9958	0.9955	0.9953	0.9954	-
RF	All 43 Features	0.9807	0.9780	0.9801	0.9791	-
	Paper 18	0.9434	0.9331	0.9489	0.9397	-
	Paper 19	0.9789	0.9806	0.9737	0.9770	-
	Paper 20	0.9831	0.9827	0.9806	0.9816	-
ANN	All 43 Features	0.9895	-	-	-	0.0333
	Paper 18	0.9430	-	-	-	0.1709
	Paper 19	0.9773	-	-	-	0.0700
	Paper 20	0.9872	-	-	-	0.0418

TABLE 4.3: Binary Classification Cross-Validation Resulst

Models	Subsets	Training Time (s)	Prediction Time (s)	Precision	Recall	F1-Score	Accuracy
DT	All 43 Features	95.36	0.80	1.00	1.00	1.00	1.00
	Paper 18	17.10	0.29	0.95	0.95	0.95	0.95
	Paper 19	11.63	0.28	1.00	1.00	1.00	1.00
	Paper 20	34.11	0.36	1.00	1.00	1.00	1.00
RF	All 43 Features	759.37	28.11	0.98	0.98	0.98	0.98
	Paper 18	414.11	25.37	0.94	0.94	0.94	0.94
	Paper 19	437.56	25.69	0.98	0.98	0.98	0.98
	Paper 20	655.92	25.10	0.98	0.98	0.98	0.98
ANN	All 43 Features	3022.27 (50.37m)	60.28	0.99	0.99	0.99	0.99
	Paper 18	2881.44 (48.02m)	50.34	0.94	0.94	0.94	0.94
	Paper 19	2898.68 (48.31m)	50.39	0.97	0.97	0.97	0.97
	Paper 20	2899.22 (48.32m)	52.61	0.98	0.98	0.98	0.98

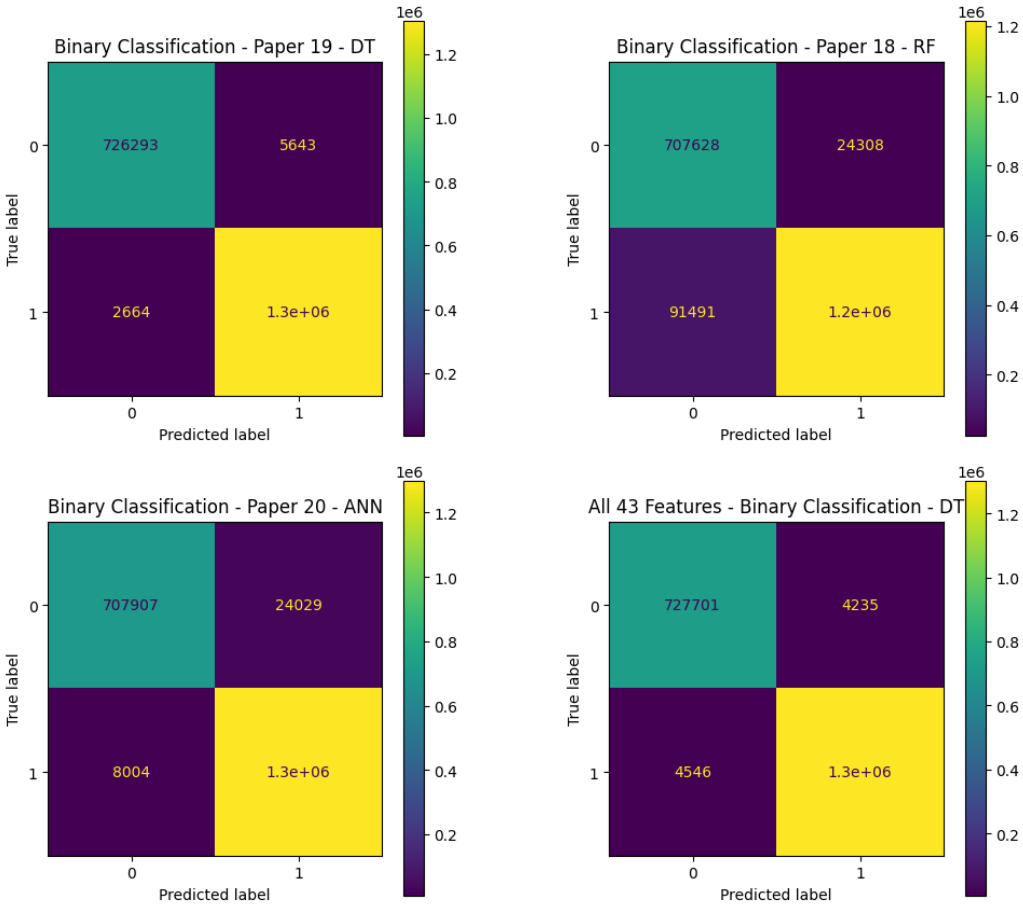
TABLE 4.4: Summarized Binary Prediction Results

Similarly, Random Forest performed well as it achieves similar but lower values in literally every category, around 0.1-0.2% in the CV results, These are still respectable scores, however, the time taken for training and prediction as seen in 4.4 let it down as they were a considerable magnitude larger than decision tree for slightly worse scores. Both the Rf model proposed by [20] and a stock RF classifier were tested and while the proposed model did train faster using the proposed FR model, the scores were not that different. All scores listed relating to RF are based on the proposed model.

And finally, the proposed ANN model by [20] also performed well but was ultimately let down by long training and more importantly long prediction times while utilizing a lot of CPU resources and RAM. And so for Binary prediction which will be used in Level 1, the choice is clear, The Decision-Tree classifier is clearly superior when based on an accuracy to prediction time ratio and will be paired with the subset created by Salem Fraihat Et al.[19] also known as paper 18, which is highlighted in green in table 4.4.

Below is the Confusion matrices of the 3 highlighted rows in table 4.4 and the DT result using all 43 features as that was the top performer, the remaining 8 will be in Appendix. C.3.1

FIGURE 4.1: Confusion Matrices the of top Binary Classification subset from each Model



4.1.3 Multi-Class Classification Results

The results of the Multi-Class classification evaluation, unlike the binary evaluation, actually had pros and cons to each model, while DT was still on top on all subsets for accuracy, RF was found to be better than ANN in training and prediction times, precision and tied when it comes to F1-Scores as RF performed better in some subsets (papers 18 and 19) and worse in others, and better than DT in precision in 2 subsets and tied in 1, meaning it correctly classified different classes more times than DT did.

Model	Model	Training Time (s)	Prediction Time (s)	Precision	Recall	F1-Score	Accuracy
DT	All 43 Features	61.91	0.98	0.92	0.92	0.92	0.98
	Paper 18	14.84	0.42	0.88	0.85	0.86	0.91
	Paper 19	9.00	0.40	0.96	0.90	0.92	0.98
	Paper 20	34.76	0.62	0.94	0.94	0.94	0.98
RF	All 43 Features	793.38	47.31	0.94	0.81	0.85	0.93
	Paper 18	428.56	42.26	0.91	0.74	0.79	0.89
	Paper 19	452.91	44.19	0.94	0.79	0.83	0.94
	Paper 20	774.50	41.99	0.94	0.81	0.84	0.94
ANN	All 43 Features	3696.36 (61.61 m)	66.21	0.93	0.85	0.87	0.96
	Paper 18	3202.28 (53.38 m)	64.32	0.84	0.78	0.79	0.89
	Paper 19	3867.02 (64.45 m)	89.82	0.87	0.80	0.81	0.93
	Paper 20	4184.44 (69.74 m)	89.68	0.90	0.85	0.86	0.95

TABLE 4.5: Summarized Multi-Classification Prediction Results

Multi classification only happens on the Level 2 classifier in the NIPS python script, and since we the level 1 classifier has already determined there is an attack, it is more important for L2 to classify the flow correctly.RF and ANN it still is much slower in prediction tine and training time compared to DT, Additionally, DT has a better F1-Score as compared to RF and ANN, which means that more classifications are accurate.

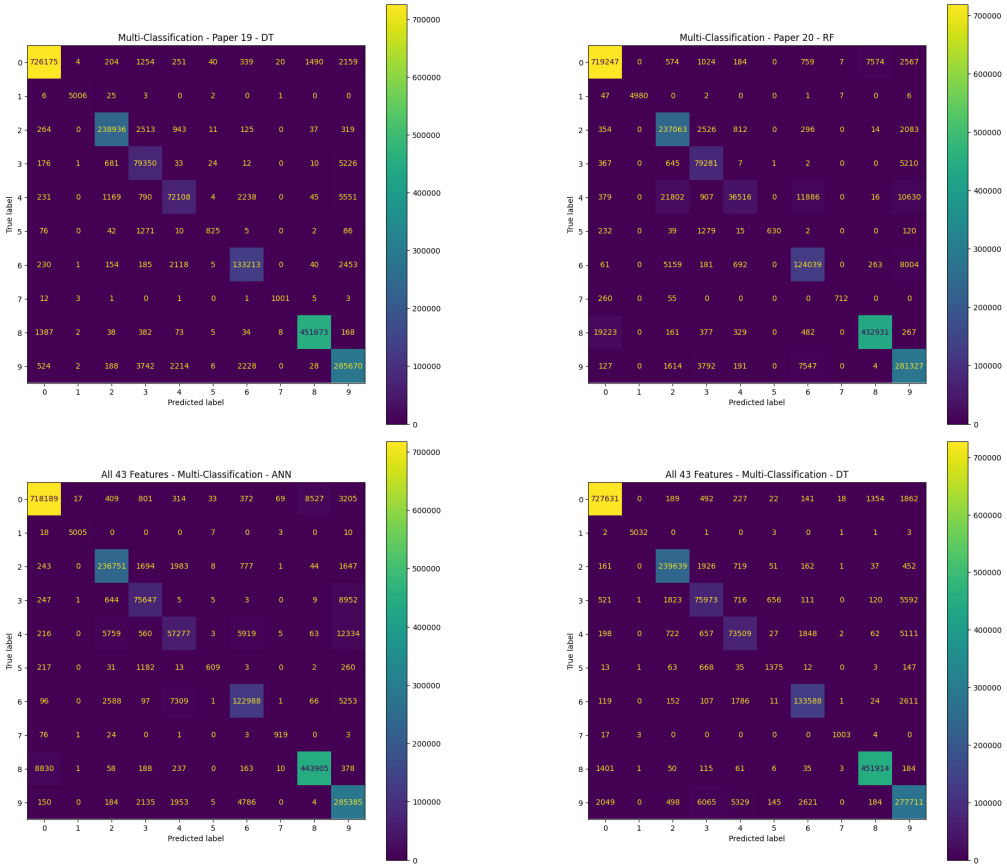
As further emphasized by the confusion matrices in figure 4.2 DT has the lowest number of false positives when it comes to class 0 which is benign flows, this is important as it the NIPS classifiers are is strict in classification, normal flows will be seen as an attack and it will end up up blocking the IP for no reason. And so for the Level 2 classifier, A Decision tree classifier will be used.

Finally, the Cross-Validation results also seem to point to the Decision -Tree classifier not only being robust, but also having the highest F1-scores means that most classifications made are accurate.

Classifier Model	Subset	Accuracy	Precision	Recall	F1	Loss
DT	All 43 Features	0.9756	0.9259	0.9276	0.9267	-
	Paper 18	0.9146	0.8779	0.8510	0.8622	-
	Paper 19	0.9789	0.9569	0.9041	0.9192	-
	Paper 20	0.9784	0.9423	0.9438	0.9430	-
RF	All 43 Features	0.9338	0.9444	0.8069	0.8455	-
	Paper 18	0.8867	0.9152	0.7434	0.7868	-
	Paper 19	0.9353	0.9387	0.7889	0.8291	-
	Paper 20	0.9432	0.9428	0.8149	0.8489	-
ANN	All 43 Features	0.9557	-	-	-	0.1536
	Paper 18	0.8900	-	-	-	0.3476
	Paper 19	0.9315	-	-	-	0.4135
	Paper 20	0.9500	-	-	-	0.1636

TABLE 4.6: Multi-Class Cross-Validation Results

FIGURE 4.2: Confusion Matrices of the top Multi-Classification subset from each Model



4.1.4 Feature Subset Evaluation

The all 43 features dataset was used as a baseline for subsequent evaluations and it set a very high standard across the board for all models to strive to achieve, however it was often the dataset that took the longest to train and generate predictions, additionally flow indicators such as source and destination IP addresses, source and destination ports were also present and could be influencing the results as the model could be associating certain IP addresses with certain classifications, typically this is bad however it was decided to leave it in, as IP addresses from certain geographic location such as Russia are more likely to be malicious, not that all russian IPs are malicious.

Next, the dataset by Salem Fraihat Et al.[19] (paper 18), Which consists of 2 subsets with 12 features for binary and 14 features for multi-classification, it typically ran the fastest, but also usually had the worst scores and generalizability while still being objectively good but still paling in comparison.

Mikołaj Komisarek Et al. [20] (Paper 19) proposed a subset, Which consists of a single subset for binary and multi-classification containing 10 features. This was without doubt the best subset as it had both fast training and prediction times while also having the best scores and being generalizable. This feature subset was used for both Level 1 and Level 2 classifiers.

And finally, Mohammed Awad Et al. [21]who proposed a subset, Which consists of 2 subsets for 13 features for binary and 18 features for multi-classification. This subset performed well from a metric standpoint, it usually took about the same time if not longer than all 43 features to train but once it trained it was faster than paper19 in classification, however, the author chose to leave the IP adresses with in both feature subsets, which could cause bias, and when the source and destination IP addresses were removed, the prediction and training times did not change much but the FP rate did increase, as many benign flows were now being classified as attacks, a difference of 20000 flows were now classified as attacks.

4.2 NIPS Evaluation

As mentioned in Chapter 4, The proposed NIPS system was developed and tested using the popular and open-source NIPS called snort as a baseline, specifically snort 2.9.20-WIN64 GRE (Build 82).

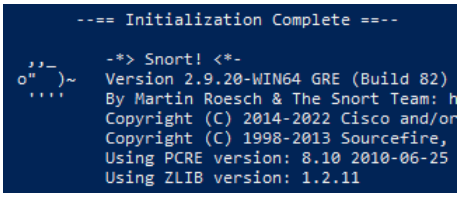


FIGURE 4.3: Snort initialization

All attacks were initiated by the kali-attacker-1 (10.0.2.3) VM to the kali-server-1 VM (10.0.1.3). Each section below will evaluate how snort and the proposed NIPS System performed in each of the attacks, along with any other observations.

4.2.1 SQL Injection

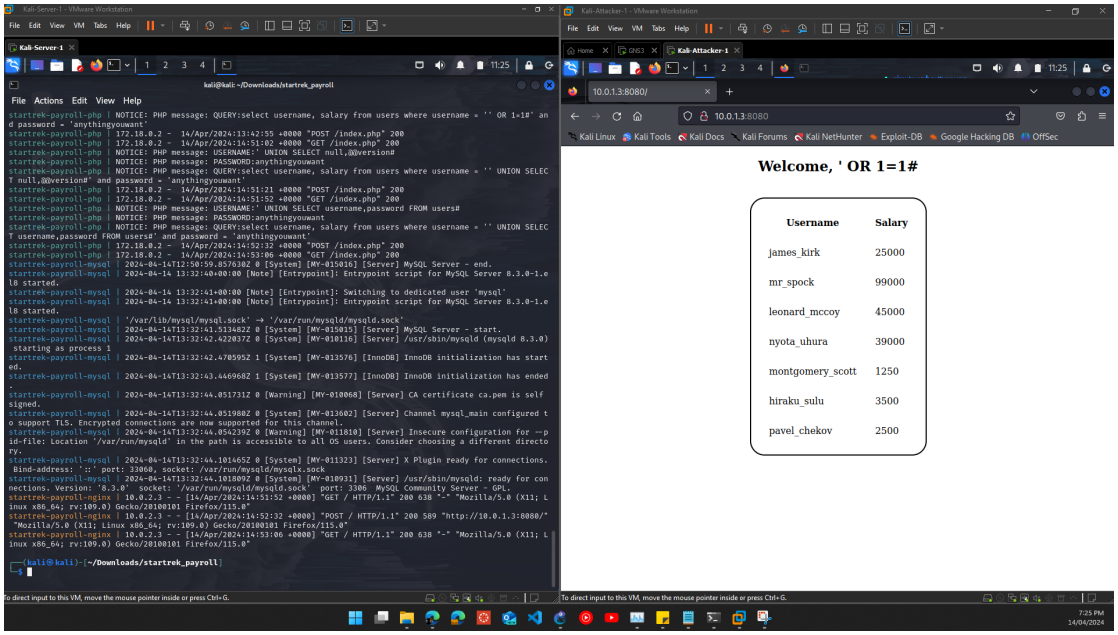


FIGURE 4.4: Kali server hosting thomaslaurenson's vulnerable container and kali attacker injecting a payload

One of the smaller sets of data with only 68446 flows in the NF-ToN-IoT-v2 dataset, both snort and the proposed NIPS were able to detect the attack, as seen in figures 4.5 and 4.6.

```
[**] [1:30040:2] SQL 1 = 1 - possible sql injection attempt [**]
[Classification: Web Application Attack] [Priority: 1]
04/13-22:09:02.172963 10.0.2.3:58830 -> 10.0.1.3:8080
TCP TTL:63 TOS:0x0 ID:62056 IpLen:20 DgmLen:571 DF
***AP*** Seq: 0xF55544A7  Ack: 0xFC779E09  Win: 0x7C80  TcpLen: 32
[Xref => http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/]
```

FIGURE 4.5: Snorts SQL Classification

with snort rightly producing a lof with priority 1 (High) and the L2 classifier classifying the attack as an injection attempt.

```
PS C:\Users\jackd\Downloads\ML-Based-NIPS-Code> & C:/Users/jackd/AppData/Local/Microsoft/WindowsApps/python3.12.exe c:/Users/jackd/Downloads/ML-Based-NIPS-Code/app/app.py
Loaded Models
LogLevel: INFO
Attack Detected, Level-2-Classifier suggests it is: Class 0: Benign. From: Source IP Address:  10.0.2.3, Source Port: 42926, To: Destination IP Address: 10.0.1.3, Source Port: 8080
At time: 2024-04-14T19:25:18.677212
LogLevel: ATTACK
Attack Detected, Level-2-Classifier suggests it is: Class 4: injection. From: Source IP Address:  10.0.2.3, Source Port: 49434, To: Destination IP Address: 10.0.2.3, Source Port: 49434
At time: 2024-04-14T19:26:48.078816
LogLevel: INFO
Attack Detected, Level-2-Classifier suggests it is: Class 0: Benign. From: Source IP Address:  10.0.2.3, Source Port: 49434, To: Destination IP Address: 10.0.1.3, Source Port: 8080
At time: 2024-04-14T19:26:48.375878
```

FIGURE 4.6: NIPS SQL Classification

4.2.2 Denial of Service attack (DoS)

The DOS attack had a similar amount of flows for the model to train with, but surprisingly almost all predictions made by the NIPS and snort were somewhat incorrect, as both detected an anomaly bUt classified the attack incorrectly. Snort declared a level 3

```
[**] [129:11:1] TCP Data with no TCP Flags set [**]
[Classification: Generic Protocol Command Decode] [Priority: 3]
04/15-11:25:43.966562 10.0.2.3:57056 -> 10.0.1.3:8080
TCP TTL:64 TOS:0x0 ID:53190 IpLen:20 DgmLen:160
******** Seq: 0x7884C83B  Ack: 0x5CABE7B3  Win: 0x200  TcpLen: 20
```

FIGURE 4.7: Snorts DoS Classification

(Low) severity, as seen in 4.7, for the attack.The NIPS fared about the same as the L1 classifier correctly classified the flow as an attack, but L2 incorrectly classified it as a benign flow.

4.2.3 Distributed Denial of Service attack (DDoS)

Perhaps due to a similar reason to the failure of L2 to correctly classify a Dos Attack, the L2 Classifier also fails to correctly classify the DDos Attack correctly instead labeling it as benign while L1 correctly classifies it as an attack.

```
[**] [129:20:1] TCP session without 3-way handshake [**]
[Classification: Potentially Bad Traffic] [Priority: 2]
04/15-11:30:04.738120 23.41.40.242:30843 -> 10.0.1.3:8080
TCP TTL:64 TOS:0x0 ID:21688 IpLen:20 DgmLen:160
******** Seq: 0x4F116079  Ack: 0x72941710  Win: 0x200  TcpLen: 20
```

FIGURE 4.8: Snorts DDoS Classification

Similarly, As seen in 4.8, Snort was able to correctly classify the attack but without much certainty as it describes the traffic as "Potentially bad" and gives it a priority of 2 (Medium). Wireshark screenshot of interface during attack can be seen in figure C.4

4.2.4 Reconnaissance/Scanning

This particular classification has the most samples, other than benign flows in the dataset as there are many different types of scanning, both the NIPS and snort did very well in the classification of this attack with snort coming out a little ahead. All 6 commands were listed in chapter 3 were run with breathing time in between each run, snort managed to catch all 6 attempts and although it classified each as a priority of 2, it gave an accurate description of each attack. Wireshark screenshot of interface during attack can be seen in figure C.5

```
[**] [122:1:1] (portscan) TCP Portscan [**]
[Classification: Attempted Information Leak] [Priority: 2]
04/13-22:17:57.243205 10.0.2.3 -> 10.0.1.3
PROTO:255 TTL:64 TOS:0x0 ID:0 IpLen:20 DgmLen:154 DF
```

```
[**] [122:1:1] (portscan) TCP Portscan [**]
[Classification: Attempted Information Leak] [Priority: 2]
04/15-11:35:49.008782 10.0.2.3 -> 10.0.1.3
PROTO:255 TTL:63 TOS:0x0 ID:0 IpLen:20 DgmLen:154 DF
```

```
[**] [122:17:1] (portscan) UDP Portscan [**]
[Classification: Attempted Information Leak] [Priority: 2]
04/15-11:37:04.647277 10.0.2.3 -> 10.0.1.3
PROTO:255 TTL:63 TOS:0xC0 ID:50351 IpLen:20 DgmLen:158
```

FIGURE 4.9: Snorts Scanning Classification

While the L1 and L2 classifiers of the NIPS also classified the attack correctly, the attack generated many flows and some of the flows were incorrect labeled, for instance when performing a TCP scan, some flows were classifies as scanning while others were classified as a DoS attack as seen in the first screenshot of figures below 4.10, which technically could be correct as a TCP scan can be a component of a DoS attack.

Additionally, a UDP scan yielded classification of some flows being an XSS attack, while others were classified as a scanning attack as seen below in figure 4.11, this is more interesting as the 2 attacks don't share much in common. Finally, The remaining scans yielded more accurate classifications. Scans such as the NULL Scan 4.12,XMAS Scan 4.13, etc..

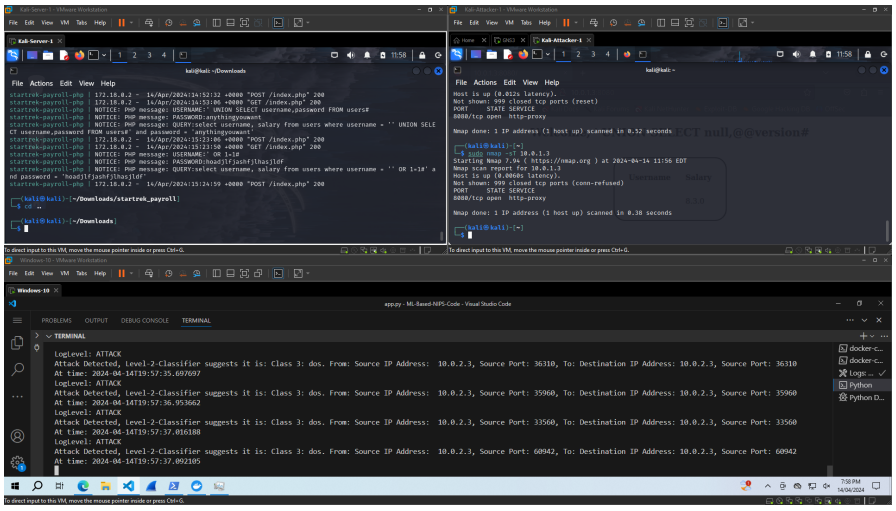


FIGURE 4.10: NIPS Scanning Classification During a TCP Scan

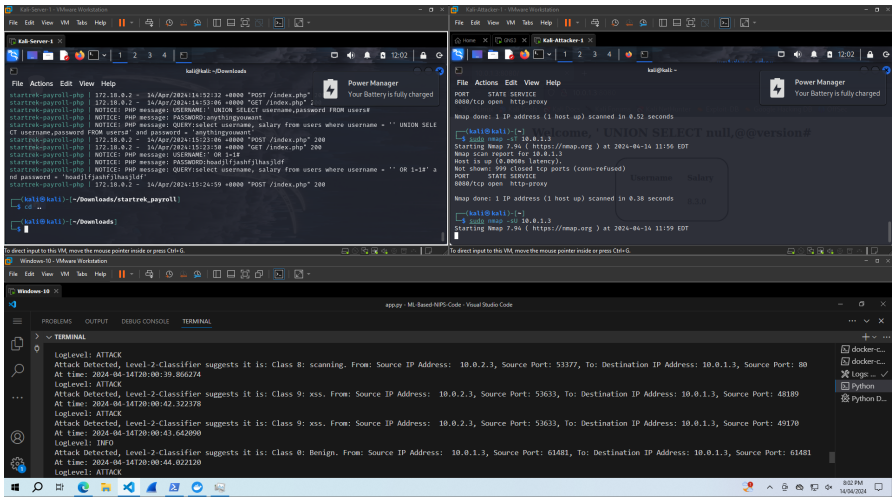


FIGURE 4.11: NIPS Scanning Classification During a UDP Scan

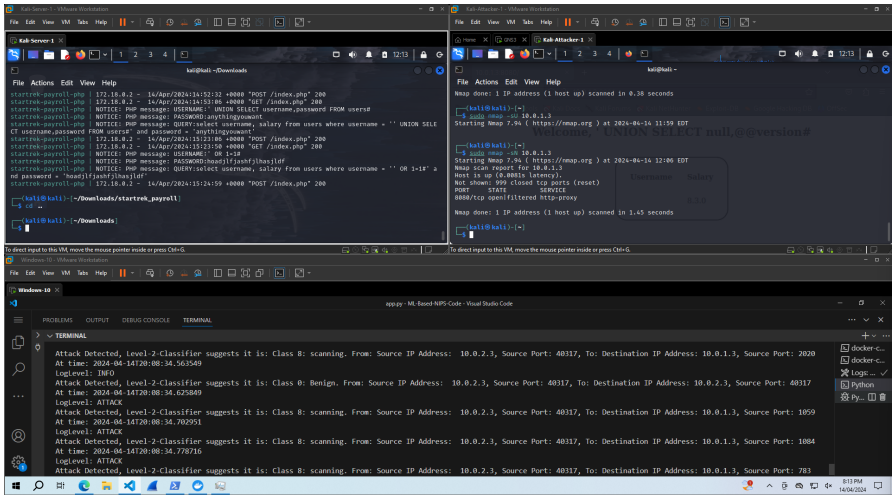


FIGURE 4.12: NIPS Scanning Classification During a NULL Scan

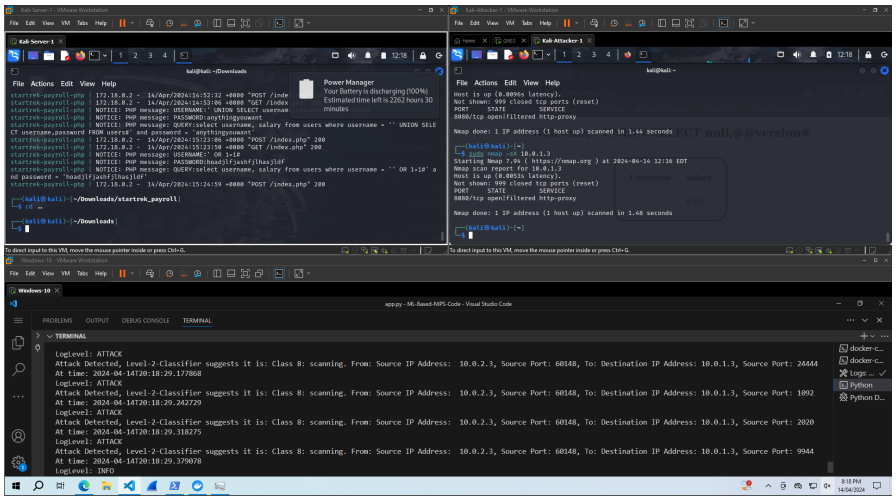


FIGURE 4.13: NIPS Scanning Classification During a XMAS Scan

4.2.5 Benign Flows

In addition to attacks, it is also important to test how the system behaves to benign traffic as most traffic in a network is benign and if either system is too strict, many false positive IP Addresses would be blocked.

Over all Both systems did not have many false positives. Snort only logs an alert is something is detected and while the Level 1 classifier was more accurate when there was an attack, on rare occasions such as running 'apt update && apt upgrade' on a linux machine, it would often classifly flows produced as a DOS attack on 8.8.8.8, the VM's DNS resolver, however, the L2 classifier caught these flows and labeled them as benign.

4.2.6 Network Speeds

As seen below, network speeds did not have a major difference between the systems, However, when the system is under load the results may present more of a difference, as nprobe can be deployed in multiple different computers throughout the network and simply send the flows to the central windows 10 machine, spreading the load of aggregating packets, but that cant be done with snort.

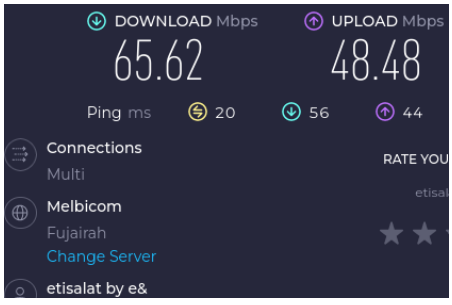


FIGURE 4.14: NIDS Network Speeds

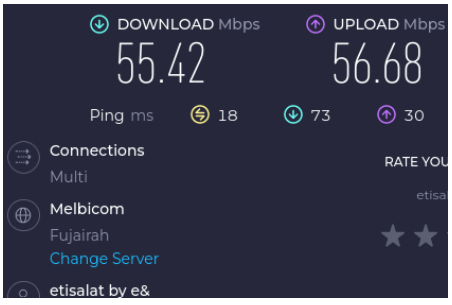


FIGURE 4.15: Snorts Network Speeds

Chapter 5

Conclusion and Future Work

5.1 Conclusions

This dissertation proves the a NetFlow based NIPS is definitely a viable option if a small network experiences a lot of traffic and wants to preserve network speeds, Additionally the system provides a growth path for the network as docker containers typically can be run in systems such as k3s and kubernetes which provides high availability and scalability through the use of load balancers, where if the NIPS docker compose is overwhelmed, more NIPS docker compose stacks can be deployed to account for it and shrink down when not in use. Further more nProbe aggregation responsibilities can be distributed to multiple systems in the network dosen't have to occur on the same system the NIPS is, it can be in separate systems and send data to the ElastiFlow container in the stack.

It has also been determined that the best classifier foe network based machine learning is a decision tree classifier as they have proven to be both fast and accurate. And from the feature subsets tested, the feature set presented by Mikołaj Komisarek Et al. [20] beats the others when it comes to classification accuracy and speed. What make this more interesting is that it manages to do this without using the 5 flow identifiers being (sourceIP, destinationIP,sourcePort,destinationPort and protocol)

Overall the Random Forest and Artificial Neural Network classifiers performed worse that expected, but generally, if more computational resources are given to these models, theoretically, they have a greater predictive power over a more simple classifier such as Decision Trees.

While the proposed NIPS system dosen't match the predictive power of snort and many other NIPS solutions the reasons listed above make the future possibilities of a AI based network analyzer that analyzes and eliminates threats very exciting.

5.2 Future Work

Exact time frames for each attack is definitely different and more research in this field needs to be conducted, as currently nProbe exports a conversation flow every 60 seconds this dosen't aggregate enough data for the classifiers to identify and predict attacks accurately from a single flow. And so from if timings for each attack is known, the

python script can aggregate the flows containing 60 seconds worth of data into a full potential attack conversation, which would provide an emphasis on the indicators of compromise within the a malicious attacks flow

Something to look into in the future is a graphical user interface for the NIPS as this could lower the knowledge needed to deploy the solution, or even offer the solution as a software as a service model, as many companies setup aws or elastic cloud instances to sent logs to as a safeguard. If more computer resources are giver as mentioned earlier, ANNs or ever Random forest or any other power hungry model which surpasses the predictive power of a decision tree could be run off site and block commands could be sent using some form of asymmetric encryption as accepting firewall rules on an unencrypted channel open to the internet is generally a horrible idea.Additionally, elements of the NIPS System need to be strengthened as many passwords are saved in plaintext.

Finally something to look into would be to explore the other datasets proposed by Sarhan et al. [18], as the way this system is implemented, models are not hard coded, any decision tree algorithm can technically be swapped it and it would be interesting to see how the NIPS behaves while training on different datasets as the other datasets are more specialized to different networks, the NF-ToN-IoT-v2 dataset for instance, is focused on smaller mainly IoT based networks, filled with raspberry pies and other IoT devices hence why all client machines in the network were underpowered linux machines to closer emulate a raspberry pi.Additionally all datasets proposed by Sarhan et al. all have the same features and so the Jupyter noteboook used for testing which is in the code repo can be used as is.

Apendix Structure

- Bibliography: This section contains all references and websites used.
- Appendix A: This section contains the Professional, Legal, Ethical and Social considerations regarding this project
- Appendix B: This section showcases the project plan and gantt charts for the duration of this projects
- Appendix C: is the supplementary section containing various bits of information from each chapter

Bibliography

[1] "Ml-based nids datasets," https://staff.itee.uq.edu.au/marius/NIDS_datasets/#RA10.

[2] "Impact of internet usage patterns on telecommunications network traffic," https://utilitiesone.com/impact-of-internet-usage-patterns-on-telecommunications-network-traffic.

[3] AAG-IT, "The latest 2023 cyber crime statistics (updated october 2023)," https://aag-it.com/the-latest-cyber-crime-statistics/, 2023.

[4] w3schools, "Sql injection," https://www.w3schools.com/sql/sql_injection.asp, 2023.

[5] "Sql injection attack: Real life attacks and code examples," https://brightsec.com/blog/sql-injection-attack/.

[6] "Cve-2024-3400 pan-os: Os command injection vulnerability in globalprotect," https://security.paloaltonetworks.com/CVE-2024-3400, (Accessed on 04/15/2024).

[7] T. N. Stack, "The drawbacks of a soar - the new stack," https://thenewstack.io/the-drawbacks-of-a-soar/, 2023.

[8] "Basic facts about the cisco router brand family," https://www.lifewire.com/introduction-to-cisco-routers-817473#:~:text=Cisco%20Systems%20produces%20a%20wide,for%20quality%20and%20high%20performance.

[9] W. Seo and W. Pak, "Real-time network intrusion prevention system based on hybrid machine learning," *IEEE Access*, vol. 9, pp. 46 386–46 397, 2021.

[10] gartner, "Trend micro tippingpoint reviews, ratings & features 2023 — gartner peer insights," https://www.gartner.com/reviews/market/intrusion-prevention-systems/vendor/trend-micro/product/trend-micro-tippingpoint, 2023.

[11] J. Andress, "Chapter 10 - network security," in *The Basics of Information Security (Second Edition)*, second edition ed., J. Andress, Ed. Boston: Syngress, 2014, pp. 151–169. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128007440000105

[12] corelight, "What is signature-based detection?" https://corelight.com/resources/glossary/signature-based-detection, 2023.

[13] "What is netflow? — ibm," https://www.ibm.com/topics/netflow.

[14] "Flexible netflow configuration guide, cisco ios release 15m&t - configuring data export for flexible netflow with flow exporters [cisco ios 15.4m&t] - cisco," https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/fnetflow/configuration/15-mt/fnf-15-mt-book/cfg-de-fnflow-exprts.html.

[15] "Flow-based monitoring – my site," https://www.ntop.org/solutions/flow-based-monitoring/.

[16] "Netflow analyzers and netflow tools — kentik," https://www.kentik.com/kentipedia/netflow-analyzers-and-netflow-tools/#:~:text=A%20NetFlow%20analyzer%20is%20a,of%20network%20traffic%20flow%20data.

[17] "What is netflow? — basics of netflow – manageengine netflow analyzer," https://www.manageengine.com/products/netflow/what-is-netflow.html.

[18] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann, "Netflow datasets for machine learning-based network intrusion detection systems," vol. 371 LNICST. Springer Science and Business Media Deutschland GmbH, 2021, pp. 117–135.

[19] S. Fraihat, S. Makhadmeh, M. Awad, M. A. Al-Betar, and A. Al-Redhaei, "Intrusion detection system for large-scale iot netflow networks using machine learning with modified arithmetic optimization algorithm," 7 2023.

[20] M. Komisarek, M. Pawlicki, R. Kozik, W. Hołubowicz, and M. Choraś, "How to effectively collect and process network data for intrusion detection?" *Entropy*, vol. 23, 11 2021.

[21] M. Awad, S. Fraihat, K. Salameh, and A. A. Redhaei, "Examining the suitability of netflow features in detecting iot network intrusions," *Sensors*, vol. 22, 8 2022.

[22] N. Thapa, Z. Liu, D. B. Kc, B. Gokaraju, and K. Roy, "Comparison of machine learning and deep learning models for network intrusion detection systems," *Future Internet*, vol. 12, pp. 1–16, 10 2020.

[23] I. Shivhare, J. Purohit, V. Jogani, S. Attari, and M. Chandane, "Intrusion detection: A deep learning approach."

[24] "Lesson 1.4 data shuffling - deci," https://deci.ai/course/data-shuffling/#:~:text=Data%20shuffling%20is%20an%20integral,which%20can%20lead%20to%20overfitting., (Accessed on 03/30/2024).

[25] "What is overfitting in deep learning [+10 ways to avoid it]," https://www.v7labs.com/blog/overfitting#:~:text=Cross%2Dvalidation%20is%20a%20robust,fold%20as%20the%20test%20set., (Accessed on 04/02/2024).

[26] "iptables - wikipedia," https://en.wikipedia.org/wiki/Iptables#:~:text=iptables%20is%20a%20user%2Dspace,to%20treat%20network%20traffic%20packets.

[27] "Jacdm/ml-based-nips," https://github.com/JacDM/ML-Based-NIPS, (Accessed on 04/15/2024).

[28] "Gns3 — the software that empowers network professionals," https://www.gns3.com/.

[29] "Vmware workstation player — vmware — mena," https://www.vmware.com/mena/products/workstation-player.html.

[30] thomaslaurenson, "thomaslaurenson/startrek_payroll: A simple sql injection vulnerable web application powered by docker," https://github.com/thomaslaurenson/startrek_payroll.

[31] "Nmap: the network mapper - free security scanner," https://nmap.org/, (Accessed on 04/14/2024).

Appendix A

PLES

A.1 Professional and Legal Considerations

All research papers, figures, tables, articles, and websites have been properly referenced and can be found in the bibliography. code or tools used by other developers will be referenced and acknowledged accordingly. All papers, code and datasets that are used in this project are open source. Source code produced will be open source published under the MIT licence.

In addition all software used such as GNS3, VMware, etc. will be licensed or free to use.

A.2 Ethical and Social Considerations

The system being developed by this project does not store any identifiable personal or private information and all datasets used for training, testing and validation are all open source, regularly used in similar projects and can be found at [1].

Appendix B

Project Management

B.1 Project Plan

All dates and times were the outcome of the formula:

$$Average\ Time = \frac{Minimum\ Time + 4 * Average\ Time + Minimum\ Time}{6}$$

FIGURE B.1: Deliverable 1 Gantt chart

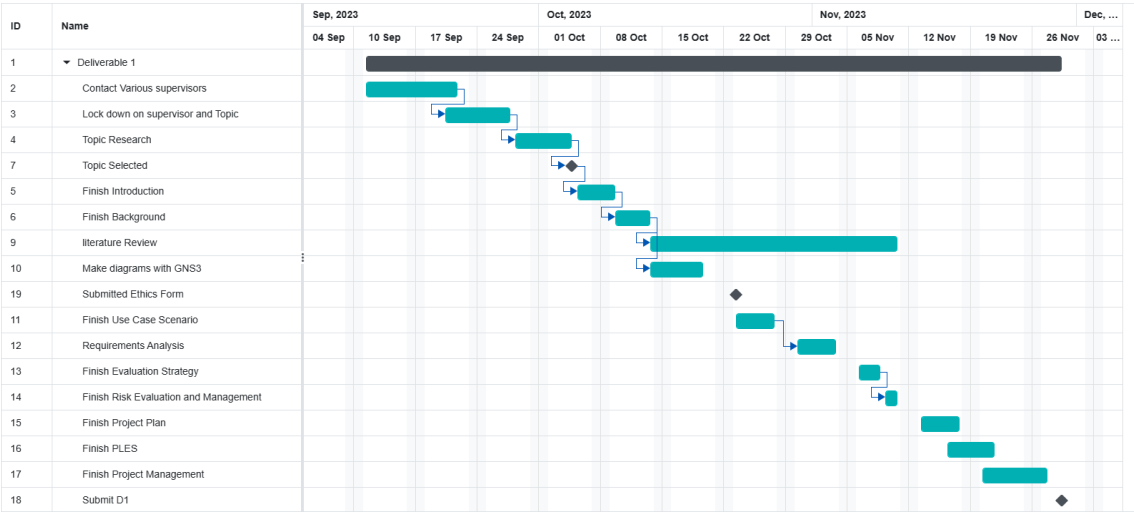
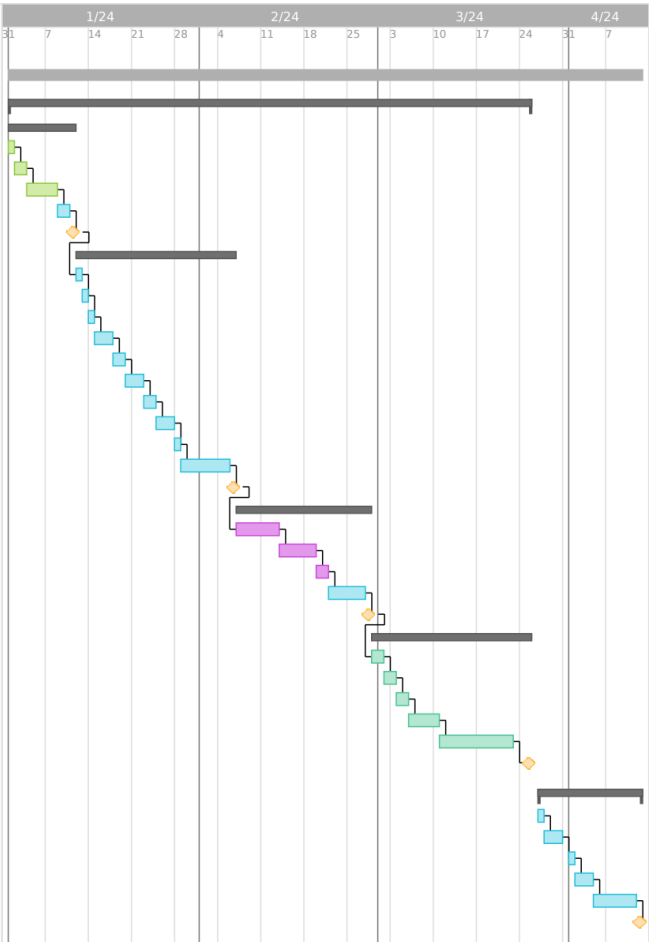


FIGURE B.2: Deliverable 2 Gantt chart

Dissertation	start	end
Deliverable 2	**01/01/24**	**25/03/24**
Research + Discovery	**01/01/24**	**11/01/24**
Look into ways of implementing Re...	01/01	01/01
Install all necessary software	02/01	03/01
Experiment with different packet b...	04/01	08/01
Experiment with docker	09/01	10/01
Meeting with supervisor to discuss ...	11/01	11/01
Development Phase	**12/01/24**	**06/02/24**
Setup GNS3 environment	12/01	12/01
Setup docker	13/01	13/01
Setup VMWare	14/01	14/01
Implement rule system	15/01	17/01
Test different feature sets	18/01	19/01
Test different classifiers for L1	20/01	22/01
Test different classifiers for L2	23/01	24/01
Validate model using test data	25/01	27/01
Record results	28/01	28/01
Design Revisions (if needed)	29/01	05/02
Meeting with supervisor to discuss ...	06/02	06/02
Integration Phase	**07/02/24**	**28/02/24**
Integrate rule system	07/02	13/02
Integrate kibana and NetFlow pack...	14/02	19/02
Integrate python ML model	20/02	21/02
Create docker compose file	22/02	27/02
Review with supervisor	28/02	28/02
Testing + Revision Phase	**29/02/24**	**25/03/24**
Run docker compose file on linux s...	29/02	01/03
Test model giving rules to block pa...	02/03	03/03
Simulate attack that system has tra...	04/03	05/03
Simulate attack model has not trai...	06/03	10/03
Revisions	11/03	22/03
Review with supervisor	25/03	25/03
Deliverable 2 Report	**27/03/24**	**12/04/24**
Receive feedback on D1	27/03	27/03
implement changes based on feedba...	28/03	30/03
Proceed with writeup on results and f...	01/04	01/04
Write conclusions	02/04	04/04
Final Polishing	05/04	11/04
submit D2	12/04	12/04



B.2 Risk Analysis

ID	Risk	Type	Impact	Probability	Description	Mitigation
R1	Health Issues	People	Serious	Low	I become physically or mentally ill.	***Avoidance***: take proper care of self and check-in with supervisor about concerns.
R2	Loss of data	Technology	Catastrophic	Low	my laptop breaks and the data can't be recovered.	***Minimization***: all major changes to doc should be uploaded to GitHub, resilio server and OneDrive.
R3	Loss of data sets	Requirement	Serious	Low	The online datasets are taken offline for whatever reason.	***Avoidance***: download and make backups of data sets that are intended to be use.
R4	Insufficient computing power	Tools	Serious	Medium	The current tools used are not sufficient to test the NIPS in either a virtual or physical scenario	***Avoidance***: Virtual testing should be carried out in the labs at Heriot-watt.
R5	Missing Supervisor	People	Catastrophic	Low	supervisor stops responding or is missing	***Avoidance***: Be informed about the supervisors availability
R6	Deadlines Missed accidentally	Estimation	Serious	Medium	In case I forget the submission date or last-minute slow uploads exceeding the time limit.	***Avoidance***: make multiple uploads beforehand even if not fully complete and set reminders
R7	Failure to finish implementation before deadline due to scope	Estimation	Serious	low	The implementation of the NIPS is not complete in time, be it the model, application, or dashboards.	***Avoidance***: start early, set milestones, follow Gant chart and cut features if absolutely necessary to provide a MVP

Appendix C

Supplementary

C.1 Chapter 1

Table C.1: NetFlow v2 Features List from [1]

ID	Feature	Description
1	IPV4_SRC_ADDR	IPv4 source address
2	IPV4_DST_ADDR	IPv4 destination address
3	L4_SRC_PORT	IPv4 source port number
4	L4_DST_PORT	IPv4 destination port number
5	PROTOCOL	IP protocol identifier byte
6	L7_PROTO	Layer 7 protocol (numeric)
7	IN_BYTES	Incoming number of bytes
8	OUT_BYTES	Outgoing number of bytes
9	IN_PKTS	Incoming number of packets
10	OUT_PKTS	Outgoing number of packets
11	FLOW_DURATION_MILLISECONDS	Flow duration in milliseconds
12	TCP_FLAGS	Cumulative of all TCP flags
13	CLIENT_TCP_FLAGS	Cumulative of all client TCP flags
14	SERVER_TCP_FLAGS	Cumulative of all server TCP flags
15	DURATION_IN	Client to Server stream duration (msec)
16	DURATION_OUT	Client to Server stream duration (msec)
17	MIN_TTL	Min flow TTL
18	MAX_TTL	Max flow TTL
19	LONGEST_FLOW_PKT	Longest packet (bytes) of the flow
20	SHORTEST_FLOW_PKT	Shortest packet (bytes) of the flow
21	MIN_IP_PKT_LEN	Len of the smallest flow IP packet observed
22	MAX_IP_PKT_LEN	Len of the largest flow IP packet observed
23	SRC_TO_DST_SECOND_BYTES	Src to dst Bytes/sec
24	DST_TO_SRC_SECOND_BYTES	Dst to src Bytes/sec
25	RETRANSMITTED_IN_BYTES	Number of retransmitted TCP flow bytes (src-dst)
26	RETRANSMITTED_IN_PKTS	Number of retransmitted TCP flow packets (src-dst)

27	RETRANSMITTED_OUT_BYTES	Number of retransmitted TCP flow bytes (dst-src)
28	RETRANSMITTED_OUT_PKTS	Number of retransmitted TCP flow packets (dst-src)
29	SRC_TO_DST_AVG_THROUGHPUT	Src to dst average thpt (bps)
30	DST_TO_SRC_AVG_THROUGHPUT	Dst to src average thpt (bps)
31	NUM_PKTS_UP_TO_128_BYTES	Packets whose IP size = 128
32	NUM_PKTS_128_TO_256_BYTES	Packets whose IP size 128 and = 256
33	NUM_PKTS_256_TO_512_BYTES	Packets whose IP size 256 and = 512
34	NUM_PKTS_512_TO_1024_BYTES	Packets whose IP size 512 and = 1024
35	NUM_PKTS_1024_TO_1514_BYTES	Packets whose IP size 1024 and = 1514
36	TCP_WIN_MAX_IN	Max TCP Window (src-dst)
37	TCP_WIN_MAX_OUT	Max TCP Window (dst-src)
38	ICMP_TYPE	ICMP Type * 256 + ICMP code
39	ICMP_IPV4_TYPE	ICMP Type
40	DNS_QUERY_ID	DNS query transaction Id
41	DNS_QUERY_TYPE	DNS query type (e.g. 1=A, 2=NS..)
42	DNS_TTL_ANSWER	TTL of the first A record (if any)
43	FTP_COMMAND_RET_CODE	FTP client command return code

ElastiFlow Parsed Attribute Names	**Dataset Attribute Names**
event.duration/10^9	FLOW_DURATION_MILLISECONDS
flow.in.tcp.window.size_max	TCP_WIN_MAX_IN
ntop.out.c2s.duration	DURATION_OUT
ip.ttl_max	MAX_TTL
app.id	L7_PROTO
flow.client.throughput_avg	SRC_TO_DST_AVG_THROUGHPUT
flow.packet_size_min	SHORTEST_FLOW_PKT
ip.packet.size_min	MIN_IP_PKT_LEN
flow.out.tcp.window.size_max	TCP_WIN_MAX_OUT
flow.out.bytes	OUT_BYTES
client.domain	IPV4_SRC_ADDR
destination.domain	IPV4_DST_ADDR
exporter ip	host.ip

Table C.2: Elastiflow to Dataset name mappings

C.2 Chapter 3

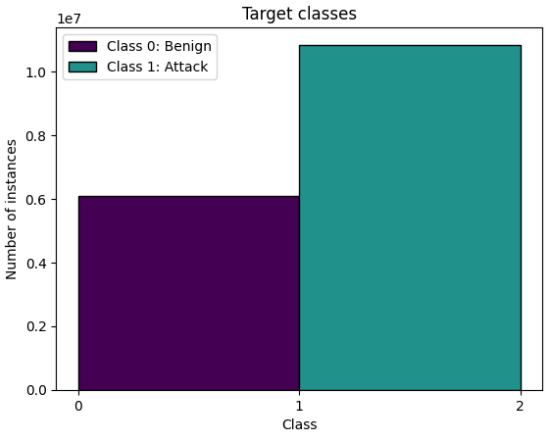


FIGURE C.1: Before Under-Sampling

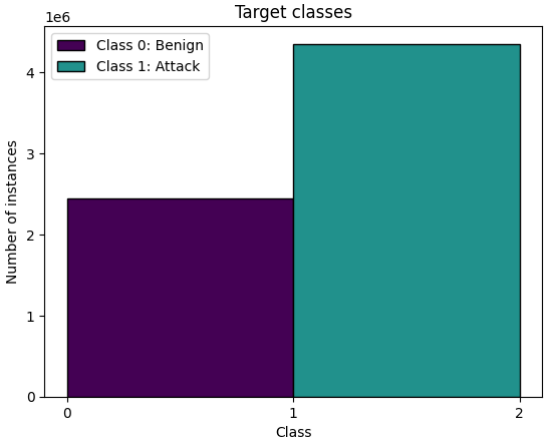


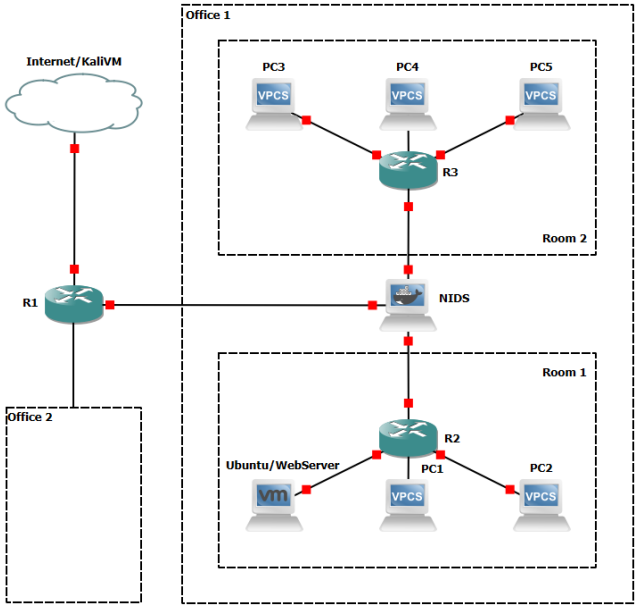
FIGURE C.2: After Under-Sampling

P stands for No. of Processors, CP is the No. of cores per processor.

System/Node	**Cores Allocated**	**RAM Allocated**	**Additional Configurations**
Windows 10-NIPS-System	8 (4P/2CP)	6144 MB	3 e1000 NICs Virtualization Enabled
Windows-10-Snort	8 (4P/2CP)	6144 MB	3 e1000 NICs Virtualization Enabled
Kali-Server-1	2 (2P/1CP)	2048 MB	1 e1000 NIC
Kali-Attacker-1	2 (2P/1CP)	2048 MB	1 e1000 NICs
GNS3-VM	1 (1P/1CP)	2048 MB	2 e1000 NICs Virtualization Enabled
Ubuntu-Router	-	1.918 GiB	2 e1000 NICs
GNS3 VPCs (2)	-	-	1 e1000 NIC

TABLE C.3: GNS3 VM Nodes configurations

C.2.1 Use Case Scenario



In the scenario painted by figure C.3, we see 2 office buildings and focus on office 1 which has 2 rooms with each rooms having 3 computers and 1 NetFlow enabled router. Each room is subnetted and the only way for computers in room 1 to talk to computes in room 2 is via the gateway provided by the NIPS which is running the ML based intrusion classification algorithm that this project aims to implement .

FIGURE C.3: Proposed use case for NIPS

- Scenario 1: Let's say that PC5 launches a DoS attack on the ubuntu/Web server in scenario, the packets sent will travel through the NIPS system, as R3 sends a flow of the conversation between PC5 and the webserver the ML classifier of the NIPS will detect the flow as an attack and drop all incoming packets from PC5 by adding a rule to the IP Tables.

- Scenario 2: An attack from the internet DDoS is attacking the web server, the NIPS will receive the exported flow from R1 and the ML classifier will detect the flow as an attack and block the IP by adding a new rule to the firewall application which will update the NIPS IP Tables.

C.2.2 User Requirements

Legend can be found in appendix: C.6

FUNCTIONAL REQUIREMENTS			
ID	***Description***	***Priority***	***Linked Objective***
FR1	*The classification model must be pre-trained to minimize setup time.*	**M**	**3**
FR1.1	*The user schould be able to load their own trained models.*	**M**	**3**
FR2	*The NIPS must be allowed to receive NetFlow flows from multiple Ethernet Ports*	**M**	**5**
FR3	*The NIPS's ML classifier must analyse the flow and classify the flow as either benign or an attack*	**M**	**2**
FR4	*If the flow is classified as an attack the NIPS must block all packets flowing through the NIPS NIC's.*	**M**	**3**
FR5	*The NIPS should produce a dashboard for SOC Analysts.*	**S**	**4**
FR6	*The classification algorithm should not be a major bottleneck and allow for a network throughput of 100Gbps.*	**S**	**2**
FR7	*The system should be able to intercept flows coming from routers and switches (NetFlow Exporters)*	**M**	**3**
FR8	*The NIPS should not treat one NIC as an input and the other as an output, both NIC's should receive flows and packets to be analysed.*	**S**	**3**
FR9	*The whole NIPS should be able to be deployed by running a docker container on a Linux server.*	**S**	**3**
FR10	*If the attack is not dependent on packets flowing through the NIPS, then an alert should be generated to the SIEM tool (Dashboard).*	**S**	**4**
FR11	*The system should be able to view provide a list of blocked IP's*	**S**	**-**
FR12	*The user should be able to add blocked IP's manually*	**C**	**-**
FR13	*The user should be able to REMOVE blocked IP's manually*	**C**	**-**
NFR14	*The System should be able to Configure any additional NIPS settings via the command-line*	**C**	**3**

NON-FUNCTIONAL REQUIREMENTS			
ID	***Description***	***Priority***	***Linked Objective***
NFR1	*The system could present the user with a separate dashboard to configure the NIPS settings called settings dashboard.*	**W**	**4**
NFR2	*The alerts could trigger other systems such as Telegram notifications.*	**C**	**4**
NFR3	*In the case that multiple NIPS are deployed in the network, when a rule is created blocking packets by an NIPS the other NIPS in the network could also add the rule*	**C**	**-**
NFR4	*The NIPS's classification model could be multi-threaded*	**C**	**2**
NFR5	*The alerts generated should draw the SOC Analysts attention by either flashing the screen or making a sound.*	**W**	**4**
NFR6	*The UI of the settings dashboard could be customizable.*	**W**	**4**

Risk Probability	Abbreviation
Must	M
Should	S
Could	C
Wont	W

Table C.6: MoSCoW Abbreviations

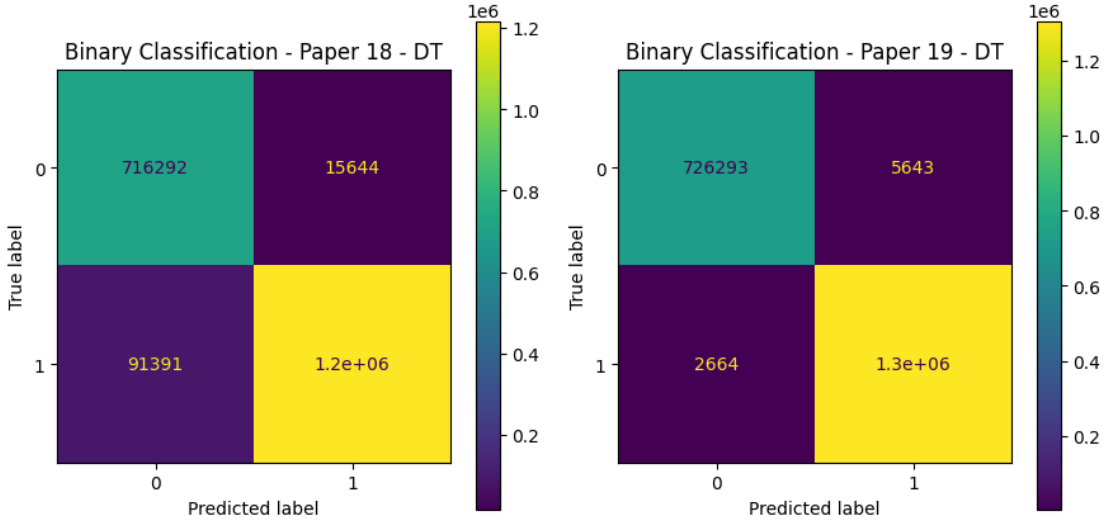
C.3 Chapter 4

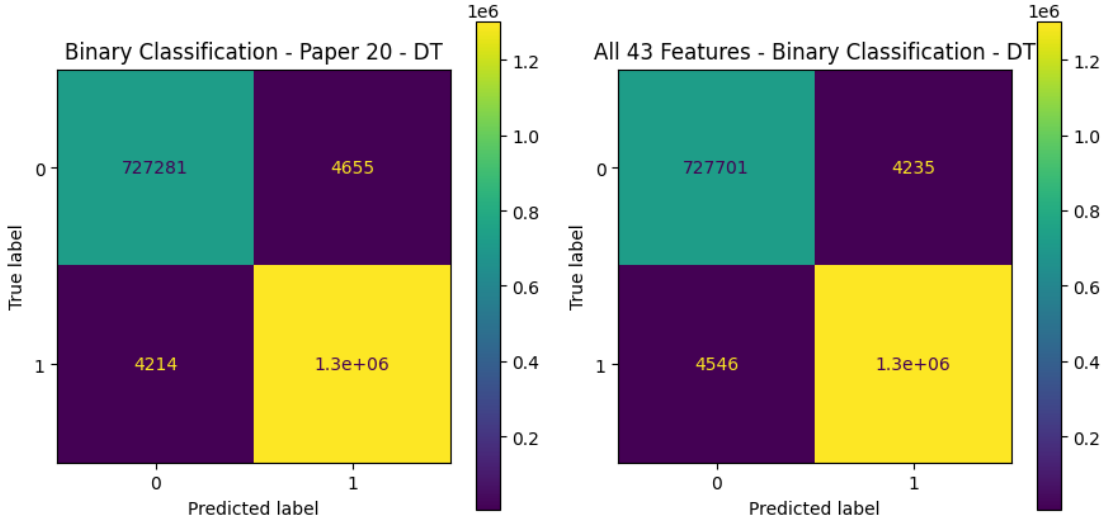
It is important to note that development of the system, testing, model evaluation, network simulation, etc. . . all happened on a single Microsoft Surface Laptop 4 with a Quad Core 11th Intel® Core™ i7-1185G7 processor, 16GB of DDR4 Memory, Intel Iris Xe(96eu) integrated graphics and a Wi-Fi 6: 802.11a based NIC running Windows 11. It is important to note that the laptop was set to Best performance and always plugged it and cooled with an external fan. These factors should be taken into account as they directly influence the results in Chapter 4.

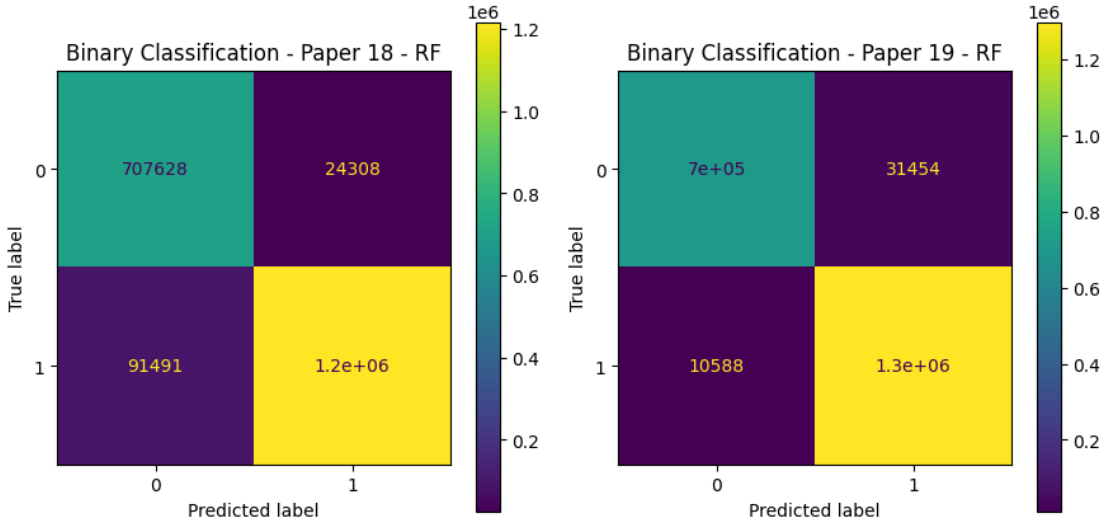
Field	Binary Tally	Multi-Classification Tally	Total Tally
TCP_WIN_MAX_IN	3.0	3.0	6
FLOW_DURATION_MILLISECONDS	2.0	2.0	4
MAX_IP_PKT_LEN	2.0	2.0	4
DURATION_OUT	2.0	1.0	3
MAX_TTL	2.0	1.0	3
L7_PROTO	2.0	2.0	4
SRC_TO_DST_AVG_THROUGHPUT	2.0	2.0	4
SHORTEST_FLOW_PKT	2.0	3.0	5
NUM_PKTS_1024_TO_1514_BYTES	1.0	1.0	2
RETRANSMITTED_OUT_PKTS	1.0	1.0	2
IN_PKTS	1.0	1.0	2
PROTOCOL	1.0	1.0	2
TCP_FLAGS	1.0	1.0	2
SERVER_TCP_FLAGS	1.0	1.0	2
DST_TO_SRC_AVG_THROUGHPUT	1.0	1.0	2
L4_SRC_PORT	1.0	1.0	2
IN_BYTES	1.0	1.0	2
IPV4_DST_ADDR	1.0	1.0	2
IPV4_SRC_ADDR	1.0	NaN	1
L4_DST_PORT	1.0	1.0	2
LONGEST_FLOW_PKT	1.0	1.0	2
SRC_TO_DST_SECOND_BYTES	1.0	1.0	2
OUT_BYTES	1.0	3.0	4
TCP_WIN_MAX_OUT	1.0	3.0	4
MIN_IP_PKT_LEN	1.0	2.0	3
DNS_QUERY_ID	1.0	1.0	2
DST_TO_SRC_SECOND_BYTES	NaN	1.0	1
CLIENT_TCP_FLAGS	NaN	2.0	2
ICMP_TYPE	NaN	1.0	1

Table C.7: Feature Importance based on 5 subsets from previous works.

C.3.1 Confusion Matrices for Binary Classifications





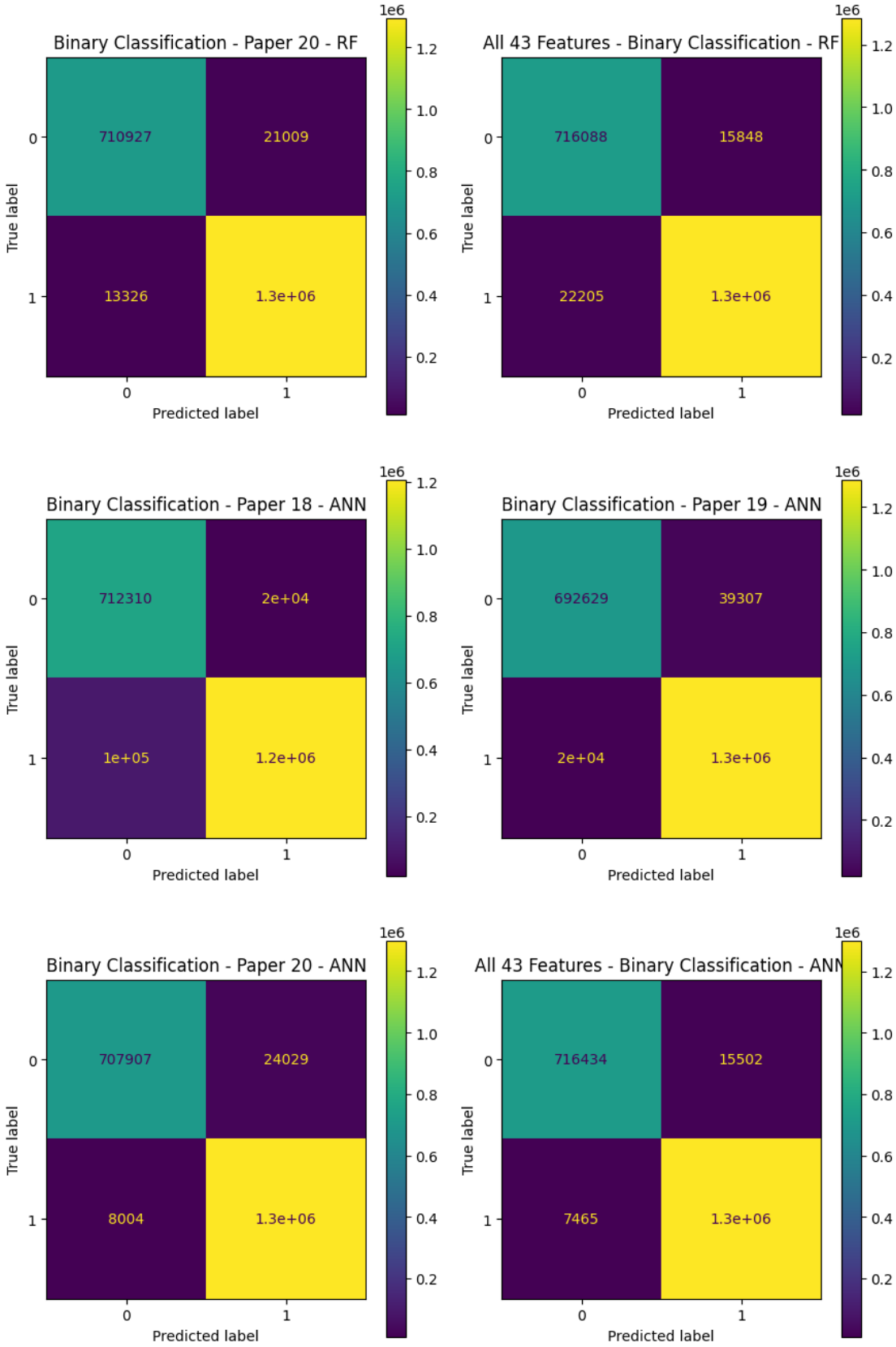


Models	DT				RF				ANN			
Subsets	All 43 Features	Paper 18	Paper 19	Paper 20	All 43 Features	Paper 18	Paper 19	Paper 20	All 43 Features	Paper 18	Paper 19	Paper 20
Training Time (s)	95.36	17.10	11.63	34.11	759.37	414.11	437.56	655.92	3022.27 (50.37m)	2881.44 (48.02m)	2898.68 (48.31m)	2899.22 (48.32m)
Prediction Time (s)	0.80	0.29	0.28	0.36	28.11	25.37	25.69	25.10	60.28	50.34	50.39	52.61
Precision (Class 0)	0.99	0.89	1.00	0.99	0.97	0.89	0.99	0.98	0.99	0.88	0.97	0.99
Precision (Class 1)	1.00	0.99	1.00	1.00	0.99	0.98	0.98	0.98	0.99	0.98	0.97	0.98
Precision (Weighted Avg)	1.00	0.95	1.00	1.00	0.98	0.94	0.98	0.98	0.99	0.94	0.97	0.98
Precision (Macro Avg)	1.00	0.94	1.00	1.00	0.98	0.93	0.98	0.98	0.99	0.93	0.97	0.98
Recall (Class 0)	0.99	0.98	0.99	0.99	0.98	0.97	0.96	0.97	0.98	0.97	0.95	0.97
Recall (Class 1)	1.00	0.93	1.00	1.00	0.98	0.93	0.99	0.99	0.99	0.92	0.98	0.99
Recall (Weighted Avg)	1.00	0.95	1.00	1.00	0.98	0.94	0.98	0.98	0.99	0.94	0.97	0.98
Recall (Macro Avg)	1.00	0.95	1.00	1.00	0.98	0.95	0.97	0.98	0.99	0.95	0.97	0.98
F1-Score (Class 0)	0.99	0.93	0.99	0.99	0.97	0.92	0.97	0.98	0.98	0.92	0.96	0.98
F1-Score (Class 1)	1.00	0.96	1.00	1.00	0.99	0.95	0.98	0.99	0.99	0.95	0.98	0.98
F1-Score (Macro Avg)	1.00	0.94	1.00	1.00	0.98	0.94	0.98	0.98	0.99	0.94	0.97	0.98
Accuracy	1.00	0.95	1.00	1.00	0.98	0.94	0.98	0.98	0.99	0.94	0.97	0.98

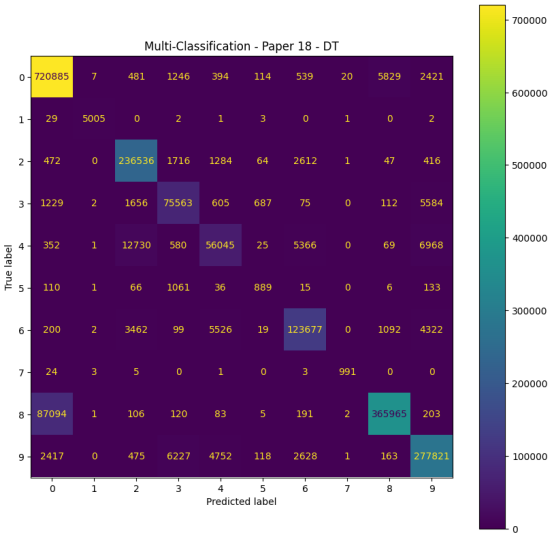
Table C.8: All Binary Prediction Results

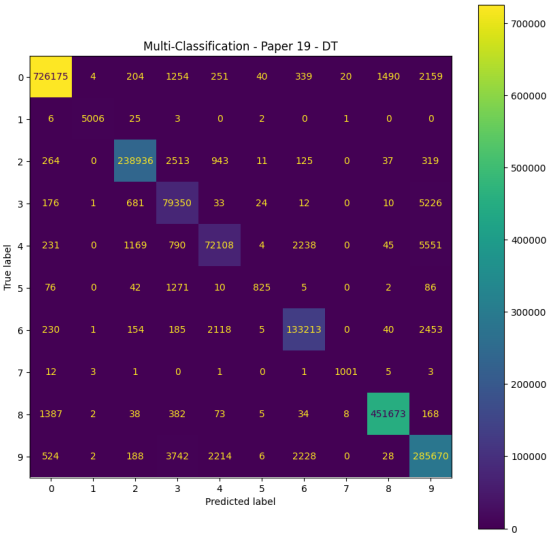
Model	DT				RF				ANN			
Approach	All Features	Paper 18	Paper 19	Paper 20	All Features	Paper 18	Paper 19	Paper 20	All Features	Paper 18	Paper 19	Paper 20
Training Time (s)	61.91	14.84	9.00	34.76	793.38	428.56	452.91	774.50	3696.60 (61.61m)	3202.80 (53.38m)	3867.00 (64.45m)	2977.20 (49.62m)
Prediction Time (s)	0.98	0.42	0.40	0.62	47.31	42.26	44.19	41.99	66.21	64.32	89.82	86.74
Precision (Macro)	0.92	0.88	0.96	0.94	0.94	0.91	0.94	0.94	0.93	0.84	0.87	0.90
Recall (Macro)	0.92	0.85	0.90	0.94	0.81	0.74	0.79	0.81	0.85	0.78	0.80	0.85
F1-Score (Macro)	0.92	0.86	0.92	0.94	0.85	0.79	0.83	0.84	0.87	0.79	0.81	0.86
Accuracy	0.98	0.91	0.98	0.98	0.93	0.89	0.94	0.94	0.96	0.89	0.93	0.95
F1-Score (Class 0)	0.99	0.93	0.99	0.99	0.97	0.93	0.97	0.98	0.98	0.92	0.97	0.98
F1-Score (Class 1)	1.00	0.99	1.00	1.00	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
F1-Score (Class 2)	0.99	0.95	0.99	0.99	0.93	0.92	0.94	0.93	0.97	0.93	0.91	0.96
F1-Score (Class 3)	0.89	0.88	0.91	0.90	0.91	0.85	0.91	0.91	0.90	0.88	0.89	0.89
F1-Score (Class 4)	0.89	0.74	0.90	0.90	0.66	0.48	0.64	0.60	0.76	0.46	0.58	0.70
F1-Score (Class 5)	0.60	0.42	0.51	0.74	0.44	0.42	0.41	0.43	0.41	0.33	0.16	0.38
F1-Score (Class 6)	0.96	0.90	0.96	0.96	0.89	0.77	0.86	0.88	0.90	0.82	0.83	0.89
F1-Score (Class 7)	0.98	0.97	0.97	0.98	0.80	0.74	0.69	0.81	0.90	0.80	0.90	0.93
F1-Score (Class 8)	1.00	0.88	1.00	1.00	0.95	0.88	0.97	0.97	0.98	0.88	0.97	0.98
F1-Score (Class 9)	0.94	0.94	0.96	0.95	0.93	0.90	0.92	0.93	0.93	0.92	0.91	0.93

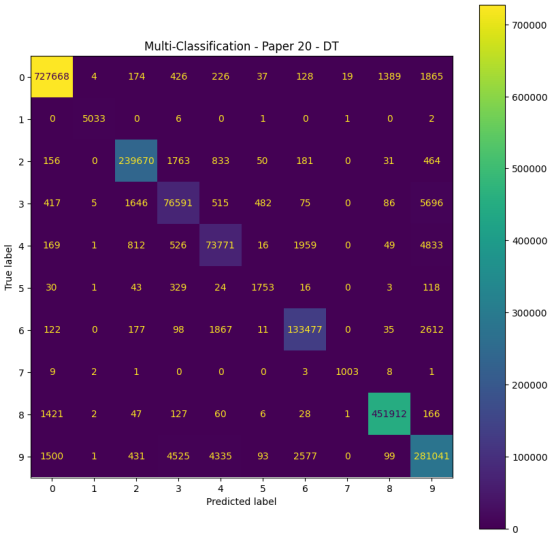
Table C.9: Multi-Classification Prediction Results

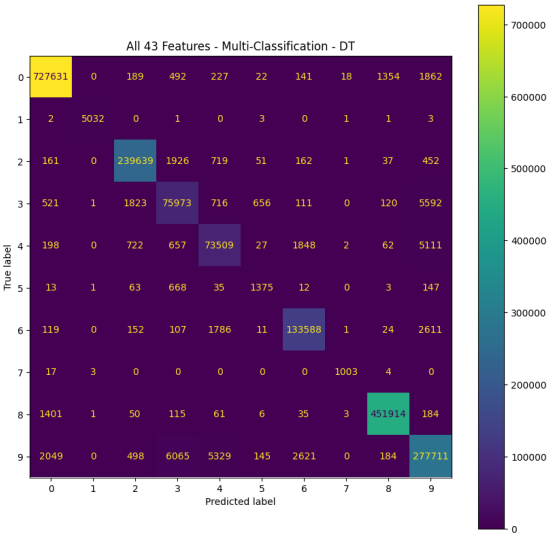

Binary Classification - Paper 20 - RF
710927
21009
13326
1.3e+06
All 43 Features - Binary Classification - RF
716088
15848
22205
1.3e+06
Binary Classification - Paper 18 - ANN
712310
2e+04
1e+05
1.2e+06
Binary Classification - Paper 19 - ANN
692629
39307
2e+04
1.3e+06
Binary Classification - Paper 20 - ANN
707907
24029
8004
1.3e+06
All 43 Features - Binary Classification - ANN
716434
15502
7465
1.3e+06
True label
Predicted label
1e6

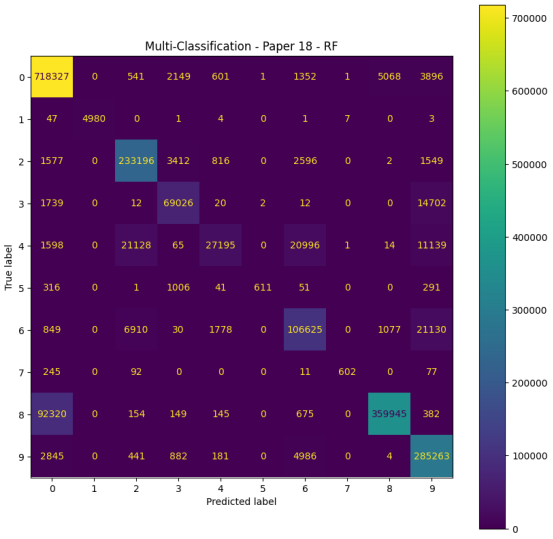
C.3.2 Confusion Matrices for Multi Classifications

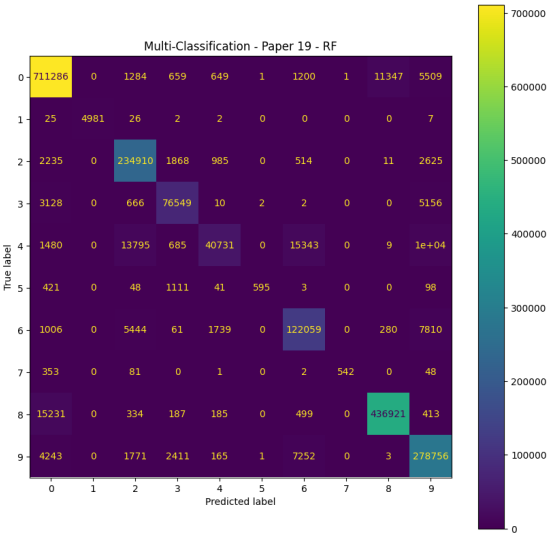


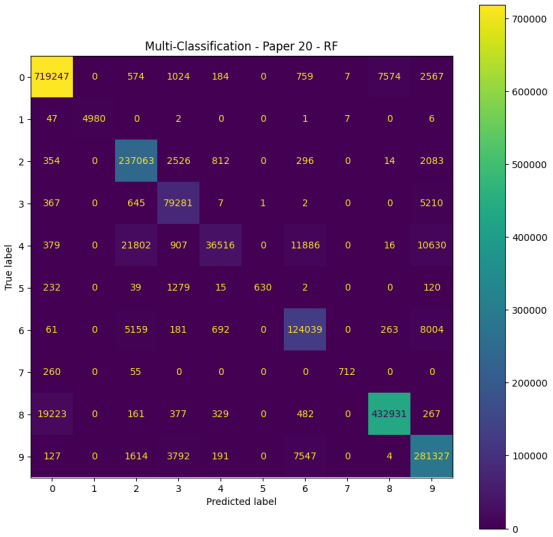


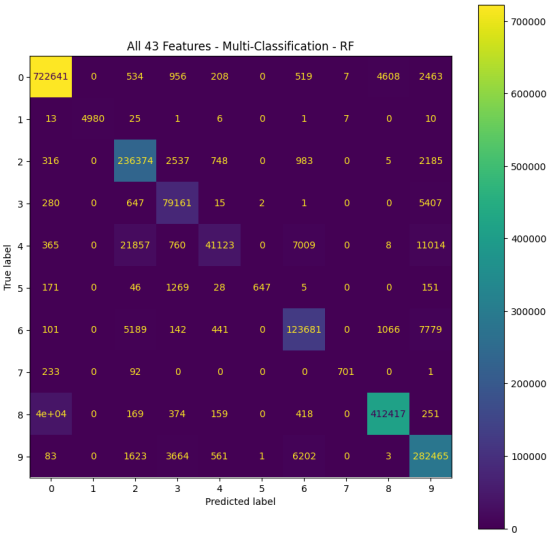


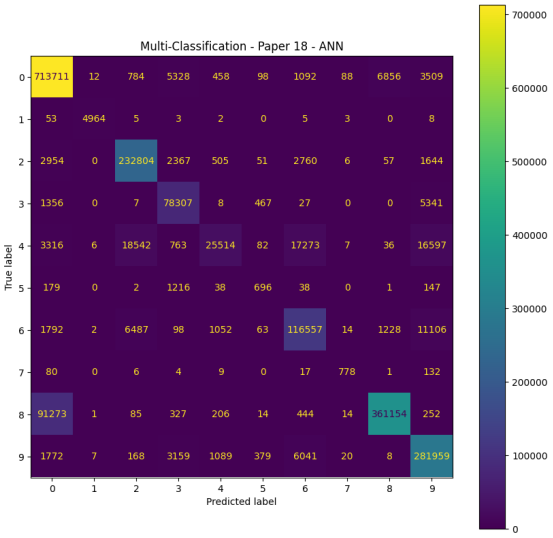


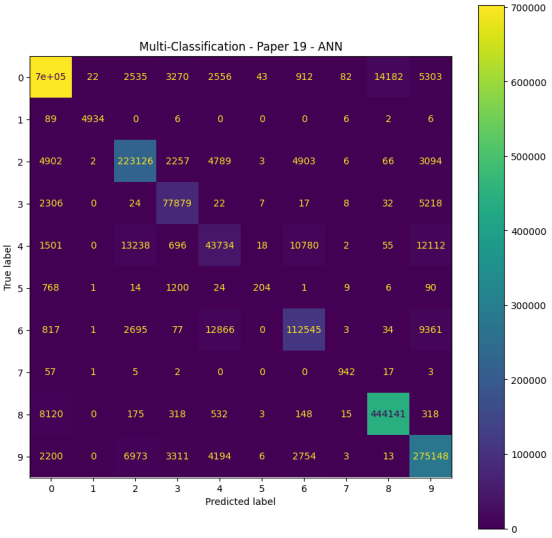


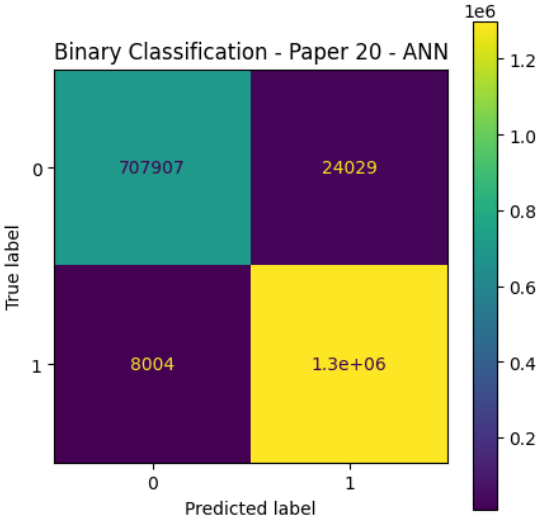


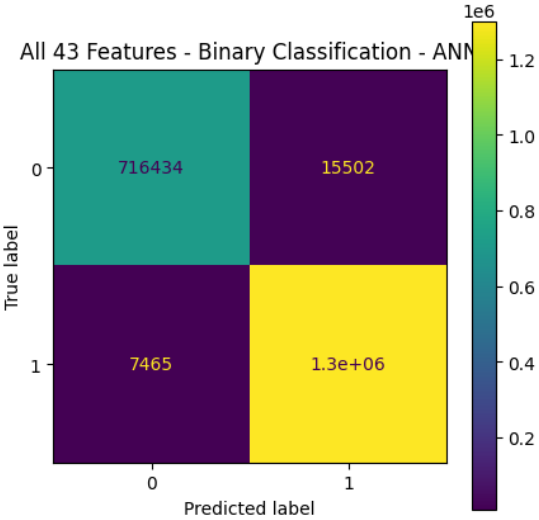

Multi-Classification - Paper 20 - RF
True label
Predicted label


All 43 Features - Multi-Classification - RF
True label
Predicted label


Multi-Classification - Paper 18 - ANN
True label
Predicted label


Multi-Classification - Paper 19 - ANN
True label
Predicted label


Binary Classification - Paper 20 - ANN
707907
24029
8004
1.3e+06
True label
Predicted label
1e6


All 43 Features - Binary Classification - ANN
716434
15502
7465
1.3e+06
True label
Predicted label
1e6

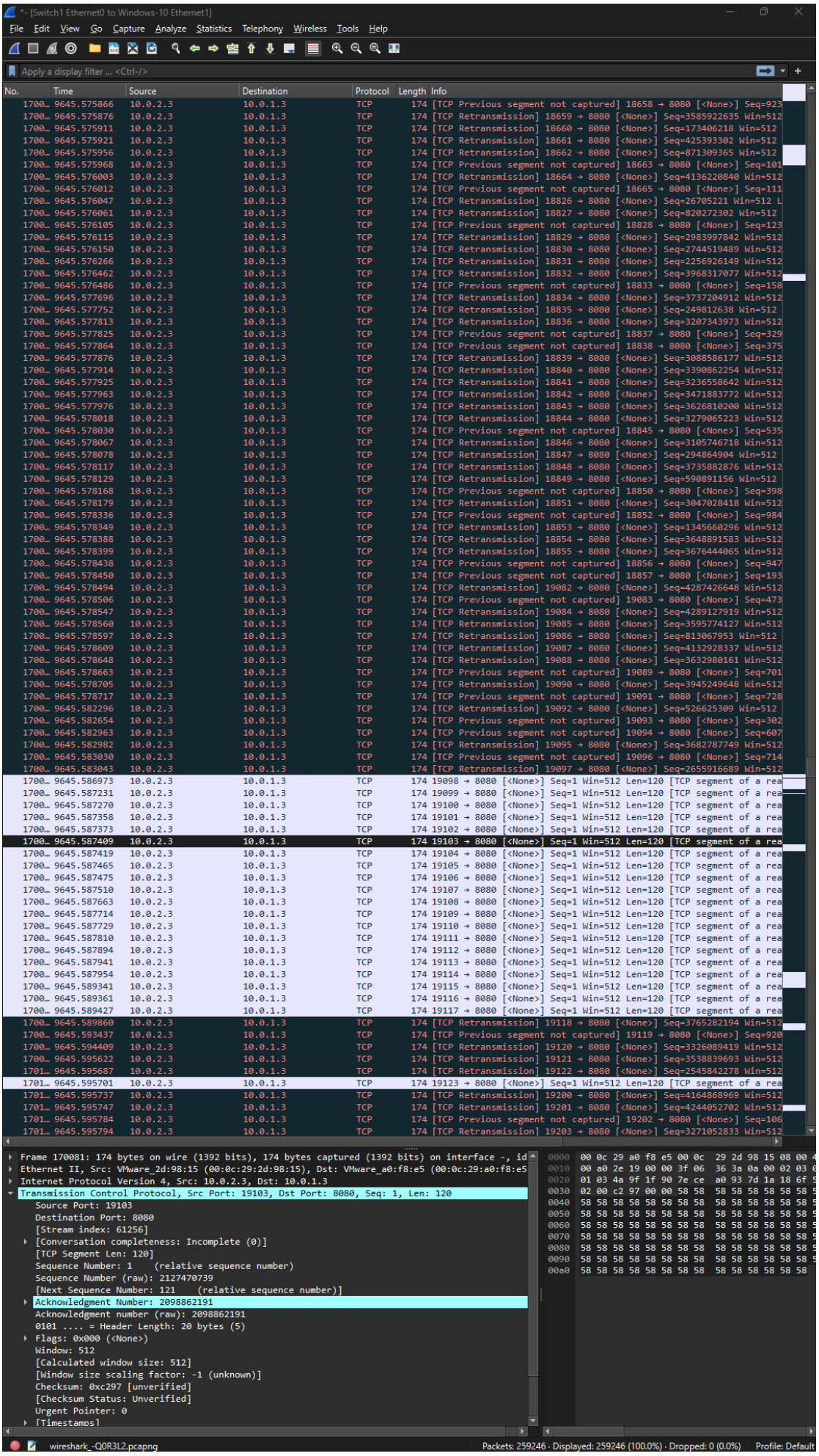


FIGURE C.4: Wireshark screenshot of network during DoS attack

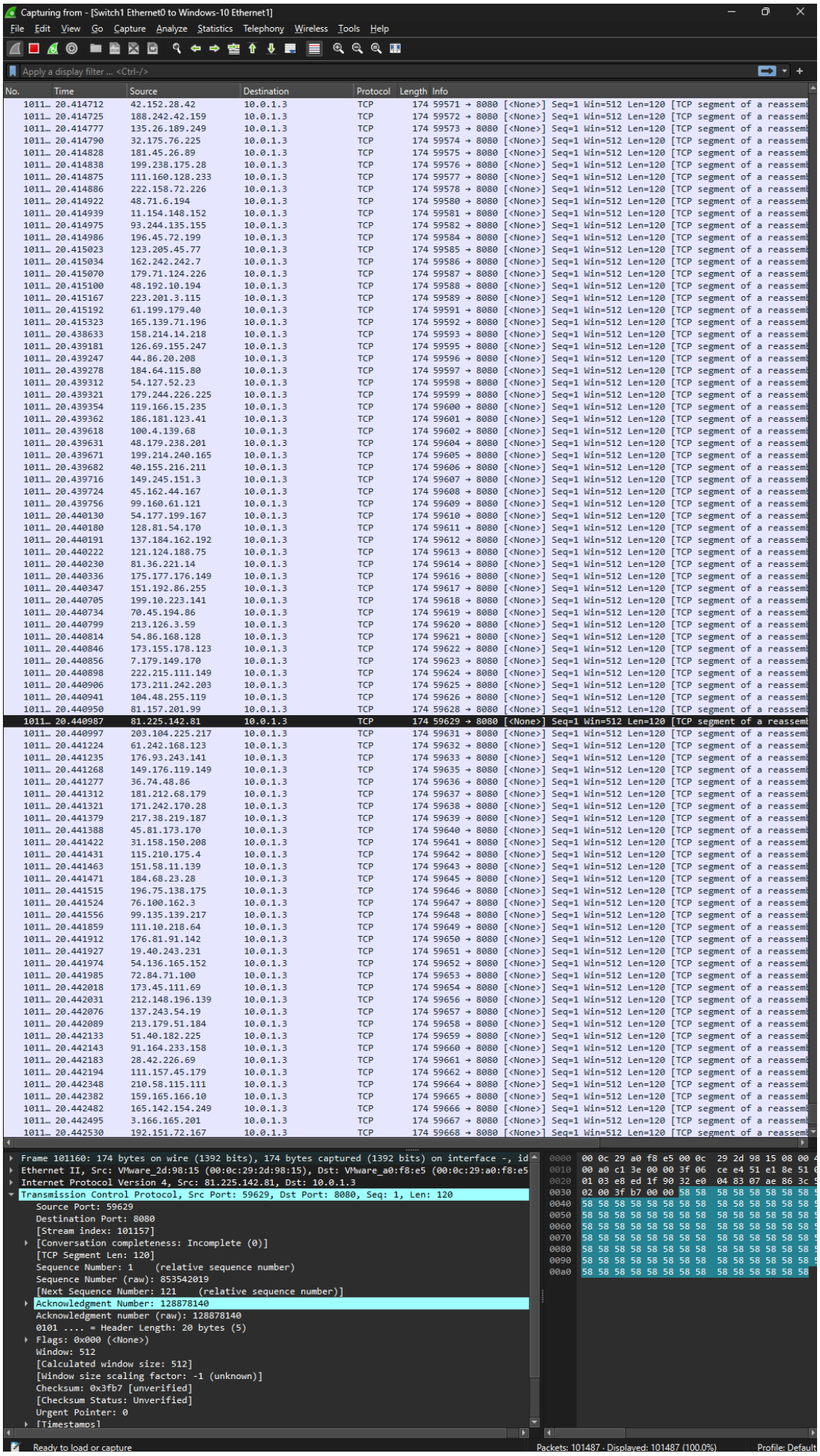


FIGURE C.5: Wireshark screenshot of network during DDoS attack

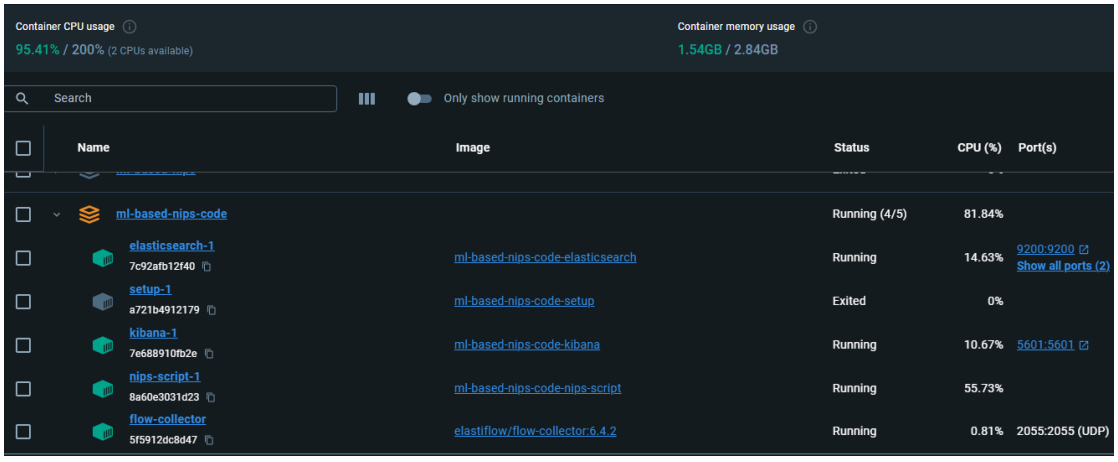


FIGURE C.6: The docker compose stack running