KLE Technological University

Creating Value
Leveraging Knowledge

School of
Electronics and Communication Engineering

Minor-2 Project Report

on

Malware Detection

By:

1. **Zeeshan Mirji** USN: 01FE21BEI019
2. **Muzammil kharadi** USN: 01FE21BEI021
3. **Prajwal Shiggavi** USN: 01FE21BEI032
4. **Abdul Razzak R Yergatti** USN: 01FE21BEI049

Semester: VI, 2023-2024

Under the Guidance of
Prof. Azharuddin

K.L.E SOCIETY'S
KLE Technological University,
HUBBALLI-580031
2023-2024

SCHOOL OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that project entitled " **Malware Detection** " is a bonafide work carried out by the student team of " **Zeeshan Mirji (01FE21BEI019), Muzammil Kharadi (01FE21BEI021), Prajwal Shiggavi (01FE21BEI032), Abdul Razzak R Yergatti (01FE21BEI049)** ". The project report has been approved as it satisfies the requirements with respect to the minor-2 project work prescribed by the university curriculum for BE (VI Semester) in School of Electronics and Communication Engineering of KLE Technological University for the academic year 2023-2024

prof. Azharuddin	**Dr. Suneetha V Budihal**	**Dr. B.S. Anami**
Guide	**Head of School**	**Registrar**

External Viva:

Name of Examiners	**Signature with date**
1.	
2.	

ACKNOWLEDGEMENT

The project Malware detection becomes successful with many individuals' kind support and constant help. We the team under Computer-Networking extends our sincere thanks to each one respectfully. The team expresses sincere appreciation to all the people who have assisted us in completing this project. All their contributions are deeply appreciated and acknowledged. The team would like to thank Dr. Suneetha Budihal, Head, School of Electronics and Communication Engineering for allowing extending our skills in the direction of this project. The team expresses heartfelt gratitude to our guide Prof. Azharuddin, whose valuable insights proved to be vital in contributing to the success of this project

By team:
Zeeshan Mirji
Muzammil Kharadi
Prajwal Shiggavi
Abdul Razzak R Y

ABSTRACT

Malware detection is an important aspect of cybersecurity, aiming at detecting malicious software that poses a threat to computers and networks. With the fast growth of malware, which ranges from viruses and worms to sophisticated ransomware and spyware, effective detection methods are critical. Traditional defense solutions like intrusion detection and thorough packet inspection are not so accurate. Traditional techniques to malware detection include signature-based detection, which uses known patterns, and heuristic or behavioral analysis, which evaluates program behavior to detect suspect activities. The demand for more advanced and continuously innovative methods to combat malware, botnets, and other malicious activities is urgent. Machine Learning (ML) emerges as a promising approach due to increasing computing power and reduced costs, offering potential as either an alternative or complementary defense mechanisms have been employed to enhance detection accuracy by learning from large datasets of known malware behaviors.

This study examines Machine Learning's ability to identify harmful malwares in a network. At the first Netflow datasets are analyzed thoroughly, which generated 22 extracted characteristics. All of these characteristics are then compared to one another using a feature selection procedure. Then, our technique compares five machine learning algorithms to a NetFlow dataset comprising typical botnets.

Results demonstrate that in 8 of the 13 scenarios, the Random Forest Classifier is successful in identifying over 95% of the botnets, and in the most challenging datasets, it detects over 55% of the botnets.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Cybersecurity is developing and the level of online crimes is constantly increasing. Because sophisticated assaults are growing more common and extensive, they are regarded as the new normal. The cybersecurity defense must likewise innovate in response to this ongoing shift.There are already answers, yet these approaches are still often combined. Systems for detecting and preventing network intrusions (IDS/IPS) keep an eye out for hostile activities and policy infractions. Detecting malware that matches recognized signatures is much easier using signature-based systems to detect intrusions. In contrast, an intrusion detection system that is based on behavior analyzes and reports on system abnormalities [1]. Despite their effectiveness, both kinds have some drawbacks. Systems that rely on signatures from recognized threats are useless against zero-day assaults or novel malware samples. Because traditional behavior-based systems rely on a standard profile, which becomes more difficult to establish as networks and applications get more complex, they may not be useful for detecting anomalies.Another approach is full data packet analysis, however it is risky in terms of sensitive user data exposure and computationally costly. Particularly in the subject of cybersecurity, machine learning (ML) has attracted a lot of interest across a wide range of applications and disciplines.

Machine learning techniques may be used to evaluate and categorize undesirable actors from a vast amount of available data, as hardware and processing capacity become more widely available. Hundreds of machine learning algorithms and techniques may be essentially divided into two categories: supervised and unsupervised learning[2]. Approaches to supervised learning are used in the context of regression, in which input is mapped to a continuous output, or classification, in which input matches to an output. Unsupervised learning has been used in dimension reduction and exploratory analysis, and it is mostly achieved by clustering. By using both of these techniques, cybersecurity can analyze malware almost instantly and do away with the drawbacks of conventional detection techniques.

We analyze NetFlow data with our technique. NetFlow records don't reveal private or personally identifiable information (PII), but they do include sufficient information to identify traffic uniquely using qualities like 5-tuples and other fields[3]. For network administration and monitoring, NetFlow is already widely used in conjunction with its open standard version IPFIX. NetFlow is an effective option because of its privacy characteristics and availability of data.

1.1 Motivation

The motivation of undertaking the project is, we noticed in today's digital world, when cyber attacks are becoming more common and complex, detecting and eliminating malware is essential.

Malware attack may ruin computers and networks, by compromising confidential information, interfering with regular business activities, and resulting in financial losses. Modern malware evolves quickly in order to evade detection, therefore traditional measures such as firewalls and antivirus software are no longer sufficient. Machine Learning (ML) provides a possible answer by allowing computers to learn from data and detect patterns of criminal activity. Unlike traditional systems, which rely on predetermined rules, machine learning algorithms may adapt and improve over time, keeping up with new and undiscovered threats. This flexibility is critical as cyber threats grow increasingly sophisticated and diversified, ranging from phishing attempts to advanced persistent threats.

This inspired us to think of a solid solution for the worst-case scenario. Everyone should have access to trustworthy cybersecurity during critical threats, particularly when it is most needed. We may reduce the impact of these threats by enhancing malware detection and ensuring that systems remain safe and functional during cyberattacks. Our objective is to develop a system that is simple to operate and can be promptly implemented in any urgent circumstance. When people need peace of mind, we want them to know that their systems and data are safe. This project aims to improve the world by assisting people and organizations in staying safeguarded and able to withstand cyberattacks. We hope that our efforts will contribute to safeguarding information and minimizing the damage caused by malware and other cyber threats.

1.2 Objectives

This project's primary goal is to use various machine-learning techniques to identify malware or botnet traffic from a NetFlow dataset.

- Develop a precise system that can detect malware or botnet activity of any size N of Netflow datasets, regardless of their clean or with the presence of malware. The system should process the data effectively, with low false positives and high precision in identifying attacks and legitimate traffic, and analyze and classify the data as usual or attack traffic.

- Various supervised, unsupervised, and potentially hybrid machine learning algorithms should be implemented and thoroughly tested. The objective is to evaluate their effectiveness in computational resource requirements, processing speed, and detection accuracy to determine the best methods for various NetFlow dataset types.

- Provide clear, solid recommendations on the best Machine Learning techniques for particular use cases according to the comparison.

1.3 Literature survey

There are significant drawbacks to using traditional techniques for malware detection. One of the earliest techniques is signature-based detection, in which the system searches for the malware's recognized patterns. This is effective against known threats but cannot detect newly released or modified malware that deviates from established patterns. Heuristic-based detection looks for unusual behavior or code in trying to find malware; this method often advertises benign software as harmful, resulting in many false alarms [4].

A further approach that checks the network for unusual activity is anomaly-based detection. It generates a model of typical network activity and marks any deviation from this model as possibly dangerous. Nevertheless, this strategy may also result in many false positives because it may be challenging to characterize "normal" behavior, particularly in diverse and complicated network contexts. To enhance malware detection in light of these constraints, researchers have

turned to machine learning (ML). ML systems can recognize intricate patterns that conventional approaches overlook by learning from vast volumes of data. Additionally, they may adjust to new malware varieties by retraining on fresh data, increasing their effectiveness against ever-changing threats [3].

Supervised and unsupervised learning algorithms are the two primary categories of machine learning algorithms utilized in malware detection. Labeled information is used to train supervised learning systems, with each sample classified as harmful or secure. Standard supervised learning techniques consist of support vector machines (SVM), which determine the best way to separate different classes of data; decision trees, which partition information based on particular traits to make a selection; random forest machines, which combine multiple decision trees to improve accuracy, and neural networks, which are sophisticated models that can recognize highly complex patterns. Conversely, unsupervised learning techniques do not require information to be labeled. They are employed in data analysis to identify trends or abnormalities. Unsupervised techniques encompass clustering, which associates comparable data points, and self-encoder, which can compress data and recognize anomalous patterns by identifying departures from the typical data [4][5].

Hybrid approaches are a way to significantly enhance malware detection by combining supervised and unsupervised learning techniques. For example, an unsupervised model may initially detect possible abnormalities; then, a supervised model could subsequently categorize these anomalies to determine whether or not they are malware. Accuracy is increased, and false positives are decreased with this combo. Generally, conventional malware detection techniques need help to stay current with novel and changing threats. Machine learning offers a more reliable solution by learning from data and adjusting to new virus trends. While unsupervised techniques aid in detecting novel, unidentified dangers, supervised learning algorithms such as decision trees, random forests, SVMs, and neural networks have demonstrated considerable potential. Combining the two approaches can improve detection performance, which makes machine learning an essential tool for contemporary network security [6].

1.4 Problem statement

Detection of Malware attack by analyzing Network Flows using different Machine Learning Algorithms.

1.5 Application in Societal Context

- Effectively detects malware through NetFlow analysis. This project enhances cybersecurity defenses for individuals and organizations. This is used in protecting sensitive data, financial transactions, and personal information from cyber-attacks and promotes trust and security in society
- It is essential to the defense of infrastructure, including transportation networks, power grids, and healthcare systems. In order to maintain operational continuity and public safety, it assists in anticipatorily identifying and mitigating potential cyberattacks that might impair critical services.
- Malware detection in NetFlow can help organizations increase their knowledge of cybersecurity risks and take preventive actions through education. By this, a more knowledgeable public is created. So there will be risks to safeguarding individuals' privacy and making the Internet safer

1.6 Organization of the report

In Chapter 1, discusses about the motivation of the project carried on with Objectives and Literature Survey. The Problem statement is described, followed by the applications in Societal Context and Project planning .

In Chapter 2, describes about Functional block diagram , followed by Algorithm and Flowchart.

In Chapter 3, focuses about the results.

In Chapter 4, presents about the Conclusion and Future scope of the project.

Chapter 2

System design

In this chapter, we will be looking towards the final system architecture which is being implemented and is described by the Algorithm and Flowchart.

2.1 Block Diagram

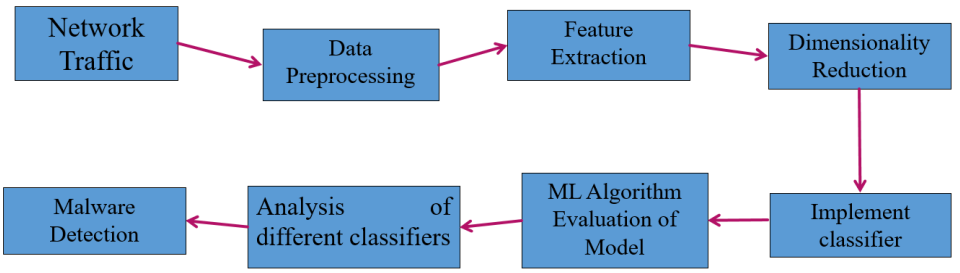

Figure 2.1: Block Diagram

1)Network Traffic: The gathering of network traffic data is the first step in the malware detection procedure. Numerous pieces of information are included in this data, such as the number of packets and bytes exchanged, the protocols utilized, source and destination IP addresses, and port numbers. Every network flow's timestamp is also logged. The fundamental input for the complete malware detection procedure is this raw data. We make sure that our next study has a complete dataset from which to discover possible malicious behaviors by recording all pertinent network-traffic.

2)Data Preprocessing: Following collection, the raw network traffic data is preprocessed. This is a crucial step since it makes sure the data is clean and well-organized, ready for additional analysis. The process of data cleaning is eliminating any unnecessary, duplicate, or incomplete records that might bias the analysis. In order to standardize the data and guarantee that every characteristic is on a same scale, normalization is also carried out. This stage makes the data more consistent and manageable, which enhances the effectiveness of machine learning models.

3)Feature Extraction: Relevant properties are located and retrieved from the preprocessed data during the feature extraction stage. This entails choosing crucial elements that are

most likely to aid in differentiating between malicious and benign network data. These characteristics include things like the length of the connection, the protocols being utilized, and the packet and byte speeds. By concentrating on these useful characteristics, we can improve the accuracy of malware detection by streamlining the data and making it easier for machine learning algorithms.

4)Dimensionality Reduction: Techniques for dimensionality reduction are used to further filter the data. The goal of this stage is to minimize the amount of characteristics while maintaining the crucial data required for a thorough analysis. To accomplish this reduction, methods like Principal Component Analysis (PCA) or t-SNE may be applied. The curse of dimensionality, which can make analysis and model training more difficult, is lessened with the use of dimensionality reduction. Additionally, it lowers computational complexity and enhances machine learning models' interpretability and performance.

5)Implement Classifier: Using different machine learning classifiers is the next step after having a simplified dataset available. These classifiers are algorithms that are meant to learn from the data and forecast the likelihood of dangerous or benign network traffic. Dense Neural networks, Random Forests, Support Vector Machines (SVM), and Logistic Regression are a few common classifiers. Every classifier has advantages and disadvantages, and using a variety of them enables a thorough analysis to choose the best one for our particular use case.

6)ML Algorithm Evaluation of Model: After implementing the classifiers, their performance needs to be rigorously evaluated. This involves using a set of evaluation metrics to assess how well each model performs in distinguishing between benign and malicious traffic. Common metrics include Accuracy, Precision, Recall, F1 Score, and the Scatter-Plot. Evaluating the models helps identify any potential shortcomings and ensures that the classifiers are reliable and effective.

7)Analysis of Different Classifiers: The performance of the different classifiers is examined and contrasted after the assessment. The comparison analysis is essential to determine which model performs the best. We can identify which classifier offers the best accuracy and dependability in malware detection by looking at how each one performs on the assessment metrics. By doing this step, we can be confident that the best tool for our malware detection system is chosen.

8)Malware Detection: The procedure culminates in the real malware detection step, which employs the top-performing classifier found during the analysis stage. This classifier is used to continually assess incoming network data in a real-time monitoring system. Any communication that seems suspicious can be appropriately flagged by the classifier based on assessment and training data. For network security to be maintained and possible attacks to be quickly addressed, real-time detection is crucial.

2.2 Implementation details

2.2.1 Feature Extraction

Motivation

NetFlow data, commonly used for network traffic analysis, presents a significant hurdle: most of the information is categorized. While converting these categories into yes/no (boolean) columns might seem intuitive, it leads to a massive increase in the data matrix size. This expansion can cause memory errors, even when using compressed sparse matrix formats. To overcome this obstacle, researchers have explored alternative approaches, including feature extraction techniques identified through a review of network traffic analysis literature.

Use of time windows

A common approach for analyzing NetFlow data involves summarizing it within time windows. This strategy capitalizes on the "temporal locality behavior" of botnets, meaning their activity often clusters within specific timeframes. Summarization also offers two advantages:
Reduced Data Size: This makes it easier to handle large datasets and potentially improves processing speed.
Real-Time Detection: By analyzing summarized data after each time window, we can achieve near real-time botnet detection.
However, defining the optimal time window size (width) and the gap between windows (stride) remains a challenge. Existing research provides various options, often based on experience rather than clear data-driven methods. Here, we opted for a 2-minute window width with a 1-minute stride.

Another hurdle emerged when considering how to group NetFlow entries within a window. Using connection start time resulted in a significant data size increase. To address this, we opted to utilize the communication duration as an additional feature within the window summary.

Extracted features

NetFlow data is a valuable resource for network traffic analysis, but it presents a particular challenge: most of the data is categorical. Converting these categories into numerical values can be problematic because it creates a significant increase in the size of the data matrix. This can lead to memory errors, even when using compressed data formats.
To address this challenge, researchers have developed techniques to extract new features from the data. These features are based on statistical summaries of the original categorical features.

2.2.2 Feature Selection

Embedded methods

Embedded methods offer another approach to feature selection in network traffic analysis. Unlike filter methods that rely solely on feature characteristics, embedded methods leverage the classifier itself during the training process. This allows them to select features based on their contribution to the classifier's performance, essentially using the classifier's "score" (prediction accuracy) as a guide.

Lasso and Ridge logistic regressionLogistic regression is a popular choice for network traffic analysis tasks. However, it can struggle with imbalanced datasets, where one class (e.g., normal traffic) significantly outweighs the other (e.g., botnet traffic). This imbalance can bias the model towards the majority class, leading to poor detection of the minority class.

To address this challenge, we can leverage cross-validation for data balancing. Cross-validation involves splitting the data into training and testing sets multiple times. During training, we can employ techniques like class weighting within the logistic regression model.By using cross-validation, we can evaluate the effectiveness of different weight values and ultimately identify the best configuration for balancing the dataset and achieving optimal performance with logistic regression. In our analysis, we found that the best F1-score is achieved by a class weight

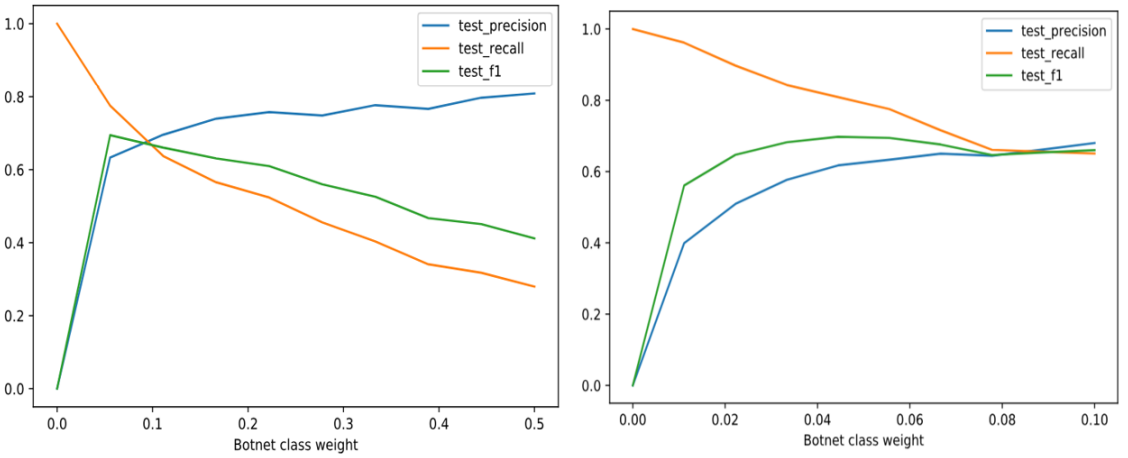

Figure 2.2: Scores wrt Botnet class weight

of 0.044 within the logistic regression model (refer to Figures 2.2). This weight balances botnet data's influence with background data efficiently, making sure that the model accords more attention to rare botnet examples. For further improvement on the performance of our model , we will be using L2 regularization (with Ridge) when choosing out features. This method will enable us to identify the most relevant variables and coefficients in our equation model.

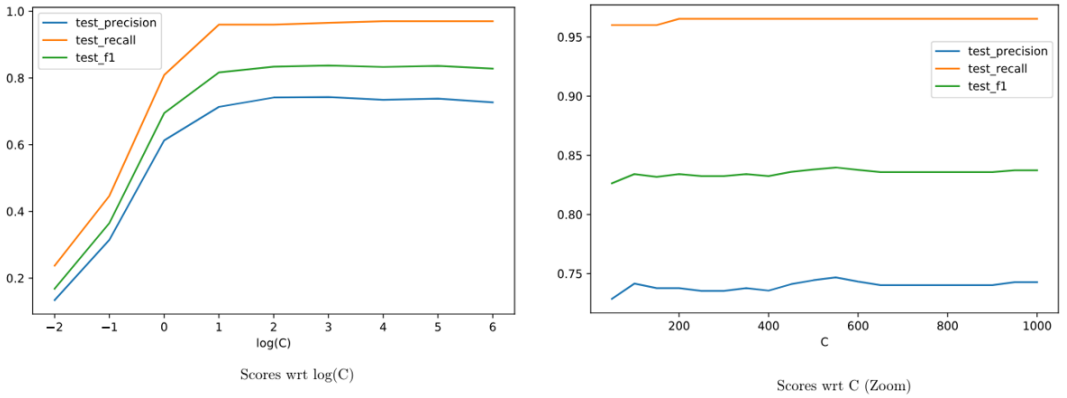

Figure 2.3: Scores wrt log(C)

The figure 2.3 show the evolution of the different scores with respect to the regularization parameter C. The best f1 score is obtained for C =550 so the feature coefficients must be analyzed with this parameter.

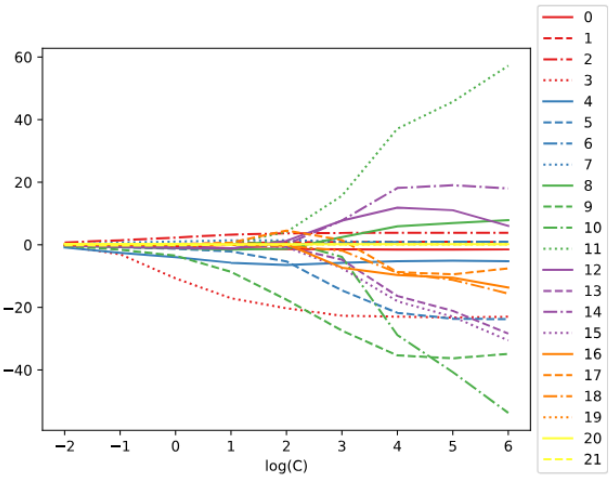

Figure 2.4: Feature coefficients wrt log(C)

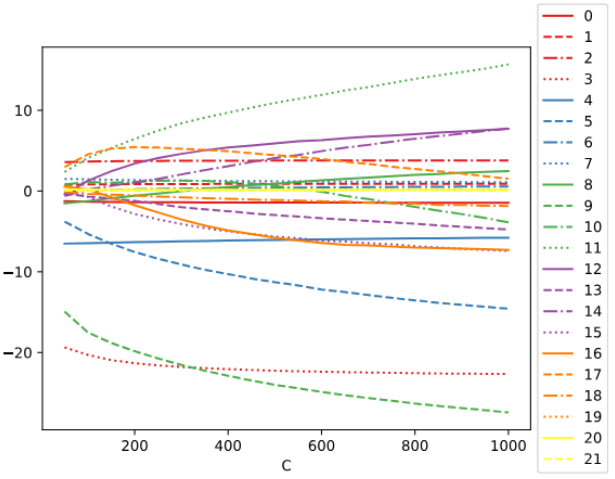

Figure 2.5: Feature coefficients wrt C (Zoom)

It was hard to use L1 regularization efficiently because it has some problems such as convergence issues which hinder implementation. Therefore, L2 regularization, otherwise known as Ridge, is still preferred over others because of increased computational power which enhances feature significance understanding.

2.2.3 Dimensionality Reduction

Using dimensionality reduction techniques is another way to reduce the number of features. The dataset's visual representation is also very useful.
Principal Component Analysis (PCA): The Principal Component Analysis method results in a set of linearly uncorrected variables and allows the dataset dimension to be reduced.

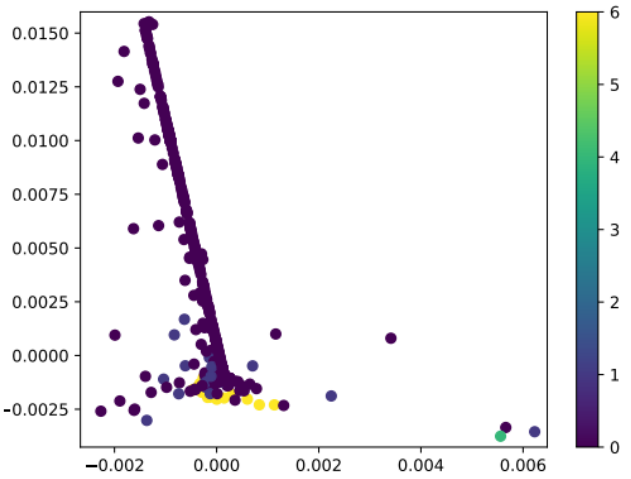

Figure 2.6: Principle Component Analysis

Fig 2.6 is the representation of the initial two PCA components with the pre-selected feature set is shown in Figure. Six botnets are identified, and specific non-botnet points are hidden. 58 percent of the dataset variation is represented by the first component, the vertical axis, and the second component, the horizontal axis, explains 35 percent of the total variation.

The botnet indicates that it meets at the base of a long, straight line of background points. One can train a K-Nearest Neighbors classifier using this representation.

t-SNE (t-distributed Stochastic Neighbour Embedding): Another technique to reduce the dataset's dimension to two or three features is t-SNE. The botnets are labeled as 6,

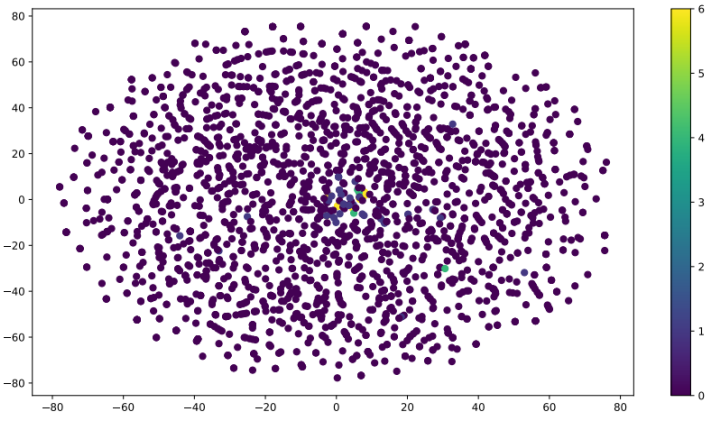

Figure 2.7: t-SNE

which displays the outcome of the dimensionality reduction. The botnet points are dispersed throughout the network, making distinguishing them from real traffic difficult. The algorithm also uses a lot of memory and time. Therefore, in this Report, the K-nearest neighbor method is no longer considered for detecting botnet activities.

2.3 Algorithm

- Start
- Gather Network Flow Dataset: Compile information on network traffic. Tools for network monitoring, such as NetFlow, can be used to do this. These tools can record data such as flow records and packet headers.
- Preprocessing Data: Clean Data (Remove unwanted Labels) and Label different features.
- Extract Features: Transform unprocessed network traffic data into machine learning-useful features. Source and destination ports, source and destination IP addresses, and other standard features, Quantity of packets, Size of the packet
- Dimensionality reduction: to distill large datasets into more manageable and relevant features, enhancing the performance and speed of our detection algorithms.
- Train machine learning models by netflow dataset.
- Analyze network flow data using trained models and test different machine learning algorithms.
- Determine which model performs the best using the evaluation metrics.
- End

2.4 Flowchart

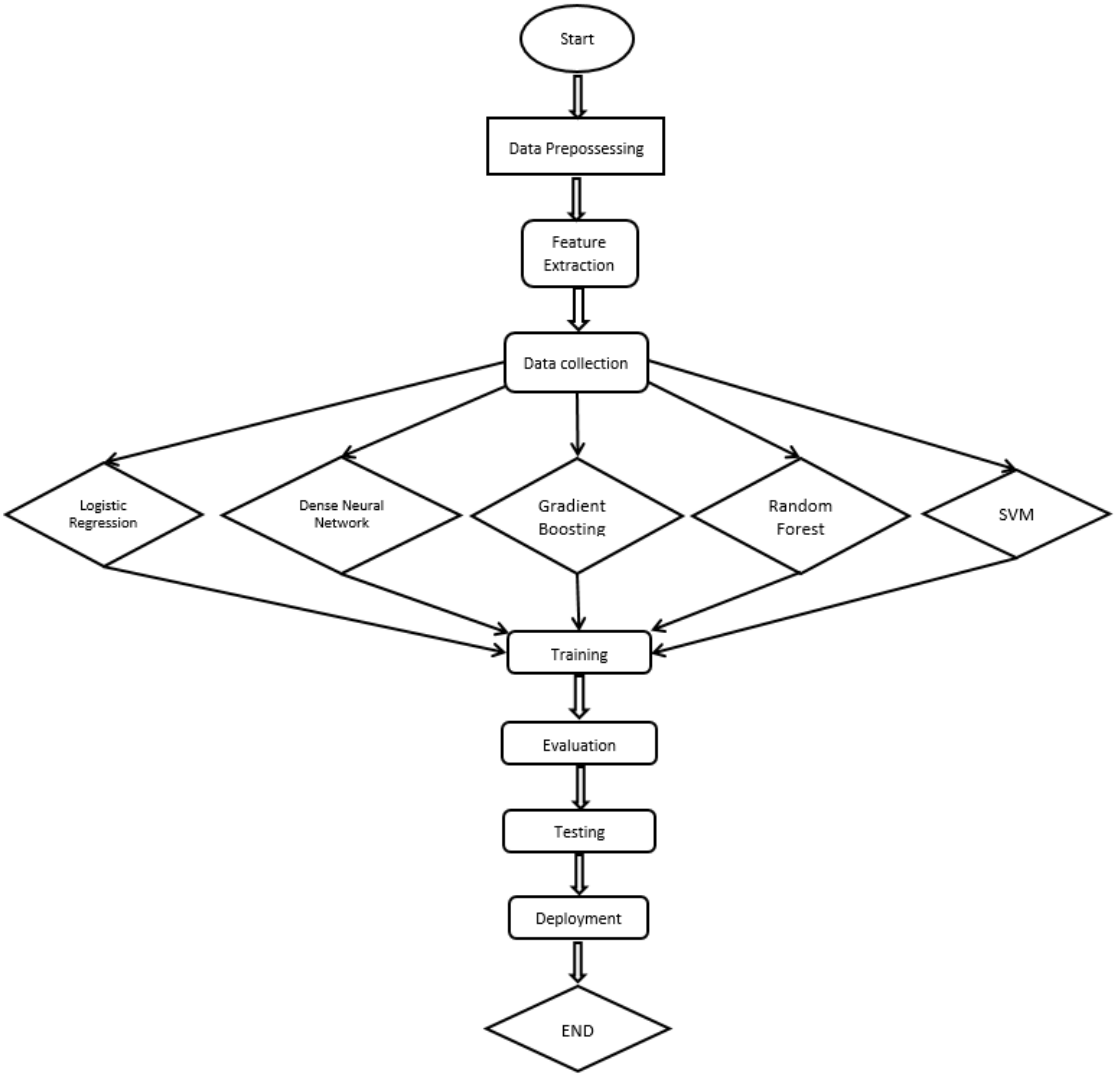

Figure 2.8: Flow Chart

Starting with the flow chart, Data preprocessing, which involves cleaning and preparing raw network data, is where the flowchart begins. Subsequently, the Data Collection stage combines the processed data into a dataset to extract meaningful attributes for Feature Extraction. The next step is Model Selection, where appropriate machine learning algorithms like Random Forest, SVM, Gradient Boosting, Dense Neural Networks, and Logistic Regression are selected. After using the dataset for training, the model's performance is assessed. It is tested on fresh, untested data to ensure the model is accurate and robust. The process ends when the verified model is deployed for real-time malware detection in network traffic.

Chapter 3

Results and discussions

3.1 Dataset

Selecting a Dataset:

- We have collected 3 types of public domain datasets with traffic flow data they are CCCS-CIC-AndMal-2020, CICAndMal2017 & CTU-13.
- We have chosen to use CTU-13 over other public datasets because it is highly available and has been used quite extensively for many similar research studies in the past.
- The features are the raw attributes in the Netflow data - StartTime, Duration, Proto, SrcAddr, Sport, Dir, DstAddr, Dport, State, sTos, dTos, TotPkts, TotBytes, SrcBytes and Label.

ID	Duration (hrs)	# Flows	Bot	#Bots	Activity
1	6.15	2,824,637	Neris	1	IRC, SPAM, CF
2	4.21	1,808,123	Neris	1	IRC, SPAM, CF
3	66.85	4,710,639	Rbot	1	IRC, PS
4	4.21	1,121,077	Rbot	1	IRC, DDoS
5	11.63	129,833	Virut	1	SPAM, PS
6	2.18	558,920	Menti	1	PS
7	0.38	114,078	Sogou	1	HTTP
8	19.5	2,954,231	Murlo	1	PS
9	5.18	2,753,885	Neris	10	IRC, SPAM, CF, PS
10	4.75	1,309,792	Rbot	10	IRC, DdoS
11	0.26	107,252	Rbot	3	IRC, DdoS
12	1.21	325,472	NSIS.ay	3	IRC, P2P
13	16.36	1,925,150	Virut	1	HTTP, SPAM, PS

Figure 3.1: 13 different Scenarios of the CTU-13 Dataset

The CTU-13 Dataset was created by the CTU University 2011 and captures real botnet traffic mixed with normal traffic and background traffic. It is made of 13 captures of different botnet samples summarized in 13 bidirectional NetFlow files.
The first scenario, which uses a malware called Neris. The capture lasted 6.15 hours during which the botnet used HTTP based C&C channels to send SPAM and perform Click-Fraud. The NetFlow file weights 368 Mo and is made of bidirectional communications described by 15 features.

3.2 Metrics

It is necessary to select certain measures to assess the effectiveness of the techniques in order to compare the outcomes of various algorithms. The conventional method is to count the number of false positives, or background communications that are classified as botnets, and false negatives, or botnets that are classified as background communications.

To do this, we consider 3 scores as follows:

- the Recall: R = tp/(tp + fn)
- the Precision: P = tp/(tp + fp)
- the f1 Score: f1 = 2 ×(R × P/R + P)

Remember that a low recall is worse than a low precision for our project for detecting malicious software since it indicates that the majority of detected communications are botnets (precision) but the majority of botnet communications go unnoticed (recall).

Naturally, it's simple to have a good recall because all you need to do is mark each transmission as coming from a botnet. In order to ensure that the recall is not too low, the compromise that has been selected is to optimize the f1 score.

3.3 Algorithms

3.3.1 Logistic Regression

An algorithm called the Logistic Regression approach classifies the flows by combining characteristics in a linear fashion. To fine-tune the model's parameters, cross-validation is required. The final values that were selected were Weight non-botnet = 0.044 and C = 550.

3.3.2 Support Vector Machine

The Support Vector Machine technique transforms the data space using kernels and then looks for a distinct line to divide the data into two classes. Because cross-validation is required to adjust the model's parameters. The selected alpha parameter for a linear kernel is alpha = 10 raise to 9.

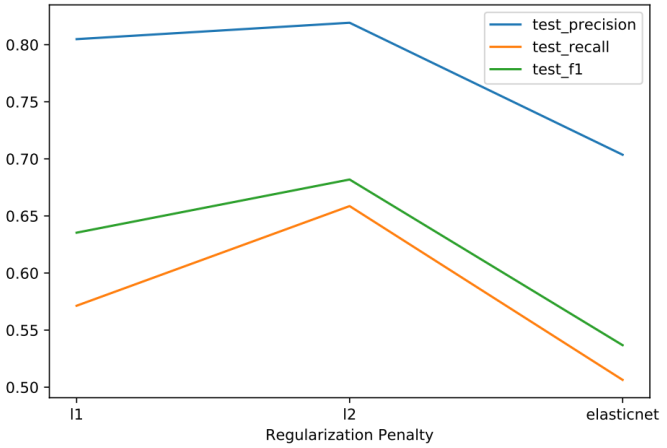

Figure 3.2: Scores wrt Regularization Penalty

Fig 3.2 To determine the most effective regularization technique, cross-validation is used. Based on the findings, a l2 regularization is recommended..

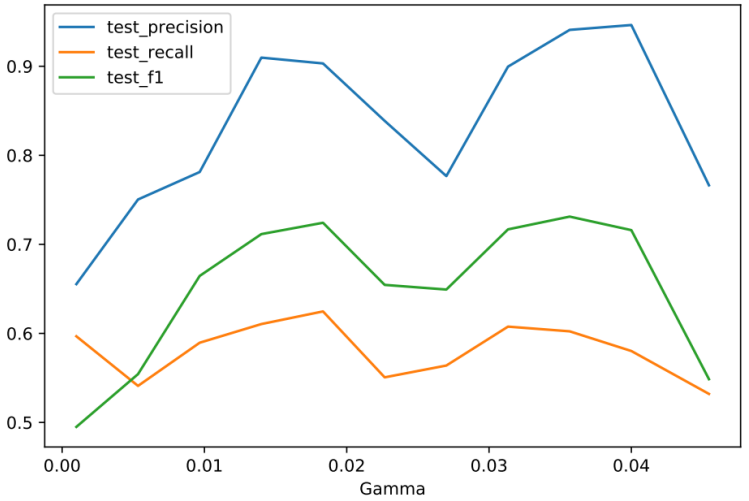

Figure 3.3: Scores wrt Gamma

Fig 3.3 displays the outcomes The optimal Y parameter for a Radial Basis Function (RBF) kernel is Y = 0.03567.

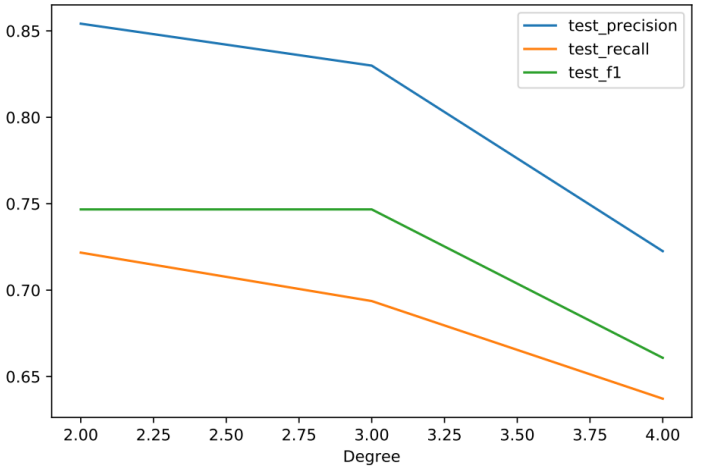

Figure 3.4: Scores wrt Polynomial Degree

Fig 3.4 Illustrates that the optimal degree for a polynomial function with a polynomial kernel is 2.

In conclusion, a polynomial kernel with a degree of two is the optimal set of support vector machines (SVM) parameters for botnet identification in the first scenario of CTU-13.

3.3.3 Random Forest

Several decision tree classifiers are used by the Random Forest method to estimate the class of each input flow. There are one hundred (100) trees in the forest, as chosen.

3.3.4 Gradient Boosting

Using the score from one decision tree to construct a new one, the Gradient Boosting algorithm builds new trees with the goal of gradually improving training. The function loss and the maximum depth of the trees (100 is the specified number of trees) are the two primary parameters that need to be adjusted.

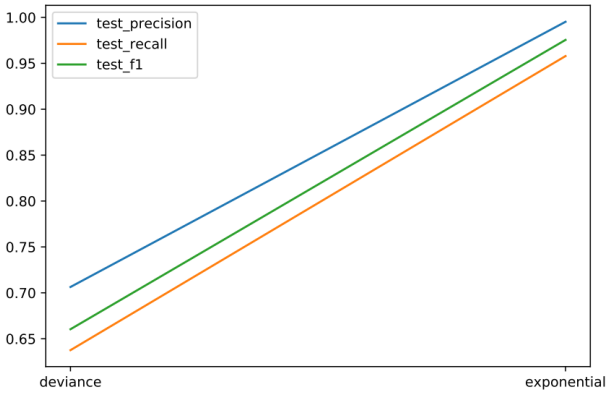

Figure 3.5: Scores wrt chosen Loss

Fig 3.5 shows that an exponential loss performs better than a deviance loss.

3.3.5 Dense Neural Network

Neural networks are getting more and more popular lately since they function very well with large amounts of data. Here, we evaluate a basic two-layered neural network, which is dense (or completely linked) and includes 256 neurons in its first hidden layer and 128 in its second.
The batch-normalization, no dropout, and ReLU activation function comprise the neural network's parameters (with the exception of the output layer, which uses a sigmoid function). There are 768 non-trainable parameters and 39 681 trainable parameters in the model. The binary cross-entropy loss for the training set and the validation set—which represents 15% of the entire set-across 10 epochs is displayed in Figure 3.6 (a batch of 32 is utilized).

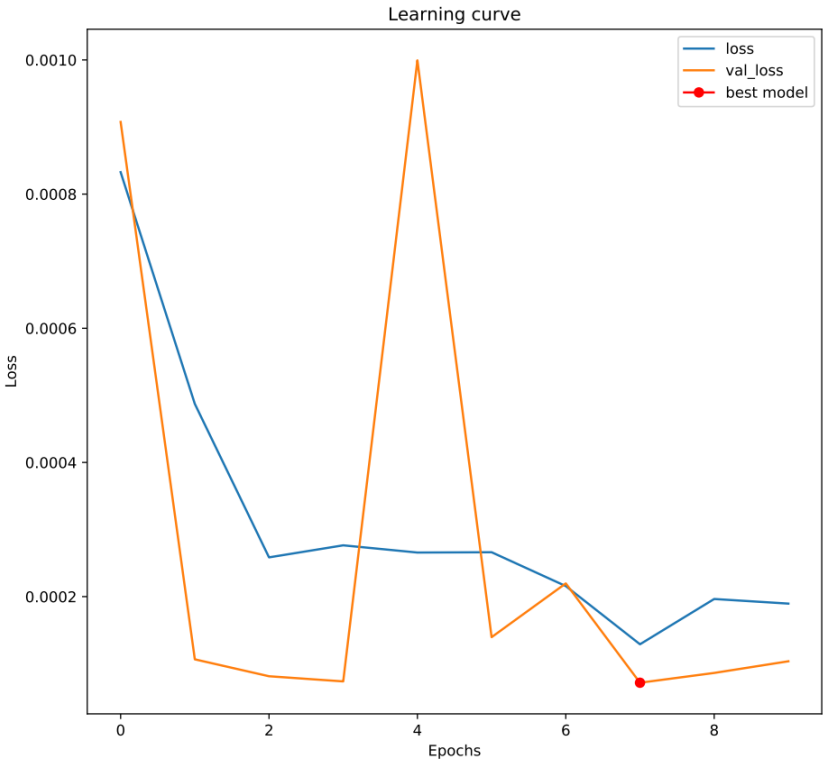

Figure 3.6: Learning curve of the Neural Network

3.4 Comparison of algorithms

We train our models using two thirds of the dataset (randomly selected NetFlows) and test them using the remaining one third of the dataset in order to compare the approaches.

Algorithm	**Training**			**Test**		
	Prsn	**Recl**	**f1-Scr**	**Prsn**	**Recl**	**f1-Scr**
Logistic Regression	0.74	0.97	0.83	0.74	0.96	0.84
Support Vector Machine	0.94	0.80	0.86	0.91	0.78	0.86
Random Forest	1.0	1.0	1.0	1.0	0.95	0.98
Gradient Boosting	1.0	0.99	1.0	0.98	0.96	0.97
Dense Neural Network	0.90	0.98	0.94	0.96	0.99	0.98

Table 3.1: Algorithm performance metrics

The above Table 3.1 shows Botnet Neris Scenario 1 - Result Summary which demonstrates that, with a f1 score of 0.97, Random Forest, Gradient Boosting, and the Dense Neural Network outperform other studied algorithms in the detection of botnets among network traffic. However, both the Support Vector Machine and the Logistic Regression approaches achieve a f1 score of 0.85; the latter with a low recall and the former with a low accuracy.

Detection of different Botnets

We test the Random Forest Classifier on all the other situations to check if the outcomes on Scenario 1 of the CTU-13 dataset can be applied to other scenarios.

Size	**Botnets**	**Training**			**Test**		
		Prsn	**Recl**	**f1-Scr**	**Prsn**	**Recl**	**f1-Scr**
Neris - Scenario 1	2 226 720	1.28%	1.0	1.0	1.0	0.95	0.975
Neris - Scenario 2	1 431 539	1.45%	1.0	1.0	1.0	0.98	0.99
Rbot - Scenario 3	2 024 053	4.99%	1.0	0.99	1.0	0.96	0.98
Rbot - Scenario 4	470 663	2.36%	1.0	0.95	1.0	0.69	0.75
Virut - Scenario 5	63 643	3.46%	1.0	1.0	1.0	0.25	0.4
DonBot - Scenario 6	220 863	5.57%	1.0	1.0	1.0	0.9	0.95
Sogou - Scenario 7	50 629	1.38%	1.0	1.0	1.0	0.25	0.4
Murlo - Scenario 8	1 643 574	6.8%	1.0	1.0	1.0	0.94	0.97
Neris - Scenario 9	1 168 424	12.51%	1.0	1.0	1.0	0.94	0.97
Rbot - Scenario 10	559 194	9.67%	1.0	0.99	1.0	0.9	0.95
Rbot - Scenario 11	61 551	2.76%	1.0	1.0	0.5	0.33	0.4
NSIS.ay - Scenario 12	156 790	10.2%	1.0	0.9	0.92	0.41	0.54
Virut - Scenario 13	1 294 025	7.57%	1.0	1.0	1.0	0.96	0.98

Table 3.2: Botnet dataset training and test metrics

Here Prsn is Precision, Recl is Recall and f1-scr is f1 Score.
Table 3.2 demonstrates that, in 8 out of 13 cases, the Random Forest Classifier is successful in identifying the majority of botnets. The quantity of the dataset appears to be a factor in the five other scenarios' low results.

Chapter 4

Conclusion and future scope

4.1 Conclusion

In summary, our research endeavors to develop and evaluate algorithms capable of identifying botnet activity inside real-world network traffic, as modeled by Netflow datasets. Relevant characteristics were retrieved following a thorough examination of the data and several reviews of network security literature. After that, their impact was examined through a selection procedure, although none of the features were so subpar as to be excluded from the training portion. After that, other algorithms were evaluated, including a Dense Neural Network, Random Forest, Support Vector Machine, Gradient Boosting, and Logistic Regression. For eight out of the thirteen cases, the Random Forest Classifier was used to detect botnets, yielding a detection accuracy of over 95 percentage of the botnets.

Following that, our crew concentrated on improving the accuracy for the five most challenging circumstances. Over 55 percentage of the situations 5, 7, and 11 have been detected thanks to the bootstrap method's usage of more data. It was only difficult to raise the scores for situations 4 and 12 (f1 values of 0.75 for scenario 4 and 0.54 for scenario 12). This might be because more sophisticated techniques (such recursive deep neural networks) are required, or it could be the result of a poor representation of the botnet behaviors using the extracted data.

4.2 Future scope

A potential avenue for improving the work showcased here would be to experiment with varying the widths and strides of the collected features for the time frame and investigate additional hyperparameters for the challenging cases. Attempting to train and test several situations concurrently is another possibility. Lastly, without utilizing the data's labels, unsupervised learning may be used to assess the behavior of botnets.

References

1. Akhtar, Muhammad Shoaib and Feng, Tao , "EVALUATION OF MACHINE LEARNING ALGORITHMS FOR MALWARE DETECTION" Presented at MDPI 2023

2. Codina Poquet, Joël, "DETECTION OF MALWARE TRAFFIC WITH NETFLOW" Presented at Universitat Politècnica de Catalunya 2017

3. Kozik, Rafał and Młodzikowski, Robert and Choraś, Michał, "COMPUTER INFORMATION SYSTEMS AND INDUSTRIAL MANAGEMENT: 16TH IFIP TC8 INTERNATIONAL CONFERENCE, CISIM 2017, BIALYSTOK, POLAND, JUNE 16-18, 2017, PROCEEDINGS 16" Presented at MDPI 2017

4. Kaspersky, B, "MACHINE LEARNING METHODS FOR MALWARE DETECTION" Presented at MDPI . 2020

5. Bekerman, Dmitri and Shapira, Bracha and Rokach, Lior and Bar, Ariel, "2015 IEEE CONFERENCE ON COMMUNICATIONS AND NETWORK SECURITY (CNS)" Presented at IEEE . 2015

6. Khammas, Ban Mohammed and Monemi, Alireza and Bassi, Joseph Stephen and Ismail, Ismahani and Nor, Sulaiman Mohd and Marsono, Muhammad Nadzir, "FEATURE SELECTION AND MACHINE LEARNING CLASSIFICATION FOR MALWARE DETECTION" Presented at MDPI . 2015