EFFECTIVE MALICIOUS FEATURES EXTRACTION AND CLASSIFICATION FOR INCIDENT HANDLING SYSTEMS

CHO CHO SAN

UNIVERSITY OF COMPUTER STUDIES, YANGON

OCTOBER, 2019

Effective Malicious Features Extraction and Classification for Incident Handling Systems

Cho Cho San

University of Computer Studies, Yangon

A thesis submitted to the University of Computer Studies, Yangon in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

October, 2019

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

…..…… ……..……

Date Cho Cho San

ACKNOWLEDGEMENTS

First of all, I would like to thank Hist Excellency, the Minister for the Ministry of Education, for providing full facilities support during the Ph.D. course at the University of Computer Studies, Yangon.

Secondly, my profound gratitude goes to Dr. Mie Mie Thet Thwin, Rector of the University of Computer Studies, Yangon, for allowing me to develop this research and giving me general guidance during the period of my study.

I would like to express my greatest pleasure and the deepest appreciation to my supervisor, Dr. Mie Mie Su Thwin, Professor, the University of Computer Studies, Yangon, for her excellent guidance, caring, patient supervision, and providing me with excellent ideas throughout the study of this thesis.

I would also like to extend my special appreciation to Dr. Khine Moe Nwe, Professor and Course-coordinator of the Ph.D. 9th Batch, the University of Computer Studies, Yangon, for her useful comments, advice, and insight which are invaluable through the process of researching and writing this dissertation.

I would like to express my special appreciation and thanks to my external examiner Dr. Thandar Phyu, Director of Technology Group, ATG Company Ltd., for useful comments and suggestions.

I deeply would like to express my respectful gratitude to Daw Aye Aye Khine, Associate Professor, Head of English Department, for her valuable supports from the language point of view and pointed out the correct usage not only throughout the Ph.D. course work but also in my dissertation.

My sincere thanks also go to all my respectful Professors and teachers for giving me valuable lecture and knowledge during the Ph.D. course work and dissertation.

I also thank my respectful Professor Dr. Abhishek Vaish for his valuable comments, advice, and insight which are invaluable to me.

Moreover, I thank my friends from the Ph.D. 9th Batch for providing support, care, co-operation and encouragement during this way. Special thanks go to all the past and current members at Cyber Security Research Laboratory.

Last but not least, I am very much indebted to my parents, U Than Kyaw and Daw Yee Shwe, my baby sister, aunts and uncles for always believing in me; for providing me with unfailing support and continuous encouragement; and for their

endless love and support during these years of my Ph.D. study and through the process of researching and writing this dissertation. This accomplishment would not have been possible without them.

ABSTRACT

Each and every day, malicious software writers continue to create new variants, new innovation, new infection, and more obfuscated malware by using packing and encrypting techniques. Malicious software classification and detection play an important role and a big challenge for cyber security research. Due to the increasing rate of false alarm, the accurate classification and detection of malware is a big necessity issue to be solved.

This research provides the classification system to differentiate malware from benign and classify malicious types. This research contributes the Malicious Sample Names Extraction (MSNE) procedure and Naming Malicious Samples using the Regular Expression (NMS_RE) technique have been contributed to label the malicious samples. This research also contributes the prominent Malware Feature Extraction Algorithm (MFEA) to point out the dominant features based on the generated report files. The features are API, DLL, and PROCESS called by malicious and benign executables through automated analysis. During the experiments, data cleansing for extracted raw data, applying the n-gram technique, and representing and preparing the malicious dataset have been performed to provide the malware classification system.

This research work makes use of two malicious datasets for malware classification. The Benign Malware Classification (BMC) dataset is used for binary class classification system to identify malicious or not and Benign Malware Family Classification (BMFC) dataset is used for multi-class classification system to identify malware family. Chi-Square and Principal Component Analysis (PCA) feature selection methods have been applied in this system to select the best features. Classification algorithms like k-Nearest Neighbor (kNN), Random Forest (RF) and Support Vector Classification (SVC) have been used for multi-class and binary class classification. The proposed approach is able to classify the malicious and benign executable files effectively.

This research work provides malware classification using Machine Learning (ML) classifiers. The findings from the experiment prove that the extracted API_DLL features provide the best evaluation metrics in terms of accuracy, confusion matrix (CM), True Positive Rate (TPR), False Positive Rate (FPR), and Receiver Operating Characteristic (ROC) curve area.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

CHAPTER 1

INTRODUCTION

During these years, new malwares have increased exponentially, which produces difficulty for malware experts and anti-virus vendors to profile the families, as they need to extract information out of this large-scale data. Though many anti-virus vendor companies, for instance BitDefender, Avast, Microsoft, and Kaspersky, provide for detecting and profiling the malicious samples, the number of malwares is rising more than ever in current together with their seriousness. Malware continues to be a blight on the threat landscape with more than three hundred million new variants detected in 2016 [80]. The goal line of malicious software analysis is to gain an understanding of the behaviors and functions of malware so that the analysts can build the defense procedure to protect an organization's sensitive information and network postures. Several new malware variants are very effective for evasion the anti-virus and anti-malware applications, and malware creators often use a similar method of attacking to build new malware in a short period. The polymorphic and metamorphic malware are increasing and attackers may use them as long as there is a profit to be made. Therefore, distinguishing and classifying the nature of malicious executables from each other are very important in malware classification.

1.1 The Importance of Malicious Software Analysis on Cyber Security

Today's modern advanced malicious software (malware) has been integrated multi-module systems using sophisticated or obfuscated techniques to attack and target the vulnerable or weak systems. The sophisticated threat actors, Advanced Persistent Threats (APT), continued to make the headlines with audacious politically motivated attacks and thefts on target organization. Any gaps in network security, software patching, or employee awareness will ruthlessly expose in the wave of destructive malware attacks. A single malicious software can attack and infect thousands or even millions of computers in various ways simultaneously. New malware variants are increasingly becoming in million number as well as viruses, worms, ransomware, adware, spyware, and trojan horses. The danger threats are rapidly increased annually, thus the detection of the malicious program plays an essential role in the cybercrime

investigation system. The modern threats could lead to damage to the computer system easily and reduce system processing and performance. Malware analysis and classification are also important in the modern malware detection system to reduce and prevent cybercrimes. Recently, many researchers are interested in analyzing and classifying the variants of malware using the Windows API (Application Programming Interface) call sequences to model malware behavior through static or dynamic analysis. It helps incident responders understand the extent of a malware-based incident and rapidly identify additional systems or hosts that could be affected. The actionable information from malicious software analysis can help an organization more effectively to mitigate vulnerabilities exploited by malware and help prevent additional compromise.

1.2 Motivation of the Research

Due to the rising of malware in number and complex nature, the protection and detection of modern malware play a critical role in cybersecurity using machine learning (ML) methods. The information security has been needed on various government and non-government organizations, as well as businesses that rely on information technology because the massive evolution of new malware types lead a major risk to the information system and security. Traditional signatures-based antivirus software may fail to classify unfamiliar suspicious programs to corresponding categories and to identify new variants of malware programs [43].

Though there are hundreds of thousands of new malwares found every day, most of them are derived from the known families of malware. Malware obfuscation techniques include mainly packing, metamorphosis, and anti-virtual technologies. These technologies have been widely used to evade the detection of anti-virus software [43]. Thus, some of the existing techniques are developed in the traditional works to detect the threats accurately. However, these techniques lack some major drawbacks such as inaccurate detection, not highly efficient, requires a great deal of time and effort to classify the malware type, and increased computational complexity.

The malware creators mainly target the vulnerable machines to launch the attacks. The scientists and researchers have carried out an incredible effort to remedy the attacks of hackers and intruders. Regrettably, it still likes a cat-and-mouse game for both analysts or researchers and malware writers. The new kind of malware can evade

or hide the traditional detection mechanisms. Therefore, this game might never-end between researchers and malicious code writers.

Even though detection technique based on signatures are being widely used, it is not effective to identify and detect "zero-day attacks" and various "obfuscation" methods. It cannot detect the new unknown attacks. Thus, there has been a growing need for fast, and efficient detection and classification technique that can also reduce the False Positive Rate (FPR).

1.3 Problem Statements and Solutions

This research is mainly proposed to classify the malicious families for malware classification. The problem statements and the solved issues are discussed as research findings in this section.

Today’s threats have become very complex and serious in their packing and encryption techniques. Every day new malware variants are becoming increasingly in quantity together with quality by using packing and encrypting techniques. Most research [6,12,42,51] avoid the major challenge of dealing with encrypted or packed malicious samples. In this research, the experiments are conducted by making use of packed and encrypted malicious samples.

The existing detection techniques like network-based intrusion detection system (NIDS) and network based intrusion prevention system (NIPS) can provide the network administrators to monitor their organizational network's or system's status. Moreover, other host-based intrusion detection systems (HIDS) can monitor file integrity, the registry at the host or system level. However, these current methods might not detect the presence of a kernel level malicious code and categorize its functionality. The kernel is the essential component of the operating systems. Thus, the proposed system concentrates on the malware that infects the user-level and kernel-level components of the operating system. This system especially focuses on the samples from Trojan categories such as Zbot, Trojan, and Startpage, Adware and benign that used the user level or kernel hooking techniques and that have the hidden functionality.

The traditional malware classification and detection systems sometimes might fail to classify and detect the known and unknown malware variants and produce false alarms. They are the main challenges of behavior-based anomaly detection. The vital goal of this research work is to identify the malicious traces reliably, that is, to cut down

the False Positive Rate (FPR). To reduce the FPR, the accurate classification of malware is solved by proposing the Malware Feature Extraction Algorithm (MFEA).

Moreover, the most critical task in malware analysis using machine learning techniques is naming the samples in supervised mode. It is found that different antivirus vendors label malware samples differently. In most cases, the labels are inconsistent with each other. It is still challenging for AV vendors in this research field due to the fastest evolution of malicious samples annually. The related works [12,50,51] use only one AV vendor's label. For this reason, the naming technique of malware samples by using Regular Expressions has been contributed in this research. Labeling a malicious executable as a variant of a known family is important for multiple security applications. The proposed system uses the Regular Expression to define the right samples' name instead of using only one antivirus-engine.

The complex program or executables embedded with malicious intentions can infect a numerous amount of computer systems through the Internet. They can be various forms likewise virus, worm, Trojan, bot or rootkit. The occurrence of malware is speedily rising on the Internet and poses a serious threat to computer and information systems. Thus, the battle between malicious code writers and researchers is virtually a never-ending game. Therefore, the scalable and applicable two benign-malware datasets are provided for malicious classification system.

1.4 Objectives of the Research

The malicious program is increasingly becoming destructive thing for both end-users and government/non-government organizations. That is why the malware family classification research is done on this research area by conducting the dynamic analysis. In this approach, dynamic malware analysis was performed by using cuckoo sandbox. After performing the analysis, API, DLL, and PROCESS calls are extracted by using proposed MFEA. The objectives of this research work are highlighted as follows:

- To study the behaviors of malware,
- To analyze the nature and variations of malware,
- To define the right label or name of the malicious samples,
- To provide the important features for effective malware family classification,
- To extract the prominent malicious function calls by using the proposed feature extraction algorithm,

- To select distinct malicious features,
- To classify malicious or not and the malware families, and
- To provide the applicable malware datasets for malicious classification research.

1.5 Contributions of the Research

This research is conducted with multi-class and binary class classification between malware families and benign which are varied in the wild. This research develops an efficient system for classifying the malwares types using effective extracted features. Firstly, the malware and benign files are analyzed and differentiated into their corresponding families. Then, the raw feature extraction process has been carried out to identify their family categories. Data cleansing and feature selection have been conducted during the experiments, then the performance results of this research work are evaluated by using the accuracy, Confusion Matrix (CM), True Positive Rate (TPR), False Positive Rate (FPR) and Receiver Operating Characteristic (ROC) curve.

The name extraction of malicious samples labelled by VirusTotal (VT) AV-vendors in JSON reports is conducted to categorize their families using proposed Malicious Sample Name Extraction (MSNE) procedure. Then noise removing process has been performed to label or name the malicious samples. After cleansing the noise data from extracted samples' names, the specific naming process has been carried out by using the Regular Expressions (RE) theory. Thus, Naming Malicious Samples using the Regular Expression (NMS_RE) technique has also been contributed to label or name the samples in this research. Thus, this research work contributes to name the executables based on their highest score of the sample's name provided by VT using Regular Expressions (RE). It is a pattern matching technique and it is used to match the sequences of characters in the text. Most researchers have applied only one label among VT scans result from multiple AV-vendors. However, there is a conflict between the antivirus vendors to name or label the samples, for example, one malicious file might be trojan or adware or spyware or downloader depend on AV-vendors' result. Therefore, this research proposed the suitable procedure to label or name the malicious sample.

This research work contributes the Malware Feature Extraction Algorithm (MFEA) to point out the dominant features based on the generated Java Script Object

Notation (JSON) report files. The features are API, DLL, and PROCESS called by malicious and benign executables through automated analysis. After that, the data cleansing and feature representation have been performed to provide the suitable feature for classification system. Therefore, this research contributes Feature Representation for the presence and absence of features in the extracted files along with the feature reduction procedures performed on the feature space for the classification efficiency.

This research also contributes two datasets for benign and malware classification. The Benign-Malware Classification (BMC) dataset is prepared to classify between benign and malware samples for malicious or not classification. The Benign-Malware Family Classification (BMFC) dataset is created to classify the malware families and benign for multi-class classification.

To remedy the issues described in Section 1.3, this research contributes malicious classification system for classifying the malware types by using Dynamic Link Libraries (DLL), Application Programming Interfaces (APIs), and Processes features. The malware types are Adware, Trojans, Zbot, Startpage and normal programs. The major challenge of dealing with encrypted or packed malicious samples are conducted in this research. In this experiment, binary class classification system is also performed to classify the executable sample is malicious or not and the multi-class classification system is performed to classify the types or families of malicious that occurred in the wild.

1.6 Organization of the Research

This dissertation is organized with seven chapters. This chapter describes the introduction, following with a brief introduction along with the importance of malware analysis on cybersecurity, the problem areas and the solutions of the thesis, the objectives, the motivation, the contributions, and finally the organization of the thesis.

Chapter 2 discusses the recent research methodologies on malicious threats classification and detection of the cybersecurity field. The evolution of malware and different types of features for static and dynamic analysis have been described in Chapter 2 along with recent research works. Then, different feature selection approaches, and classification methods have been discussed in Chapter 2.

Chapter 3 describes different malware analysis techniques for analysts and reverse engineers by stating the useful tools and techniques. The valuable analysis

software and techniques which can be used to identify the malicious files are also described in Chapter 3. Next, the advantages and limitations of these analysis techniques have been described in this chapter. Chapter 3 describes the learning methods in malicious detection and classification. Moreover, the feature selection methods, and machine learning techniques for malware classification are described in this chapter.

Chapter 4 supports the proposed system of family classification for malicious threats. This chapter highlights the proposed Malicious Feature Extraction Algorithm (MFEA) for API sequences, Dynamic Link Library (DLL) and Process features from different categories. This chapter also presents the proposed Malicious Sample Name Extraction (MSNE) Procedure for family categorization. Then, the proposed Naming Malicious Samples using Regular Expression (NMS_RE) is highlighted in Chapter 4. The preprocessing and postprocessing phases have been conducted to remove noise data and prepare the malicious or not classification dataset and malicious family classification dataset.

Chapter 5 highlights the implementation of the malware classification system for malicious families. It describes the need of malware analysis lab environment and highlights the specification of the experiment environment for the Benign-Malware Feature Extraction (BMFE) and Malware Family Classification system.

Chapter 6 describes the datasets description and experimental results. It provides the evaluation of the experimental results by measuring the Accuracy, Confusion Matrix, TPR, FPR and ROC curves with the use of the proposed feature extraction algorithm along with the naming procedure of the malicious sample.

Finally, Chapter 7 concludes with the future work and limitations of the research works.

CHAPTER 2

LITERATURE REVIEW AND RELATED WORKS

This chapter presents the malware classification, analysis, and detection which have been done by various researchers in cybersecurity fields. The new malware number is increasing and it is difficult to identify their types because today threat actors or creators can easily modify the current version of malware into new variants by using obfuscation or encryption techniques. Therefore, malware analysis plays an important role in identifying and detecting the known and unknown malicious software. The analysis can normally be divided into static analysis and dynamic, but some researchers combine these two into a hybrid approach. The recent research trends in malware classification have been discussed in the next sections of this chapter.

2.1 Evolution of Malware

The term “malware” came from **mal**icious soft**ware**, it is a program that can covertly insert into another one and the intention is to attack and compromise the target systems in terms of CIA (confidentiality, integrity, or availability). Malware is one of the most common attacks or threats to victims' hosts such as operating systems, network systems, and IoT devices, which can lead to huge damage and disruption on government and non-government organizations [78].

The very first Internet Worm and Microsoft-DiskOperatingSystem "viruses" were designed to be annoying and harmless to both users and computers. They were easy to remove and did not have any security threat at all. At first, malware authors did not use any concealing or encrypting methods to the virus. However, things have dramatically changed lately, as malware creators are now paying attention to gain financial profits. They have started to use the malware hiding techniques from users and anti-virus software, to attack and exploit the vulnerabilities for as much as and as fast as possible. Malware authors created new hiding methods to encrypt and conceal their creations. Nowadays, cybercrimes are committed by using technology and electronic devices such as computers, mobile phones, and USB devices. The most salient artifact within cybersecurity is malicious software and computer system and BYOD are now facing the challenges of these security risks and threats. They make

malicious codes evasion techniques for AV. The popular malware that described in [57] are highlighted in Figure 2.1.

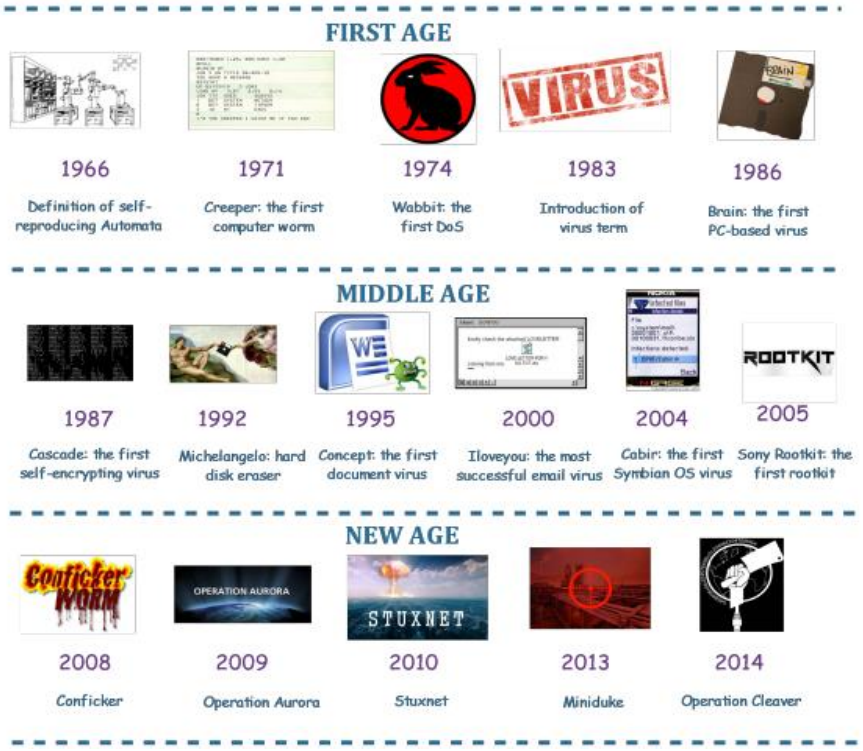

Figure 2.1 The Malware Evolutionally Up to 2014 [57]

According to the report in [90], the rates of malware infections have been rising for the last ten years. The annual increasing number of malwares is shown in Figure 2.2.

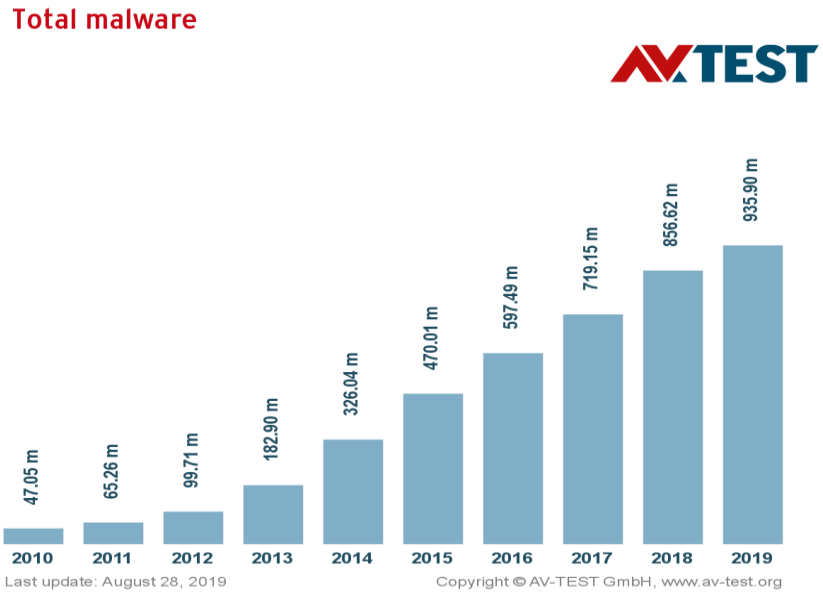

Figure 2.2 Annual Malware Increasing Rate [94]

Malware is an effective tool for attackers and hackers to infect a victim's computer and collect personal data.

2.1.1 Types of Malware

Any software that happens to pose a threat to users, computers, or network systems can be considered as malicious software. It can be various such as scareware, adware, viruses, rootkits, trojan, worms, and spyware, etc. [31]. The following subsection describes common different types of malicious software that can harm the computer system, network security, IoT devices, etc.

2.1.1.1 Virus

It can replicate by inserting multiple copies into host programs or data files and it needs user action or interaction, like that by exploring a file or executing a program. It can be divided into two categories such as compiled viruses (execution a program by the OS) and interpreted viruses (execution a program by an application) [78].

2.1.1.2 Trojan Horses

It is a common type of malware and it can disguise as legal software. It can be applied by cyber threat actors and hackers trying to gain access to users' systems. It uses the Social Engineering (SE) techniques to trick the users into loading and executing Trojans on victims' systems. After installing or executing the codes, it enables to steal user's sensitive data and monitor the computer system, and gain backdoor access to OS. Trojan can perform various actions such as delete, block, modify, copy, scan the users' data and information, and disrupt the networks or computer performance.

Trojans cannot perform self-replicating procedures like viruses and worms [100]. It can be categorized into different types the way they affect the host computer. Some of the different types of trojan are described as follows:

1. **Trojan-Banker -** to steal bank information (online and mobile banking, electronic-payment and credit-cards or debit-cards).
2. **Trojan DDoS** - to perform the Denial of Service (DoS) attack that uses the infected computers.
3. **Trojan-Downloader** - to secretly download the malicious software and install it into the computer – including backdoor, dropper, and adware.

4. **Trojan-Dropper** - to install malicious program or code – or to prevent antivirus detection by dropping a piece of malware code.
5. **Trojan GameThief** - to steal user information through online games.
6. **Trojan-IM (Instant Messenger)** - to steal the user credentials information from IM programs.
7. **Trojan Ransom** - to restrict access to the data on a victim's computer by locking the data on the user's hard disk as long as the user does not pay the ransom or bitcoin. The attacker will only restore the encrypted data after paying the money.
8. **Trojan-Spy** - to spy the victim's computer system by using the keylogger or some other sniffing tools to collect the keystrokes, screenshot or running services or applications.
9. **Trojan-Mail finder** - to collect the email information from the user's computer [100].

The above-mentioned trojan types are commonly found in our computer systems but there are still left some known trojan families such as ArcBomb, Clicker, Notifier, Proxy, PSW [100].

2.1.1.3 Worms

It is a program that replicates itself and it does not need any user interaction to infect the other computers. It can be divided into two types:

- **Network Service Worms** - It can propagate itself and/or infect other hosts by taking advantage of a weakness in the network [78].
- **E-Mail Worms** - It can infect other systems via email. It can be used to launch the Distributed DoS by consuming the network resources [84].

Worms use the various propagation techniques such as e-mail, IM, peer-2-peer (P2P) file sharing, IRC channels, Local Area Networks, and Wide Area Networks, to infect the victims' computer systems.

2.1.1.4 Backdoor

It is a malicious code that can permit the attacker to enter the system by installing itself into a computer. It can provide the intruders connect to the computer system with slight or no authentication login and execute commands on the local system

[31]. It provides the hacker to control the system remotely via command line, bash, or special console. It can be used to hide the existence of the malicious program on the victim's system after compromising the system. However, it can also be used in technical support teams legitimately for improving or providing useful information to users [57].

2.1.1.5 Rootkits

It means that to gain root-level access by leveraging the highest privilege of access to exploit a target system [62]. It is used to hide the existence of other malicious code by modifying or hooking the internal functionality of OS. It uses the kernel level and user level hooking e.g. Import Address Table, Interrupt Descriptor Table, System Service Descriptor Table, or Direct Kernel Object Manipulation hooking [61]. Hooking techniques can be used not only to exploit the attack on target but also to monitor the computer system legitimately.

2.1.1.6 Adware

Adware is an **ad**vertisement soft**ware** that can record the users' information such as user browsing history, browsing or surfing from websites, search queries, and so on, for advertisement. Sometimes it is combined with the legal software by the software inventors to gain financial profits by showing the commercial ads to users or selling the user's interests to the associated companies [57]. It can automatically play the advertisements on the browser of the user computer without user permission or desire [66]. Some people claim that adware is not harmful by nature, but it is annoying by popping up the offending adult games or pages on the user browser.

2.1.1.7 Spyware

It is installed on the user computer without the user's permission or knowledge. It can capture and exfiltrate data from a user computer to the attacker. By using the spyware, the attacker can monitor and steal the user information such as passwords or credit card numbers, the system changes on user browsers [81]. It can affect the user computer in two ways:

- stealing personal information such as browsing history, email, saved passwords, and user browsing habits

- slowing down computer resources such as frequent system crashes or freezes the computer.

2.1.1.8 Phishing

Phishing is a popular cyber-criminal activity that uses SE techniques to trick the user and gain access to user information by revealing confidential information, such as users' credentials, login passwords. One of the most popular phishing techniques is email phishing techniques by sending the embedded malicious code attachments to victims.

A brief history of some malware has been categorized into yearly according to the report, literature review from [57, 92, 97]. And it describes in Table 1. At first, the malware was created for fun and personal reputation without using the obfuscation techniques. Then later the Internet has become very popular and the number of users has increased significantly. However, users do not have enough awareness of security. Finally, malware is created mainly for financial gain, espionage, and sabotage by targeting users who do not have security awareness in the IT era. Table 2.1 shows some popular malware in the wild from the year 1966 to 2019.

Table 2.1 Emerging of Popular Malware

Year	Name	Year	Name
1966	• Self-reproducing Automata	2006	• Stration worm
1971	• Creeper (worm)	2007	• Storm worm (Storm botnet by infecting 1 to 10 million computers)
1974	• Rabbit (DoS)	2008	• Rustock • Conficker (infect 9 to 15 million MS servers)
1983	• Introduction of the term virus	2009	• Operation Aurora
1986	• Brain virus	2010	• Stuxnet (target Iranian nuclear facilities)

1987	• Cascade (self-encrypting virus)	2011	• ZeroAccess rootkit • Duqu worm
1992	• Michelangelo (Hard Disk wiped)	2012	• Flame (target cyber espionage in Middle Easten countries)
1995	• Concept (Macro virus MS word)	2013	• CryptoLocker • Gameover ZeuS
2000	• Iloveyou (e-mail virus)	2014	• Regin dropper
2001	• Sadmind worm	2015	• BASHLITE (DDoS attacks)
2002	• Smile (metamorphic virus written in assembly) • Beast (known as RAT)	2016	• Locky ransomware
2003	• ProRat • SQL Slammer	2017	• WannaCry
2004	• MyDoom worm (DDoS attack)	2018	• BitCoinMiner • Formjacking
2005	• Sony Rootkit • Zlob Trojan	2019	• Cryptojacking

2.2 Malicious Features for Classification and Detection

Malicious patterns or behavior extraction is the process of extracting or calculating the nature or behavior of malware from each sample of the dataset. For malware researches, the dataset is normally the programs written in different forms i.e. Dynamic Link Library (DLL), Executable (EXE), Component Object Model (COM) and code snippets.

Features extracted from these binary program files either through static analysis or through dynamic analysis [38]. We adopt the figure stated in [38] for different types of features based on the analysis. Figure 2.3 shows different types of static, dynamic and integrated features used for malware detection in [38].

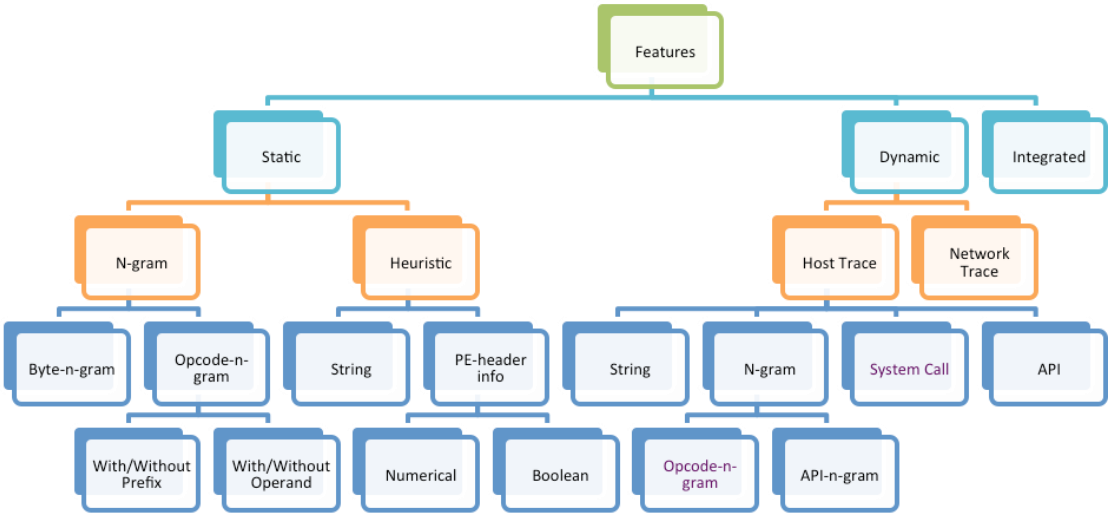

Figure 2.3 Extracted Features based on Analysis Types [38]

2.2.1 Extracted Features through Static Analysis

The effective malicious feature extraction depends on analysis type and so features differ based on analysis type. Malicious executable files analysis and extraction of related features can be done in two ways, static/signature and dynamic/behavior.

The static features are extracted without running the executable file. Reverse engineering is a common approach to extract static information from a binary file. Disassembly and hexadecimal dumping of binary file are the two main techniques to pre-process and get static features from the sample [38].

Extracting strings, import functions, meta-data, byte code n-gram, and opcode n-gram and so on, from binaries to analyze and deduce sample's destructiveness is called the static analysis [57]. The authors in [33] they extracted the sequence of common API calls by adopting the DNA sequence alignment algorithm. They discussed two ways to collect API calls from a static approach:

- Statistical analysis such as counting the occurrence of called APIs and
- ML or DM (Data Mining) approaches to the captured API information.

These APIs can also be used to create behavioral patterns in dynamic analysis.

The authors in [70] used static byte n-gram features by conducting semi-supervised learning techniques for classification. The authors used two thousand executable files for the experiment and got 0.16 FPR in [70]. In [88], the researchers used the sequences of API call as the characterizing feature from over 30000 executable files and attained over 93% accuracy. In [83], the researchers also used the static API

like in [88] extracted from normal executable 361files and virus executable 353 files. They got the 93.71% classification accuracy in their research work.

Opcode-n-gram is a sequence of the n^{th} consecutive opcode, each sequence is created with a sliding window of n over the disassembled code of the sample. Opcode-n-gram with and without operand features has also been used as features for building malware classifier [38]. In related work [79], the authors used opcode features on 300 cleanware and 500 malicious files, they got a TPR of 0.992 and an FPR of 0.53. The author in [71] also used opcodes and API calls through hybrid nature, they got an accuracy of 96.6% on a dataset of 1000 malware and 1000 goodware files. The authors in [82], IDA Pro has been used to disassemble each file and the sequences of opcode were extracted in their experiments on 200 malware and 40 benign files.

The authors in [43] divided the system into three main parts: feature extraction, decision making, and new malware detection. Then they applied information gain to reduce the extracted features into 500 features. They used seven classifiers (sklearn) to test its accuracy on malware classification [43].

In [21], the authors used the extracted API calls in their works. In [21], the authors extracted APIs from the IAT table using static analysis. They compared the extracted API sequence with another sequence and calculated the similarity between them to classify malware family. They found that malware within the same family are about 40% similar and false positive rate calculated 16%.

The authors presented an MD system for PE, based on a set of APIs calls and some TPFs features. They used the chi-square (χ^2) measure method to select the suitable APIs and TPFs. Their dataset consists of 552 executables including 338 malicious and the rest are clean, and 20% was used for the testing dataset. And they tested on 12 different malware categories. In their work, the total final APIs are over 680 and TPFs are 50. They achieved an overall accuracy of 98.27% by combining APIs and TPFs using a boosted decision tree (B-J48) classifier. In their work, they only considered non-packed (non-compressed) programs [8]. The authors in [74] used a tool hex dump to extract PE file information such as strings, API functions, and byte sequence. The accuracy was 97.11% and FPR was 3.8% on 4266 samples for binary classification in their work.

The authors in [57] stated that the analysis through static is fairly straightforward and fast. But there is a trade-off between accuracy and simplicity, this

type of analysis cannot be effective against the complex malware and may leave the main malicious behavior patterns. Additionally, the obfuscated techniques applied by malware creators likewise polymorphic, metamorphic, compression, encryption, and packing, render the analysis complexed, time-consuming and almost unrealistic. So, the researchers developed and performed the analysis through a dynamic method that is more resistant to these kinds of complicated techniques.

2.2.2 Extracted Features through Dynamic Analysis

The static investigation faces some weaknesses that mentioned the previous section. In dynamic, the malware behavior has been observed by running the executables in a virtual machine. By capturing the behavior of malware with executing the program in a safe VM, the problem of static analysis is answered by dynamic analysis. Therefore, the dynamic analysis provides more accurate analysis reports than the static and is mostly used to achieve accurate malicious detection [50].

According to [38] dynamic features can be derived from host trace and network track-based features. The activities of internal memory, files and file system, registry, hardware performance counters and status of running processes of the host are considered as host trace and used as features. The recent works that perform the classification, clustering, and detection through dynamic analysis are described as followed.

The authors in [3] proposed MalClassifier that allowed to identify the malware family based on network activity. The semantics behavior and the order of network flow sequence are used to create n-flow. The authors used the similarity measure with fuzzy to calculate the similarity degree between the flow sequence and extracted profiles. To evaluate the performance of the classifier, the authors used the datasets of ransomware and botnets' network traffics and their system obtained an f-measure of 96% with Random Forest (RF) classifier for the classification of the family.

The researchers used another way to classify the samples by capturing the system call sequences (API) for classification purposes. They built a neural network based on convolutional and recurrent neural networks to get the best features for classification. They combined the n-grams convolution with full sequential modeling to get hierarchical FE architecture. The classification accuracy was under 90% in their work. Their system achieved an average precision and recall of 85.6% and 89.4%

respectively by combining the convolutional and recurrent neural network architecture [34]. Many types of research have worked that used n-grams for MCD and the authors in [74] used n-gram firstly on Microsoft Windows malware detection.

The authors in [67], they extracted API calls from each binary file and applied the frequencies of these API to the classifier. Then, three feature sets were generated like ‘API call list', 'API arguments' and ‘API and arguments list', and each set had been tested separately. The API list obtained 0.946% TPR and 0.132% FPR with RF classifier.

The authors in [12] applied the Information Retrieval (IR) theory to classify malicious programs. They employed a dynamic analysis method to extract the sequences of API calls and system calls, providing the discriminatory features from different layers. The authors experimented total 425 samples from 10 families. They got 97.7% accuracy with the use of IR and mutual information approaches.

In [10], the author analyzed the static and dynamic data extracted from malicious and benign files to predict a test dataset for classification. They applied the RandomForest library on R-Cran (Rstudio) to select the best features from the previous ChiSqSelector’s result. Goodware oversampling had been performed in their work. RF, SVM, and NN were applied in a novel combination and their system provided not only an increase in accuracy but also minimize the training time within an accuracy of 99.60%.

The researchers proposed a classification framework to study the malware behavior dynamically using information theory and a machine learning technique concept. The authors proposed MCSF and then extracted the features behavioral patterns including duration, network attributes, API sequence frequencies and count on files read, written or created, etc. by execution the malware. Moreover, the authors selected the most promising features using information theory concepts [50]. The authors used only Kaspersky AV system for labeling the malware samples. The FPR and FNR of experiments were 0.596% and 0.016% respectively on full feature set.

The authors conducted the research using the combination approach of data mining and dynamic analysis. Their method provided the best performance in malware detection using classification via the regression method. They experimented with two datasets. In the first dataset, the correctly classified instance number was 75.8201% from the 4024 malicious programs with Regression algorithm. In the second dataset,

the correctly classified instance number was 98.321% from 3131 malicious programs. They tested 100 malware programs by adding to the new malware and compute a true optimistic ratio. A total of 88 malware programs were detected by using classification via regression [52]. The total number of tested samples was quite small in their approach.

They performed the analysis through a dynamic method to extract API calls from malicious binaries. CWSandbox and Cuckoo Sandbox were used in their research work and used 3131 malicious binaries from 24 classes. Apriori Program was used in their work to find frequent itemsets. They used the Prototype clustering method to cluster their dataset. They got F-measure 94.7% in their framework [60]. The authors in [85] used the combination of the LSTM deep learning model and the RF model to propose the ASSCA architecture. They used the information gain to select the subsequence features from the n-gram and the ACC was 0.957%.

The dynamic techniques mostly emphasized on API calls in [13,17,33,40] to represent malware behaviors. The authors in [26] also performed the classification using API sequences for three different suspicious file types and benign. The TPR and FPR were 93.2% and 6.8 % respectively in their work. The researchers in [40], the authors proposed the variants of malware classification technique based on behavior profile. The authors used TEMU to monitor malware behaviors. They captured the API calls and other information, then established a multilayer dependency chain by converting the function flow into a multi-layer behavior chain. To assess the validity and accuracy of the method, they downloaded 200 samples of twelve types from the Anubis website. To identify the malware variants similarity, similarity comparison algorithm had been used in their work.

The researchers in [58] performed classification through run time analysis using cuckoo. The total number of samples, 42,068, was used for classification, 67 % was used to train and 33 % was used to test the dataset. The authors extracted and used 151 API calls as main features, the first 200 API was used for sequence during execution. In their approach, they employed a combination of features that achieved TPR of 0.896, and FPR of 0.049 by applying an RF classifier.

In [51], the authors experimented with a 552 PE dataset with their corresponding API calls. These samples were executed in a Windows 7 virtual environment using a Cuckoo sandbox. Term Frequency Inverse Document Frequency had been applied to

select relevant 4-gram API call features. The authors used four machine learning methods for training and testing the data. They got the accuracy between 92% and 96.4%.

The authors from [29] extracted separately different features through runtime analysis, API, system libraries usage and operations. Four different classifiers and correlation-based feature selection method from WEKA tool had been applied in their work. Bigram API and API frequency approaches gave the best performance by using the RF for four datasets. The dataset C provided the best accuracy 97.33% with RF classifier.

In dynamic analysis, the behavior of malware is checked by tracing the API calls and network parameters by running the suspicious files in a simulated or VM environment. The extracted API-calls are used to detect malicious behavior through behavior or dynamic or runtime analysis method. The API-calls from different categories such as process, registry, file, and network contain the parameters, function names, and return values of an executable. This method tries to extract distinct features to categorize malicious software programs using the API-frequency and API-Sequence [33,57]. API-frequency may indicate how important an API-call for malware, while API-Sequence indicates the knowledge about how important consecutive behaviors of the malware.

This section presents the current research works that have done through dynamic or run-time analysis. Later, researchers are now working by proposing the hybrid nature for analysis and features such as hybrid analysis and hybrid features combination and feature fusion methodologies. And most of the attributes that use to detect and identify the malicious programs are API-calls which are based on the number of occurrences of API (frequency), the order of API (sequence), and system calls.

2.3 Feature Selection Methods for Malware Classification and Detection

The process of selecting the most important attributes is the main role in mining technology. It plays a significant role in improving the classifier's performance and also improving the accuracy by discarding or reforming the shape of features or attributes such as the noisy, redundant, and irrelevant from the dataset. To improve the classification rate of malicious and benign executables and to decrease the FP and FN, this stage is essential to find the best API features. Not all the extracted features can be

used in training for the following reasons.

- Consuming large memory usage
- Taking long training time for classifiers
- Producing the false alarm rate due to a large number of noisy, redundant or irrelevant features [46].

Feature selection is a central phase for the malware family classification (MFC) and malware and cleanware detection (MCD) research areas. As mentioned, one of the objectives is to choose the smallest number of features that keep the classification rate as high as possible to allow to use the minimum quantity of resources for the malware detection task [10]. The efficiency of classification was improved by implementing the attribute selection technique with the reduced number of attributes for training [13].

Table 2.2 shows some of the FS methods that have been used in recent research works on MCD.

Table 2.2 Features and FS Methods on Recent Research Works

Study	Features	FS	# of samples	Accuracy
[22] 2006	Sequences of n bytes	HFS (wrapper approach)	1512 viruses 1488 benign	93.65%
[46] 2007	Binary and assembly n-grams, LFC 1,2,4,6,8,10-grams	IG	597 clean 838 malicious	96.30%
			1,370 clean 1,082 malicious	96.15%
[49] 2011	API String	IG	1368 malware 456 clean	97.4% for MFC 94.6% for MvsC
[68] 2012	API from 6 DLLs input arguments	ReliefF	826 malicious 385 clean	98.4%
[42] 2015	n-gram (Static + Dynamic)	Tf-Idf	4,288 samples from 9 families	~96%
[8] 2016	API, TPF	chi-square χ^2	338 malwares 214 benign 12 categories	98%
[10] 2016	Static + Dynamic	ChiSqSelector	7630 malware 1818 goodware	99.60%

[6] 2018	memory access patterns	IG (50,000), then CfSubsetEval (29)	952 files for malware types, 983 files for malware families	0.784 % for malware families, 0.668 % for malware types
[51] 2018	API calls	TF-IDF	552 malicious and benign	96.4%
[85] 2018	API calls	IG	-	95.7%

In [6] the authors used memory access patterns to distinguish malicious families. The feature selection process was performed and trained by ML Models. They took 50,000 features by selecting the highest IG rank. CFS in Weka was also performed to find the best 10,000 features. However, the accuracy was 0.845% with RF classifier in their work.

In [43] the authors applied an improved information gain to reduce the extracted features into 500 features and they also applied shared nearest neighbor (SNN) to find new malware. The new malware detection accuracy was 86.7%.

The authors in [46] used information gain after extracting the n-grams to choose the top 500 features. They experimented on two different datasets: the first data set contains a collection of 1,435 executables (597 cleanware and 838 malware), and dataset2 contains 2,452 executables, (1,370 clean and 1,082 malware). IG attribute selection method was used in [36, 49, 47] and their accuracies with 98%, 94.6, and 97.7% respectively. The accuracy of hybrid model was 97.4 for both datasets n = 6,4 respectively in [46]. Tf-idf was applied in [42] to get the most relevant features.

Among the three FS methods such as filter, wrapper, and embedded methods, most researchers commonly used the filter-based approach in malicious classification and detection research areas. The filtering approach does not depend on any particular algorithm. It is very fast and computationally less expensive than the other two methods. It can be easily scaled to very high-dimensional datasets [32]. This section discusses the malicious behavior patterns and FS approaches used by malware researchers and analysts in current research works.

2.4 Malicious Classification and Detection in Machine Learning

The usage of ML techniques in cybersecurity is becoming increasingly than ever before. ML can be used in many fields and applications such as Amazon, Google, Health Diagnosis, Recognition Systems, and Facebook to improve the user experiences, purchases, suggested connecting friends socially, recommendation and recognition systems for daily life. Machine learning has been deployed in many fields such as speech recognition, computer vision, robot control, natural language processing, and other applications. And ML has a powerful ability and capability to do many things for cybersecurity. It can be used to identify the APTs which are more complex than the normal malware or threats [16]. Various ML techniques have been successfully applied to highlights the wide-ranging problems in computer and information security. ML techniques can be used in many intrusion detection systems (IDS) because it can detect new and unknown attacks. It can be applied in many areas of Information Security such as spam and phishing email detection, virus detection, and surveillance camera robbery detection [15].

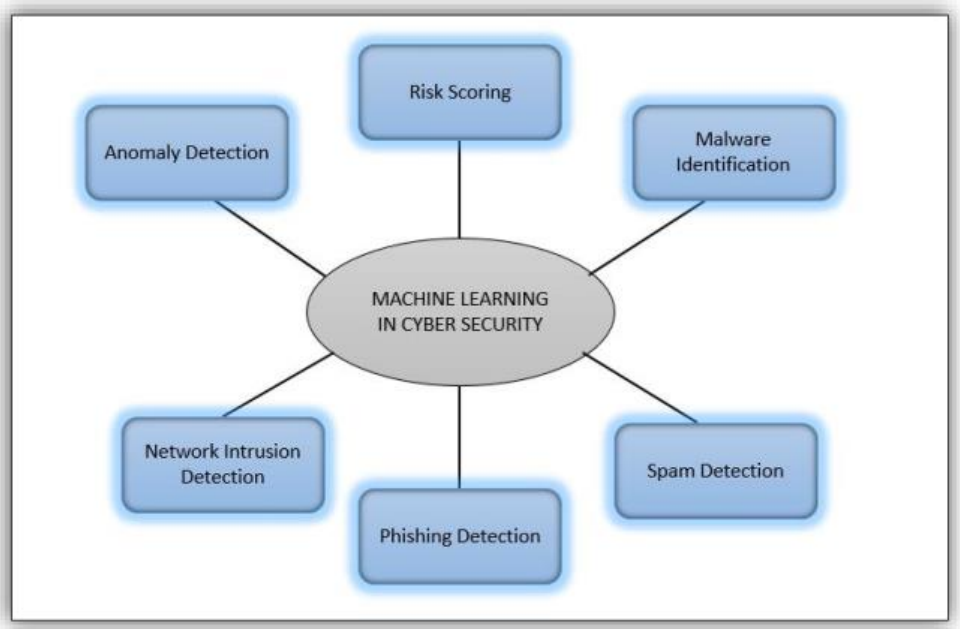

Figure 2.4 Machine Learning in Cyber Security [73]

The Network Intrusion Detection, Anomaly Detection, Phishing Detection, Spam Detection, Malware Identification and Risk Scoring Systems can be developed by using the ML techniques. Moreover, ML-based technologies ever more aid to fight large scale fraud, assess and enhance business processes, improve testing procedures and develop new solutions to existing problems.

This section provides some of the related works on malware detection and classification using ML approaches.

The authors experimented with under 5000 binary programs including clean and malicious. The hexdump was used to convert binary to Hexa files. DLL was used as the main feature in their work. They got the 97.76 percentage accuracy using multi-naïve Bayes [74]. The authors in [5] used the IG, PCA and ML methods likewise NaiveBayes, J48, and SVM on detection using static type.

They proposed an approach for analyzing the malware behavior using CWSandbox and converted malicious files by converting XML files into WEKA ARFF formats. They applied NaiveBayse, J48, BayseNet, IB1, SVM, and classification via regression algorithms using the WEKA tool [52].

They proposed a feature extraction method based on grayscale images, n-gram on opcode, and import functions. The authors used IDA Pro to extract the opcode from malware files. They constructed a control flow graph of functions by using IDA Pro. Their system classified unknown malware with an accuracy of 98.9%. They detected 86.7% new malware by applying SNN clustering [43].

To sum up the contents of this section, many authors provided classification and detection in various ways such as by extracting the code flow, byte-code, API, string, opcode based on executable files, images, pdf, doc, and so on.

2.5 Summary

This chapter presents the existing works of malware classification using Machine Learning techniques for static, dynamic, and hybrid features. This chapter discusses previous researches about feature extraction methods based on malware analysis and classification. These research efforts use different malware modeling techniques using static or dynamic features generated from malware samples. Besides, this chapter covers and discusses their modeling techniques and the attribute selection approach of the previous researches.

In this chapter, six sections have been described and Section 1 highlights the overview of malware infection techniques and Section 2 discusses the evolution of malware and describes common malware types that occurred in the wild. Section 3 further presents the signature-based so-called static-based, and behavior based so-called dynamic-based analysis tools and techniques together with recent research works. Then

the malware attribute extraction and selection approaches have been discussed for each type of analysis.

This chapter also introduces state-of-the-art analysis techniques and discussed the benefits and hindrances of each technique. It provides a list of suitable programs that can be applied in analysis, thus aiding the readers to be familiar in malware forensics and cybersecurity research fields. Chapter 3 presents the theoretical background of research work in the field of malware research. Particularly, the machine learning methods will provide for malware classification, attributes representation, and attributes selection methods.

CHAPTER 3

THEORETICAL BACKGROUND

This chapter consists of five sections: the techniques of analyzing the malware such as running or without running the executable; naming the malicious executables; extracting; selecting the attributes or features through analysis; and applying machine learning methods for classification. The subsection includes the pros and cons of malicious software analysis, malicious feature extraction and selection, classification and detection based on machine learning classifiers, and the last section provides a summary of this chapter.

3.1 Malware Analysis Techniques

Due to the growing of malicious code in the information technology and cyber world, the knowledge and understanding of new unknown malicious code or program protection is an important topic in the detection system using ML methods. There are normally two ways such as static and dynamic to carry out the analysis for detecting and finding the new malware.

The process of analyzing the software or program without executing the program is referred to as static analysis that can classify and detect the known and unknown malicious code [80]. It is the first approach for analyzing and detecting the malicious software that has been stated in [45]. In dynamic, the malware behavior has been observed by running the executables in a virtual machine or in a controlled environment.

3.1.1 Static Malware Analysis Tools and Techniques

The static malware analysis inspects the assembly code of binary file to identify the retrieve code flow and sequential instructions to identify the malicious sample without actually executing the sample [24]. The disassembler tools, IDA Pro and OlleyDbg, can be used to decompile the Windows executable file that can provide the instructions of assembly information, information about malware, and extract patterns to find the attacker's intention. The following analysis tools can be used for analysis of

portable executables (PE) files [55].

PE Studio is a GUI tool that performs malware assessments on executable files such as imported and exported function names and strings [75].

Mastiff is a framework for static malware analysis that extracts the important characteristics from different file formats automatically [58].

Peframe extracts static file properties and it can help malware analysts and researchers to detect packer, XOR, DS (digital signature), mutexes, anti-debug, anti-VM, suspicious functions and API, and more information about the suspicious or malicious files. It is a command-line tool [96].

Exeinfo PE can identify the signatures of commonly used packers [75].

CFF Explorer has the ability of content header editing [55,75].

ExifTool can extract various metadata that embedded into files and also extract some data from executable files [75].

strings2 can extract the ASCII code and Unicode encoded strings, and also extract the strings from a running process. However, it does not support GUI mode [75].

PEiD supports the GUI tool that can detect the common packers, cryptors, and compilers [55].

Dependency Walker is used to exploring functions imported and DLLs by a piece of malicious program. It constructs the DLL hierarchical tree structure that will be loaded into memory when the malware is running [31].

IDA Pro is a popular powerful disassembler tool of HexRays. It has been used by many malicious software analysts, reverse engineers, and vulnerability analysts [31].

Signsrch is a **sign**ature **s**ea**rch**ing tool and it is very useful in reversing engineering. It is easy to modify and can recognize tons of compression, multimedia and encryption algorithms It uses the signatures of cryptography for detecting the malicious encryption techniques [95].

Static analysis extracts feature directly from the byte-code or disassembled instructions, thus it is not required to run the program. The features from static analysis includes string signatures, API calls, n-grams byte sequences, syntactic library call, control flow graphs and operational code (opcode) frequency distribution [18]. The static analysis tools can find the memory corruption flaws if the source/program code of the program or software is available. The common popular analysis techniques through static for malware detection are File Format Inspection (FFI), Fingerprinting

Technique (FT), Extraction of Strings (ES), Disassembly method and Scanning with Anti-Virus (SAV).

3.1.1.1 Advantages and Disadvantages of Static Analysis

The advantages of conducting static analysis:

- can profile the code flow clarity
- can know malware author's concepts and styles easily

The disadvantage of conducting static analysis:

- might raise ambiguous results as malware writer uses evasion or obfuscation techniques for detection [24]
- a relatively complex and complicated work.

3.1.2 Dynamic Malware Analysis Tools and Techniques

The dynamic or run-time analysis method performs the running or executing the malware in a safe and controlled environment. It inspects the malware run time behavior to determine which function calls are intercepted consecutively, so-called hooking, by malware to determine an overview of malicious behavior in a virtual machine [12]. A virtual system is a virtualization software such as VMware, VirtualBox, and QEMU (Quick Emulator), etc.

The following tools can be used for dynamic analysis of portable executables (PE) files [31].

- **Process Monitor, or procmon -** to monitor file system, registry, process, network, and thread activity.
- **Process Explorer** - to monitor the running processes and show them in a tree structure.
- **Regshot -** to compare the two registry snapshots before and after the analysis.
- **Wireshark** - to capture the packet that intercepts and logs network traffic.
- **INetSim** - to simulate common Internet services.
- **Volatility** - can use it to extract injected DLLs, perform rootkit detection, find hidden processes.

The malware patterns or behaviors like API calls, DNS requests, registry keys file system, network, and processes can be extracted through dynamic or run-time analysis

by using these analysis tools. According to [57], the dynamic analysis can monitor the behaviors that described as below:

Volatile Memory: Malware can cause the buffers overflow and it can use the abandoned memory locations to gain access to the victim device. By dumping the memory and then analyzing its result can determine how malware utilizes the memory.

Registry or Configuration Changes: The registry changes that might be evidence of dynamic analysis. The suspicious files often change registry values to access the system persistently like APT threats.

File Activity: Malware can do altering, adding and/or deleting the system files. The malware's valuable behaviors or information can be obtained by monitoring file activities.

Processes/Services: Malware may also disable Anti-Virus engines to achieve its functions, and jump to other processes to install new services or obstruct analysis to get the access persistently to the system.

Network Connection: By monitoring the network activities, destination Internet Protocol address, port number and the protocol types such as TCP or UDP can be captured and analyzed to detect the interaction between malware and command-and-control (C&C) server.

API Sequence calls: It reflects the software's behavior; it is very convenient to represent and model software with its API.

The criteria described above can provide valued information about malicious programs, which is hard to be collected by other detection systems. Additionally, dynamic analysis can be automated with a developed framework and it leads to enable the large-scale analysis. However, the limitations might happen due to evolving malware characteristics versus anti-dynamic analysis such as anti-VM and anti-debugging techniques. [57]

There is another way to analyze the malicious file is using a sandbox. NIST defines the sandbox in [78]: it is a security model where applications/programs are run within a safe environment or a sandbox. It can also restrict access to system resources, such as file system and memory, to keep its applications isolated from the host's other applications. It can also provide malware incident handling and prevention. It can be reset or restore itself to a current good state every time it is initialized.

The sandboxes record the changes of the file system, registry keys, and network

traffic, then generate a standardized report format. There are common sandboxes that can leverage a quick analysis of malicious files. Sandboxes such as GFI Sandbox, Anubis, Joe Sandbox, ThreatExpert, and Cuckoo Sandbox, can analyze malware for free.

GFI Sandbox records the changes at the file system, registry level such as any Windows API calls made and network traffic [30].

Anubis used the emulation environment to execute the samples as a guest in Qemu. It can produce the detail changes of the registry, file system, process, and other activities as output from the analysis [30].

Joebox uses SSDT and EAT hooking techniques in the kernel to monitor the malware's behavior, as opposed to hooking at the user level. It can analyze the executables, DLLs, kernel drivers, Word documents, PDFs, and more [41].

ThreatExpert runs files in a simulated environment and reports the differences of file system, memory, registry, newly created processes, mutexes, network, etc. [41].

Cuckoo Sandbox is open-source and it is an automated system to analyze the samples. It can perform different file types likewise executables, emails, documents, pdf, websites, etc. And the analysis report contains the trace of API calls, the general behavior of submitted files, memory dump and analyze network traffic, even when encrypted with SSL/TLS [19].

3.1.2.1 Advantages and Disadvantages of Dynamic Analysis

The advantages of this approach are:

- able to reflect the behavior of the program accurately
- not affected by encryption, compression, metamorphosis, etc.
- insensitive to packing/obfuscation techniques [10]
- able to allow for logging vulnerable behavior in a runtime environment [7]

The disadvantages of this approach:

- is time-consuming and resource intensive than the static analysis [18]
- can provide a limited view of the features that the program could exhibit given different input values [10]
- could run destructively and can damage all information on the machine due to anti-virtualization technique [7]

It can overcome the disadvantage of static analysis on ambiguous and self-modifying code and the limitation of dynamic analysis is the inability to map comprehensive functionalities of malicious executables [12].

Although it has the power of revealing the true behavior of obfuscated software, it needs a virtual machine environment, which makes it more demanding than static analysis [37].

3.2 Malware Feature Extraction and Selection Techniques

This section contains the various feature extraction and selection methods that are applied in the recent related works. The process of feature extraction is transforming the large, ambiguous collection of inputs into features sets. The transform feature can provide an organized, more manageable subset of information. And the time required to extract the features from the input dataset depends on the feature extraction methods. The method can also affect the performance of the system in terms of efficiency, robustness, and accuracy [63].

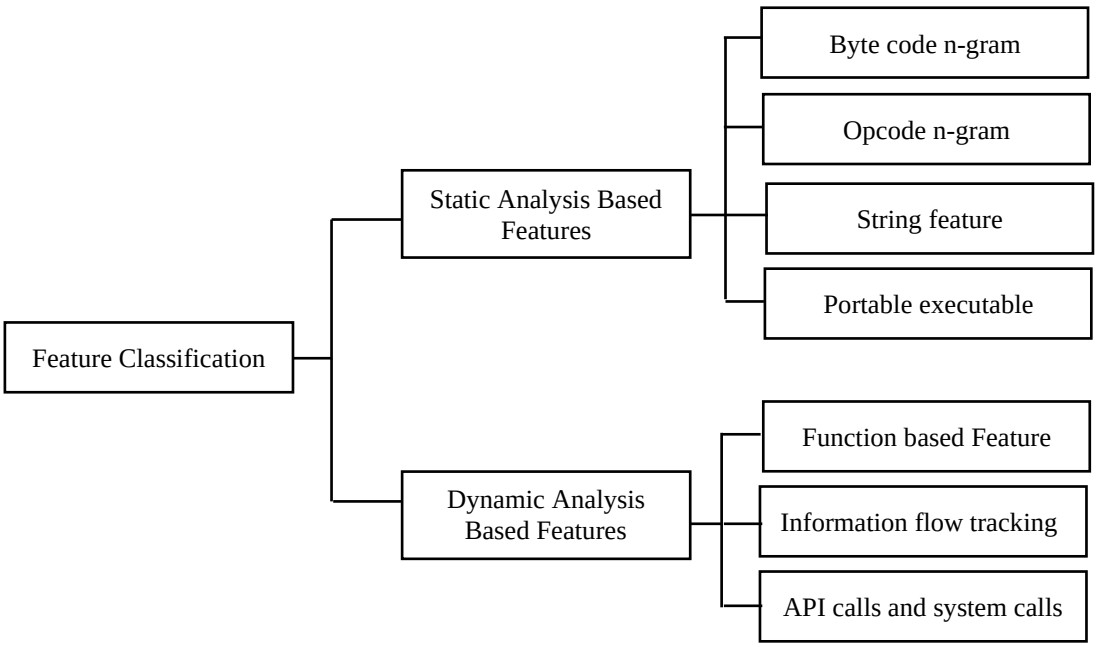

Figure 3.1 Feature Extraction Methods based on Analysis

These different feature extraction methods are shown in Figure 3.1. In [63], the authors described various feature extraction methods based on the type of analysis.

3.2.1 Static Based Features

N-gram Byte codes are bytes sequences n-grams features that pull out from malwares

used as signatures for diagnosing malware [63]. Related work [74] used the opcode and byte code feature extraction methods in their work.

Opcodes are the first part of a machine code that identifies what operation to be executed by the CPU. It can be applied as a feature in malicious software detection by testing frequency or computing the similarity between opcode-sequences [76].

N-grams are all of the contiguous subsequences of a sequence of a length N [1]. N-grams used along with opcodes and API. The authors in [86] used the Intel Pin and constructed three types of features from it, n-gram opcode (n=1,2) and system calls from dynamic traces.

Strings can also indicate suspicious file existence. The strings disclose the hacker's intention and goal because it contains the key semantic information of malicious [87].

Portable Executables (PE) features are mined through static using structural information of PE. These meaningful features show that the file was infected or manipulated to perform malicious activity [63]. The attacker's intension can be guessed easily from the information of the file format.

Besides these features, Binary block comparison (BBC), API functions, and CFG construction are also carried out to extract the static features.

Binary block comparison (BBC) – In BBC, the binary file is disassembled and it is divided into several blocks based on the function call and instructions structure. Then, to find similarities, the blocks are compared from other known malicious files by using the Hungarian algorithm [48].

Control Flow Graph (CFG) – In [14] and [39], the authors extracted the information of API calls from the Control Flow Graph (CFG) of disassembled files. Then the transformation was performed from CFG into the grams of API calls. These grams were converted into feature vectors for training the classification algorithms. The K-fold cross-validation and Weka2 were used to train and test datasets. True Positives, True Negatives and accuracy were used to measure the performance.

API function – The researchers in [69] extracted the information of API call from the import address table (IAT) in the header of the binary. The authors applied the Fisher score to rank the API calls that often appeared in malicious and benign binary files. The authors in [89] used the objective-oriented Association mining (OOA) based classification on API call information extracted from Portable Executables (PE).

3.2.2 Dynamic Based Features

Due to the limitations of static analysis, the evolution of dynamic analysis research has been carried and observed. The following features can be extracted through dynamic analysis according to Figure 3.1.

Function-based features are extracted through behavior analysis of software. These functions exist in a file for execution and utilize them to produce various attributes representing the file. Dynamically analyzed function calls including system calls, API calls, instruction sets, information flow tracking, their parameter passing, etc. [63]. The authors described the sequence of API features through dynamic analysis in [67, 30, 40, 17, 33, 27].

Information flow tracking (IFT) was used in [93] and the authors presented a system called Panorama. It used the system-wide IFT to track the flow of critical information based on taint graphs.

Dynamic-link libraries (DLLs) are the libraries files and they use to share the code among multiple applications. It is an executable file but it doesn't run alone, and export functions can be used by another application. The static libraries were the standard preceding to the use of DLLs, and they still exist, but they are much less common than DLLs. The benefit of using DLLs is that the running processes can be shared with the memory used by the DLLs. For a static library, if a library is used by two different running processes, it would raise the memory usage into double, because it will be loaded into memory twice.

An additional benefit is that it does not need any redistribution. When a user needs to distribute an executable, the user can use the known DLLs on the host OS no need to redistribute them. By doing this the software creators and malware inventors can decrease the software size [77].

The process is a program being executed by Windows. It consists of at least one or multiple threads that are being executed by the CPU. Malicious software can create a new or modify an existing process to execute the program outside of the current one. By tradition, the malware contains its own independent process, however, the newer malware commonly executes its own code as part of another process [77].

Application Programming Interface (API) is a set of methods and procedures which allow accessing the features of data of an OS, application or another service. In an operating system context, API calls that are related to a specific task are grouped and

called as DLL or library. After accessing the DLL information from the import data directory, the necessary API calls specific to the PE file can also be extracted. API is one level closer to OS system call than DLL, hence the list of all API calls by a PE file can be very discriminative to identify the real intentions of a PE file. API calls information can also be used to create a feature set and so many of previous works have used this information as features [38].

3.2.3 Feature Selection Techniques

Feature or attribute selection methods are used for reducing the dimension size of a dataset. Some of the features that are not effective in the analysis are removed from the data set. The efficiency of classification can be improved by implementing the attribute selection technique with a reduced number of attributes for training. Selecting an important attribute is one of the essential techniques in data mining especially during the data processing. The main role of this phase is to improve the classification performance as well as improving the detection accuracy by selecting the prominent features from the dataset.

The extracted raw data is hardly appropriate to apply to mining or learning algorithms. It usually needs the preprocessing steps such as to remove undesirable and inappropriate information and to represent it properly for learning algorithms. To reduce the processing time and to improve the classification performance, this research investigates two different feature selection methods.

The ML library for sklearn has been used for feature selection. Chi-squared test (χ^2), and Principal Component Analysis (PCA) methods are applied to select the best malicious features.

3.2.3.1 Chi-Square (χ^2)

Among the three FS methods such as filter, wrapper, and embedded methods, most researchers commonly used the filter-based approach in malicious classification and detection research areas. The filtering approach does not depend on any particular algorithm. It is very fast and computationally less expensive than the other two methods. It is easy to scale to very high dimension datasets [21]. So, the proposed system applies the χ^2 method from the filtering approach. The chi-square distribution (χ^2) is a distribution of theoretically or mathematically which has wide applicability in

statistical work. The term ‘chi-square' is used the Greek letter ‘χ' to define this distribution. It is seen that the elements on which this distribution is based are squared so that the symbol χ^2 is used to denote the distribution.

The χ^2 method evaluates the features individually by measuring their statistics with respect to the classes. It is a non-parametric statistic test to determine if the two or more classifications of the samples are independent or not. It requires to discretize the numeric attribute into several intervals using an entropy-based discretization method [44]. The chi-square statistic (χ^2) is defined as in [91]:

$$\chi^2 = \sum_i \frac{(O_i - E_i)^2}{E_i} \qquad \textbf{Equation 3.1}$$

where O_i is the number of observed frequencies i, and E_i is the number of expected frequencies i, and i runs from 1, 2, …., n.

It is a statistic approach and very effective for the feature selection process. The proposed system chooses χ^2 to select the feature because it can handle the multi-class data with excellent performance. The proposed system used the implementation of χ^2 from the sklearn ML library with python.

3.2.3.2 Principal Component Analysis (PCA)

PCA is a popular dimensionality reduction technique. It is an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component (PC)), the second greatest variance on the second coordinate, and so on [65]. It is widely used in image processing and attribute extraction and reduction for malware detection in security [54,42].

The goal lines of PCA are discussed as follows:

- extracting the most significant data from the dataset,
- compressing the size by storing the only vital data,
- simplifying the dataset description, and
- analyzing the structure of the observations and the variables.

PCA calculates the new PCs that are attained as linear combinations of the original variables to accomplish the goals that described above. It is used to visualize and explore high dimensional data sets. It reduces a set of correlated, high dimensional variables to a lower-dimensional set of linearly uncorrelated synthetic variables called

principal components (PC). It reduces the dimensions of a data set by projecting the data onto a lower-dimensional subspace [20]. The detail explanation of PCA has been described in [28].

The standard PCA can be summarized into six steps and described as follows:

(i) Determine the covariance matrix of the normalized d-dimensional dataset.

(ii) Determine the eigenvectors and eigenvalues of the covariance matrix.

(iii) Sort the eigenvalues in descending order.

(iv) Select the k eigenvectors that correspond to the k largest eigenvalues where k is the number of dimensions of the new feature subspace.

(v) Construct the projection matrix from the k selected eigenvectors.

(vi) Transform the original dataset to build a new k-dimensional feature space [25].

3.3 ML Techniques for Malicious Family Classification

Machine learning can be separated into supervised learning and unsupervised learning. The goal of supervised learning is to get the correct output against the input data. In contrast, the goal of unsupervised learning is to find the regularity from input data [68]. A learning algorithm must use pairs of feature vectors and their corresponding target labels to induce the values of the mapping function's parameters that produce the best classifier, as measured by a particular performance metric. In binary classification, the classifier must assign instances to one of the two classes (e.g. predict whether or not a patient has a particular disease.

In multiclass classification, the classifier must assign one of several labels to each instance. In machine learning, multi-class or multi-class classification is the problem of classifying instances into one of three classes or more. While some classification algorithms allow more than two classes to be used naturally. However, these things can become polynomial classifiers with a variety of strategies. The multi-class classification should not be confused with multi-label classification, which must predict multiple labels for each instance. In multilabel classification, the classifier must assign a subset of the labels to each instance [20]. ML has a powerful ability and capability to do many things for cybersecurity. It can be used to identify the Advanced Persistent Threats (APTs) and zero-day attacks which are more complex than the normal malware or threats. And it can be used in many intrusion detection systems

(IDS) because it can detect new and unknown attacks. It can be applied in many areas of Information Security such as spam and phishing email detection, phishing website detection, and virus detection. Malware family classification has been performed on our dataset by evaluating the performance metrics as shown in Figure 3.2. To classify the malicious and benign software, the proposed system used the three ML methods such as Random Forest, SVC, and k-Nearest Neighbor from the sklearn ML library.

3.3.1 Random Forest (RF)

It is an ensemble method that combined the multiple decision trees. It is a non-parametric approach, the interpretable, efficient, and high prediction accuracy for different data types. It can handle missing values, continuous, categorical, and, binary so it is suitable for high dimensional data modeling. RF can overcome the problems of overfitting and no need to prune the tree because of the bootstrapping and ensemble scheme [59]. It provides a unique combination of prediction accuracy and model interpretability among popular machine learning methods. It is able to achieve accurate predictions and better generalizations due to the utilization of random sampling and ensemble strategies in RF [59].

It is not very sensitive to the parameters used to run it and it is easy to determine which parameters to use as an advantage [9]. However, RF has a limitation when using it for regression. It is not possible to predict beyond the range of the response values in the training data due to the way of constructing the regression trees [23]. For parameter turning, the n_estimators=250 and random_state=100 have been used for classification experiments. The number of estimators means the number of trees in the RF classifier. If the random_state is the int, random_state is the seed used by the random number generator.

3.3.2 K-Nearest Neighbors (kNN)

It is learning or non-generalizing learning based on instances. It is instance-based learning and also known as a lazy learner. The lazy is called not because of the simplicity of its appearance, but because it doesn't learn a discriminative function from the training data but memorizes the training dataset instead. It is a sub-category of the non-parametric approach [64]. It doesn't try to construct the general internal model, but simply stores the training data instances. Classification is calculated from a simple

majority vote of the NN of each point: a query point is assigned the data class which has the most representatives within the NN of the point. It implements the learning based on the NN of each query point, where the integer value of k is specified by the user [56]. There are many distance functions, e.g. Euclidean distance, Manhattan distance, Minkowski distance, cosine similarity measure, and correlation. Minkowski distance has been used in this research work.

The performance of classifier mainly depends on the value of k and also depend upon the distance metrics that applied in the classifier. The best choice of k-value depends on the data. The effect of noise reduces when the value of k is larger on the classification, however, it makes the class boundaries less distinct. A good choice of k value can be nominated by using heuristic techniques, for example, Cross-Validation (CV). The value of k =3 provides better performance results in this experiment. The accuracy of a classifier can be cruelly decreased by the presence of noise or unrelated features, or if the feature scales are not consistent with their importance. By selecting or scaling the features might improve the classification.

3.3.3 Support Vector Classification (SVC)

Support Vector is used for both Regression, Classification, and other learning tasks. The researchers developed a library for Support Vector Machines (SVMs), Libsvm library package. Libsvm supports the Support Vector Classification (SVC) for two-class and multi-class [11]. It is supported the learning tasks that described as follows:

1. Support Vector Classification (SVC for two-class and multi-class).
2. Support Vector Regression (SVR)
3. One Class SVM.

It is based on the concept of decision planes that define decision boundaries. The hyperplane, decision plane, separates between a set of objects having different class memberships. The classification is performed by discovering the hyperplane that maximizes the margin between the two classes with the help of support vectors.

The learning of the hyperplane in SVM is done by transforming the problem using some linear algebra i.e. using kernel above is a linear kernel which has a linear separability between each variable.

For higher-dimensional data, other kernels are used as points that cannot be classified easily. The Kernel SVM takes in a kernel function in the SVM algorithm and transforms into the required form that maps data on a higher dimension which is separable.

Types of kernel function are described as follows:

1. Linear SVM:

$$\langle x, x' \rangle \qquad \textbf{Equation 3.2}$$

2. Polynomial kernel, the degree of the polynomial should be specified. It allows for curved lines in the input space.

$$(\gamma \langle x, x' \rangle + r)^d \qquad \textbf{Equation 3.3}$$

where *d* is specified by keyword degree, *r* by coef0.

3. Radial Basis Function (RBF) Kernel, it is used for non-linearly separable variables. For distance metric squared Euclidean distance is used. Using a typical value of the parameter can lead to overfitting our data.

$$exp\,(-\gamma \parallel x - x' \parallel^2) \qquad \textbf{Equation 3.4}$$

γ is specified by keyword gamma, must be greater than 0.

4. Sigmoid kernel, similar to logistic regression is used for binary classification Kernel trick uses the kernel function to transform the data into a higher dimensional feature space to make it possible to perform the linear separation for classification.

$$(\tanh(\gamma \langle x, x' \rangle + r)) \qquad \textbf{Equation 3.5}$$

where r is specified by coef0.

The fit time complexity is more than quadratic with the number of samples which makes it hard to scale to a dataset with more than a couple of 10000 samples [56]. The main advantages of SVM are described as follows:

- Efficient results in high dimensional spaces
- Helpful wherein the quantity of dimensions is higher than the quantity of data samples
- Efficient usage of memory as SVM uses only a subset of training points in the decision-making function

- Different kernel functions can be used for the decision function. Common kernels are available, but the user can also develop custom kernels [4].

The disadvantages of SVM are described as follows:

- SVM is not suitable for large datasets because of its high training time and it also takes more time in training compared to other classifiers.
- It works poorly with overlapping classes and is also sensitive to the type of kernel used.

For malware classification experiments, the parameters turning has been performed for C and kernel parameters. C is the penalty parameter of the error term where C=10 has been used in this experiment. It controls the trade-off between smooth decision boundary and classifying the training points correctly. The linear kernel has been used for model training and testing the malware classification datasets.

3.4 Performance Evaluation Metrics

The binary classification is a predicting from one of two classes, for example (e.g. “black” or “white”, “dead” or “alive”, etc.), Multiclass problems involve classifying something into one of N classes (e.g. “red”, “white” or “blue”, etc.). The performance of ML classifiers has been evaluated using Confusion Matrix (CM), Accuracy (ACC), TPR, FPR and Receiver Operator Characteristics curves (ROC).

Confusion Matrix (CM). It is a popular way to describe a classification model. CM can be formed for binary and multi-class classification models. It has been formed by comparing the predicted class label of a data point with its actual class label. After comparing the whole dataset repeatedly, the comparison results are formatted in a matrix form. This resultant matrix is the confusion matrix [72]. And figure 3.6 describes the typical structure of a CM.

The Figure 3.3 provides the Confusion Matrix Structure of multi-class classification. The diagonal values are the TP number and the row of its own except TP number is FN number. The column of its own class except TP number is the FP number. Then the rest are the TN number of the classification.

1. True positive (TP) is the number of predicted malware samples correctly classified as malicious.
2. True negative (TN) is the number of predicted benign samples correctly classified as benign.

3. False positive (FP) is the number of predicted benign samples incorrectly classified as malicious.
4. False negative (FN) is the number of predicted malware samples incorrectly classified as benign.

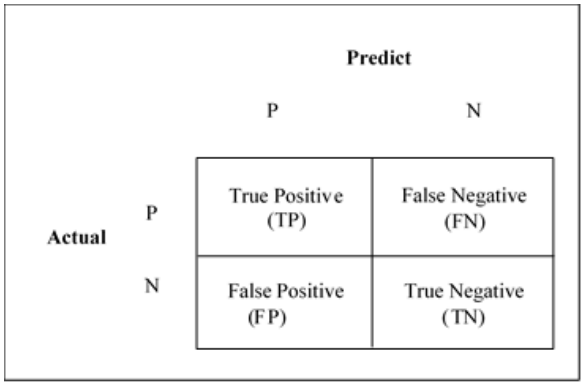

Figure 3.2 Confusion Matrix (CM) Structure

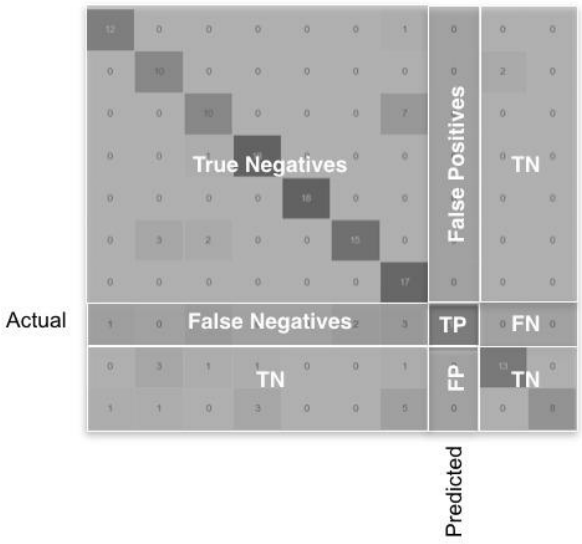

Figure 3.3 Confusion Matrix for Multi-class Classification

Accuracy (ACC). It is a common evaluation method of classifier performance. It is used to define as the percentage of the overall accuracy of correct predictions. For binary classification, the accuracy can be calculated from the following formula [72]:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \qquad \textbf{Equation 3.6}$$

True Positive Rate (TPR). It is also known as sensitivity or Recall and it is used to identify the relevant data points with percentage. It is defined as the number of instances

of the positive class that were correctly predicted [72]. It is also called a hit rate, coverage, or sensitivity. The value of Recall can be computed as follow:

$$TPR = \frac{TP}{TP + FN} \qquad \textbf{Equation 3.7}$$

False Positive Rate (FPR). It is known as false alarms or (1 - specificity), determining the total number of incorrect positive predictions among all negative samples in the dataset. It can be calculated from the following formula:

$$FPR = \frac{FP}{FP + TN} \qquad \textbf{Equation 3.8}$$

Receiver Operating Characteristic (ROC). It can be used for both binary and multiclass classifiers. TPR and the FPR of a classifier are used to plot the ROC curve. TPR is also called recall or sensitivity and it is the total number of correct positive results, predicted among all the positive samples in the dataset. FPR is also defined as (1 - specificity) or false alarms, determining the total number of incorrect positive predictions among all negative samples in the dataset [72].

The ROC curve is a valuable tool for several reasons. The curves of different models can be compared directly in general or for different thresholds. The area under the curve (AUC) can be used as a summary of the model skill. ROC Curves summarize the trade-off between the TPR and FPR for a predictive model using different probability thresholds.

ROC vs. Precision-Recall (PR) Curves can generally be used as described followed:

- If the dataset has nearly equal numbers of observations per class, ROC curves should be used.
- If the dataset has a nature of imbalance class, PR curves should be used.

3.5 Summary

This chapter describes different malware analysis techniques for analysts and reverse engineers by stating useful tools and techniques. The analysis tools and techniques that can be used to detect malicious files are also described. Moreover, this chapter presents the different malware feature extraction techniques based on the analysis types. Then, the feature selection methods that are applied in the malware classification system also described in Section 3.2. The machine learning techniques

for malicious or not classification and malware family classification are also discussed in Section 3.3. Section 3.4 describes the evaluation metrics of the malicious software classification for this experiment.

CHAPTER 4

THE PROPOSED MALWARE FAMILY CLASSIFICATION SYSTEM

This chapter provides a classification system between malicious software families and clean software. Nowadays, new variants of malicious software are very operative and active to evade anti-virus (AV) and anti-malicious applications. The writers who create the malware are using the same attack methods to generate new variants in a short duration. Thus, classifying the malicious executable families is an essential research topic in cybersecurity and cyber-crime incident handling system. The classification system aims to find traces of malicious, to reduce the FPR, which is the most important challenge in these research fields.

This chapter highlights the proposed Malicious Feature Extraction Algorithm (MFEA) for API, Dynamic Link Library (DLL), and PROCESS sequences from different categories. This chapter also presents the proposed Malicious Sample Names Extraction (MSNE) Procedure and Naming Malicious Samples using Regular Expression (NMS_RE) technique for family classification. The overall system flow of the proposed family classification system has been described in Figure 4.1. The benign and malware classification system flow compose of seven different steps:

1. Collecting the executable files
2. Executing and analyzing these files, and generating the analysis JSON reports using the cuckoo sandbox
3. Preprocessing
    (i) Extracting the name of malicious sample using proposed MSNE name extraction procedure
    (ii) Cleansing Noise Data
    (iii) Naming Malicious Samples using RE Technique (NMS_RE)
4. Extracting Effective Malicious Features using Proposed MFEA
5. Postprocessing
    (i) Data Cleansing
    (ii) Applying N-gram
    (iii) Representing and preparing the Malware and Benign Dataset

6. Selecting Malicious-Benign Features
7. Applying ML algorithms and classified malware into their families.

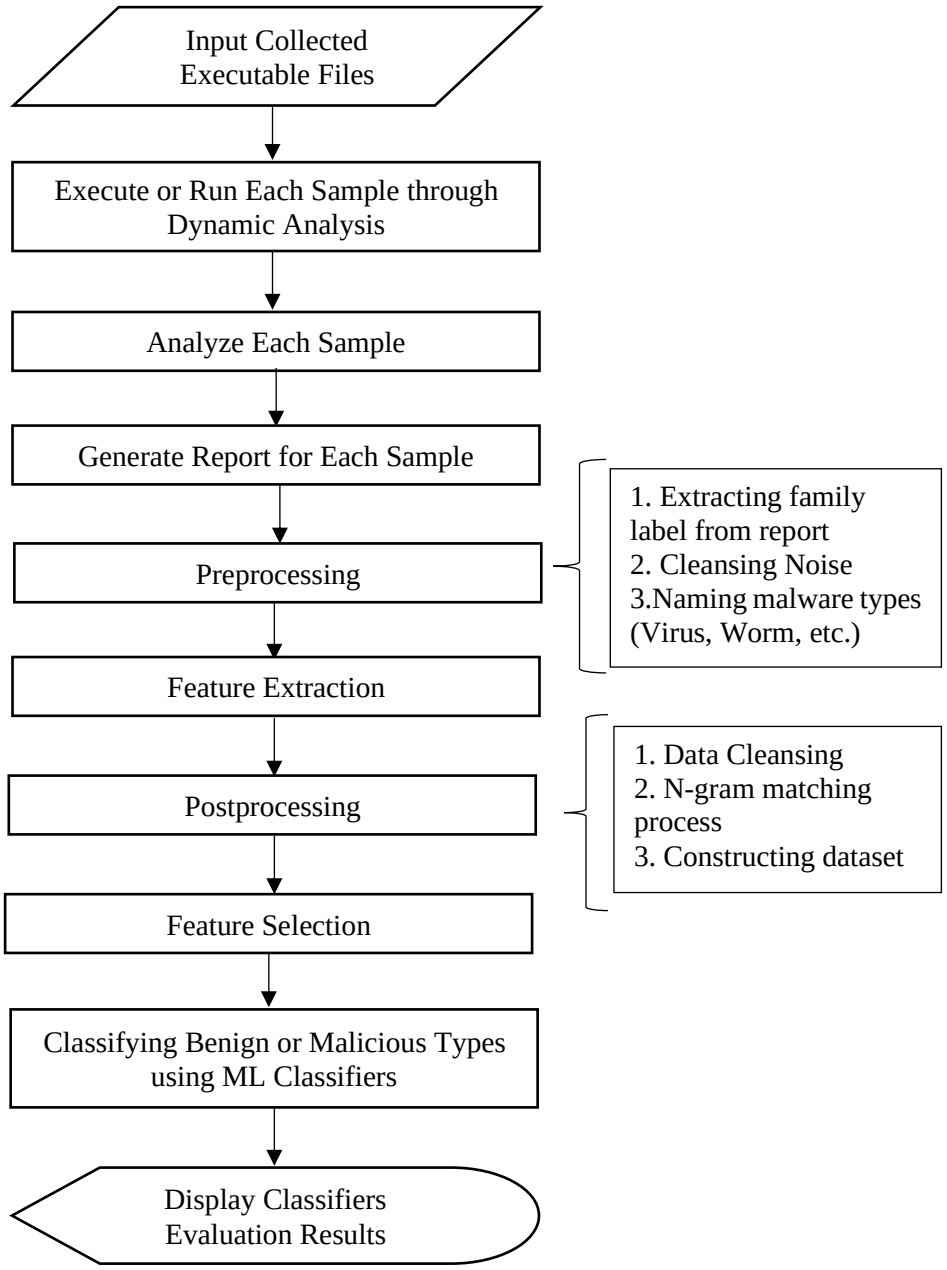

Figure 4.1 Benign-Malware Family Classification System Flow

The classification system provides two different datasets by experimenting three different types of feature approach. The approach 1 only uses the API features. The combination of API, and DLL features is called approach 2 (API_DLL). The approach 3 contains API, DLL and Process features (API_DLL_PROCESS) approach. These features are extracted from different categories such as system, registry, crypto, services, file, resource, network, and so on.

4.1 Gathering Malicious and Benign Samples

Before performing the executable files analysis, the malware and benign samples are collected from the online websites. There are generally two different ways to collect the samples. The first way is using the honeypot to collect suspicious files. The second way is downloading the malicious samples from online websites.

The honeypots are implemented to capture the threats by using the emulated and/or real vulnerabilities, or weaknesses, e.g. by assigning the easily guessable SSH password or using telnet or guessable login passwords. The malicious samples can be collected by using or creating this technique. The use of honeypots is to capture the threats information as much as possible. The honeypots, e.g. Mwcollectd, Dionaea, or Nepenthes, is a good way to capture the suspicious threats likewise botnets, DDoS, DoS, and worms.

Another way is downloading malicious samples from websites. There are many websites to download the samples for analysis. The malicious samples have been collected from the virusshare [99] website over 40000 samples and tested in this research work. The samples from different categories have analyzed like Trojan, Adware, Backdoor, Virus, Downloader, Worm, Zbot, Startpage, Ransomware, Hupigon, Installmonster, and Onlinegame, etc. These samples are downloaded from the virus share website in this research work. The clean or benign samples are downloaded from the following websites:

- http://nirsoft.net/
- http://321download.com
- http://aplusfreeware.com/
- http://completelyfreesoftware.com
- http://freeware.intrastar.net/
- Executables from windows program files

4.2 Executing, and Analyzing Malicious Samples and Generating Reports

Malicious software analysis is a process of finding the malware behaviors, and findings the attacker's goals, or intention. The malicious software analysis contains a complex procedure in its analysis or activity. Likewise, in forensics, debugging, disassembling, and reverse engineering, these types of activities take lots of time to

process the analysis. The analysis aims to understand how malware behaves so that the researchers or analysts can protect or guard the organization or system by preventing malware attacks.

There are generally two ways for analyzing the malicious software used by the researchers or analysts such as static/code analysis and dynamic/behavior analysis. The behavior of malware can understand quickly and detail by using these techniques. Moreover, the risks and purposes of malware's behavior can know during analysis. The challenge in static analysis is modern malware poses a complex nature that is implemented using anti-debugging systems.

Behavior or Dynamic analysis performs the execution or running the malicious programs and observes the changes when malware is being executed. The infecting can occur when malware is executing or running in the computer and it can be very risky. It could lead to damage to the computer systems by modifying or deleting the files, changing the registry, stealing sensitive information, and so on. The advantage of carrying out the analysis through dynamic is that the analysts can completely recognize how malware acts or works.

To execute and analyze the malicious samples, cuckoo sandbox has been used in this system. The malicious files, such as Windows executable files, and DLL, are monitored in real time by executing them, instead of executing these files manually. The analysis environment is setup to run the malicious files automatically in an isolated and safe virtual environment and then capture the analysis reports comprehensively. After executing and analyzing the malicious files, analysis reports generating has been performed by using JSON format. The analysis is processed in a clean and separated virtual environment every time for each analysis using a clean snapshot of the VM.

The JSON reports contain many analysis results such as the API calls traces, file system information for example created-files, deleted-files or downloaded-files, network information, registry information and so on.

4.3 Preporcessing

This section performs three main parts such as extracting the name of the malicious sample provided by VirusTotal [98], cleansing the extracted raw data, and naming the malware by using the most common name used by multiple antivirus vendors. The Preprocessing here means the step before performing the Malicious

Feature Extraction (MFE) step and the Postprocessing also means the step after MFE step. The contribution of choosing the suitable family name or label exists in this phase of the proposed approach. The following Figure 4.2 describes the step by step procedures for extracting malicious samples' names and cleansing noise data.

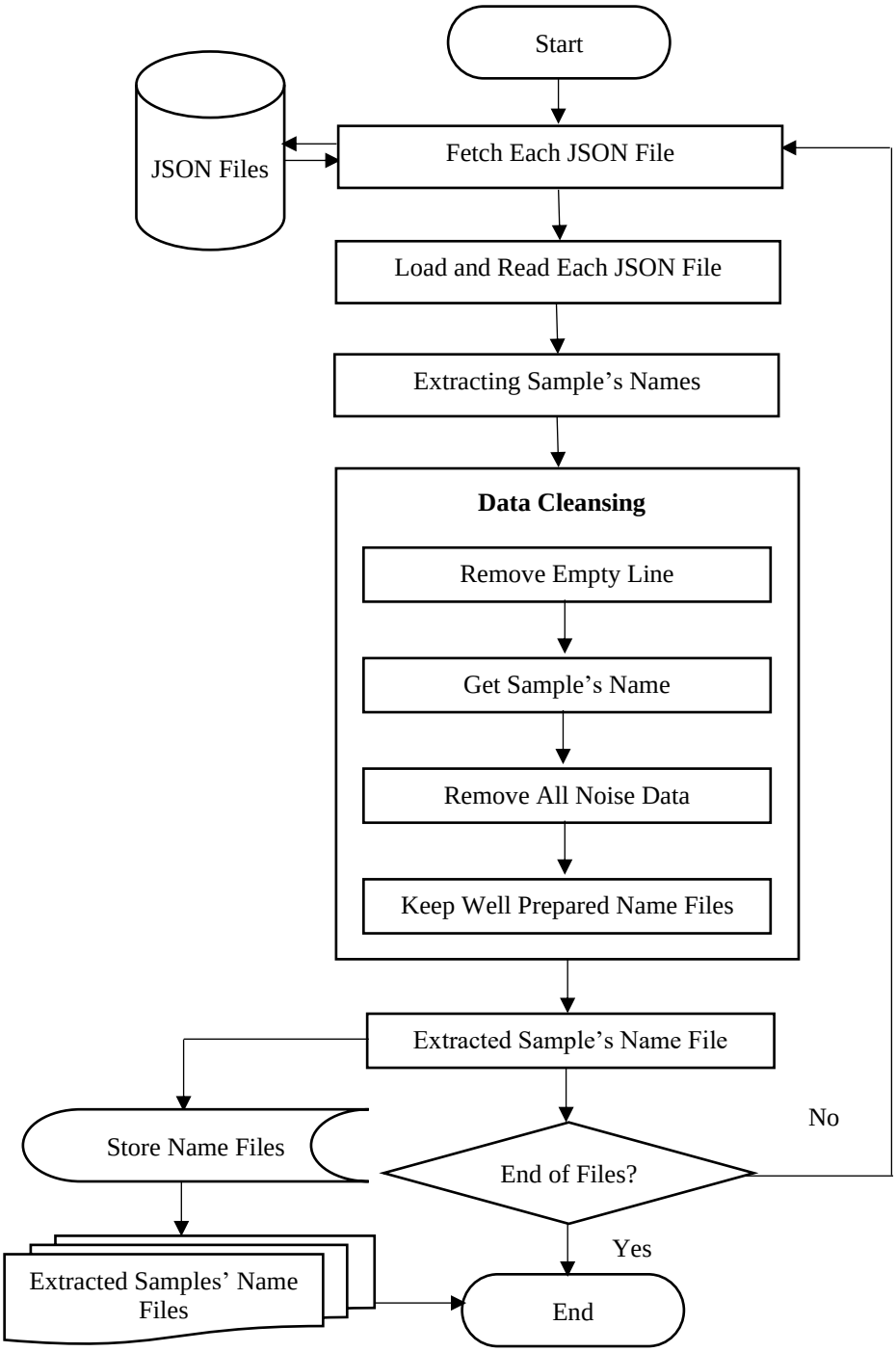

Figure 4.2 The Process of Extracting Malicious Sample Name and Cleansing Noise Data

4.3.1 Extracting Malicious Sample Name

The contribution of choosing the suitable family name or label exists in this phase of the proposed approach. As the new variants of malicious executable files are increasing, there is a conflict between the antivirus vendors to name or label the samples, for example, one malicious file has a different label or family name as trojan or adware or spyware or downloader. According to the analysis report files, the names of samples can get by using the keyword "result" from cuckoo JSON.

Malicious Sample Name Extraction (MSNE) Procedure

Input: JSON reports

Output: extracted sample names with text files f_i

```
1: begin
2:      if (JSON ≤ JSONs)
3:          for line in JSON
4:              result = line.find ("\"result\":")
5:              name = line[result:]
6:              print (name)
7:          end for
8:      end if
9: end
```

4.3.2 Cleansing Noise Data for Extracted Malicious Sample Name Files

After processing the malicious sample name extraction, raw data cleaning has been performed in the proposed system. All unnecessary data has been removed using the following steps as conditionally:

Data Cleansing for Extracted VT Labels

Input: extracted malicious samples' names files f_i

Output: raw labels or names files l_i

1. Remove the empty line, if there is an empty line
2. Get the names of samples
3. Remove all noise data
4. Keep Well Prepared Name Files

The procedure number from lines 1 and 3 in the data cleansing procedure has been

performed after applying the RE to name the samples.

4.3.3 Naming Malicious Sample using RE Technique (NMS_RE)

It is most critical task in malware analysis using machine learning techniques in supervised mode. It is found that different antivirus vendors label malware samples differently. In most cases, the labels are inconsistent with each other. Thus, the naming of malware samples has been contributed in this research.

After cleansing the noise data from extracted sample's name files, the naming process has been carried out by depicting the Figure 4.3. Therefore, this research contributes to label or name the executables based on their highest score of the sample's name provided by VirusTotal (VT) using Regular Expressions (RE).

Naming the Malicious Samples by using RE technique (NMS_RE)

```
Input: raw labels or names files li
Output: family names for malicious samples
1: begin
2:      frequency = {}
3:      if (l ≤ li)
4:          match = re.finall(r'\b[a-zA-Z]{2,15}\b', l)
5:          while (word € match)
6:              count = frequency (word, 0)
7:              frequency[word] = count+1
8:              frequency_list = frequency.keys()
9:              while (words € frequency_list)
10:                 count_word = frequency[words], words
11:                 print (count_word)
12:                 sort the count_word to get the highest majority score of VT name
13:                 choose the max_number of words as labels
14:             end while
15:         end while
16:     end if
17: end
```

Figure 4.3 Naming Malicious Samples using RE Technique (NMS_RE)

RE is a useful pattern matching technique for characters, or sequences of characters in the text. It allows the developers and users to find desired characters or words and to

replace these characters with something that the users preferred [13]. Figures 4.3 provides the procedure of naming the malicious samples using proposed NMS_RE techniques. According to the analysis report files, the labels of samples can get by using the keyword "result" from cuckoo JSON. RE has been used in this proposed system to choose the family of malicious programs and the number of characters for the sample's name is limited to 2 to 15 words. After applying the RE, the number of word or frequency of word found in the raw labels' files are stored in an array. Then the sorting step is performed and choose the name with highest score from them. The final result text files contain the number of occurrence and sample name of the single executable file.

Then the samples are categorized into their corresponding family. The key role is to find the family label that gives both the best detection rate and the most precise labeling [58]. This proposed system performs malware classification with 5 different families such as Clean, Adware, Startpage, Zbot, and Trojan.

4.4 Extracting Malicious Features using Proposed MFEA

The malicious features (categories from network, system, file, process, and registry, etc.) are extracted by using the proposed malicious feature extracting algorithm. The attributes or features are extracted according to the called API, DLL, and Process by malware.

Application Programming Interface (API) - The APIs from Windows mostly concerns itself with the interaction between the Operating System (OS) and the application. For communication between the different Windows applications among themselves, Microsoft has developed a series of technologies alongside the main Windows API. Malicious software uses these API to accomplish the infection process.

Dynamic-link library (DLL) - A dynamic-link library (DLL) is a module that contains functions and data that can be used by another module (application or DLL). A DLL can define two kinds of functions: exported and internal. The exported functions are intended to be called by other modules, as well as from within the DLL where they are defined. Internal functions are typically intended to be called only from within the DLL where they are defined. Although a DLL can export data, its data is generally used only by its functions. However, there is nothing to prevent another module from reading or writing that address.

DLLs provide a way to modularize applications so that their functionality can be updated and reused more easily. DLLs also help reduce memory overhead when several applications use the same functionality at the same time, because although each application receives its own copy of the DLL data, the applications share the DLL code.

The Windows application programming interface (API) is implemented as a set of DLLs, so any process that uses the Windows API uses dynamic linking [101]. The Zlob Trojan poses a serious threat because it has the ability to download various malware DLL files onto the computer system.

Process - An application consists of one or more processes. A process, in the simplest terms, is an executing program. One or more threads run in the context of the process. The process is a program being executed by Windows. It consists of at least one or multiple threads that are being executed by the CPU. Malicious software can create a new or modify an existing process to execute the program outside of the current one. By tradition, the malware contains its own independent process, however, the newer malware commonly executes its own code as part of another process [77].

The feature extraction algorithm has been described in Algorithm 1. For feature extraction, R defines as a set of JSON report files and F contains a set extracted API features files for each malware sample.

Algorithm 1: Malicious Feature Extraction Algorithm (MFEA)

Input: R contains a collection of JSON report files R_i

Output: F contains API features file F_i, APIFDB

API = {file, registry, process, system, network, etc.}

```
while R do
  for each API in R do
      if API is existing in R then
          Extract this API from R;
          F += API;
          if this API is not existing in APIFDB or APIFDB is empty
              APIFDB += API
          end if
      end if
  end for
end while
```

The APIFDB defines as a total global **API F**eatures **D**ata**b**ase used by benign and malware samples. The process of extracting the API features will perform as long as there is an API feature in report file R. Then, the extracted features are stored in file for each sample. If the extracted feature does not exist in APIFDB or APIFDB is empty, this feature will add into APIFDB to create the dataset. The MFEA can provide to extract the DLL and Process features by using the keyword dll and process_name instead of api keyword.

The following Figure 4.4 provides the process flow of extracting the malicious and benign features from the JSON report files.

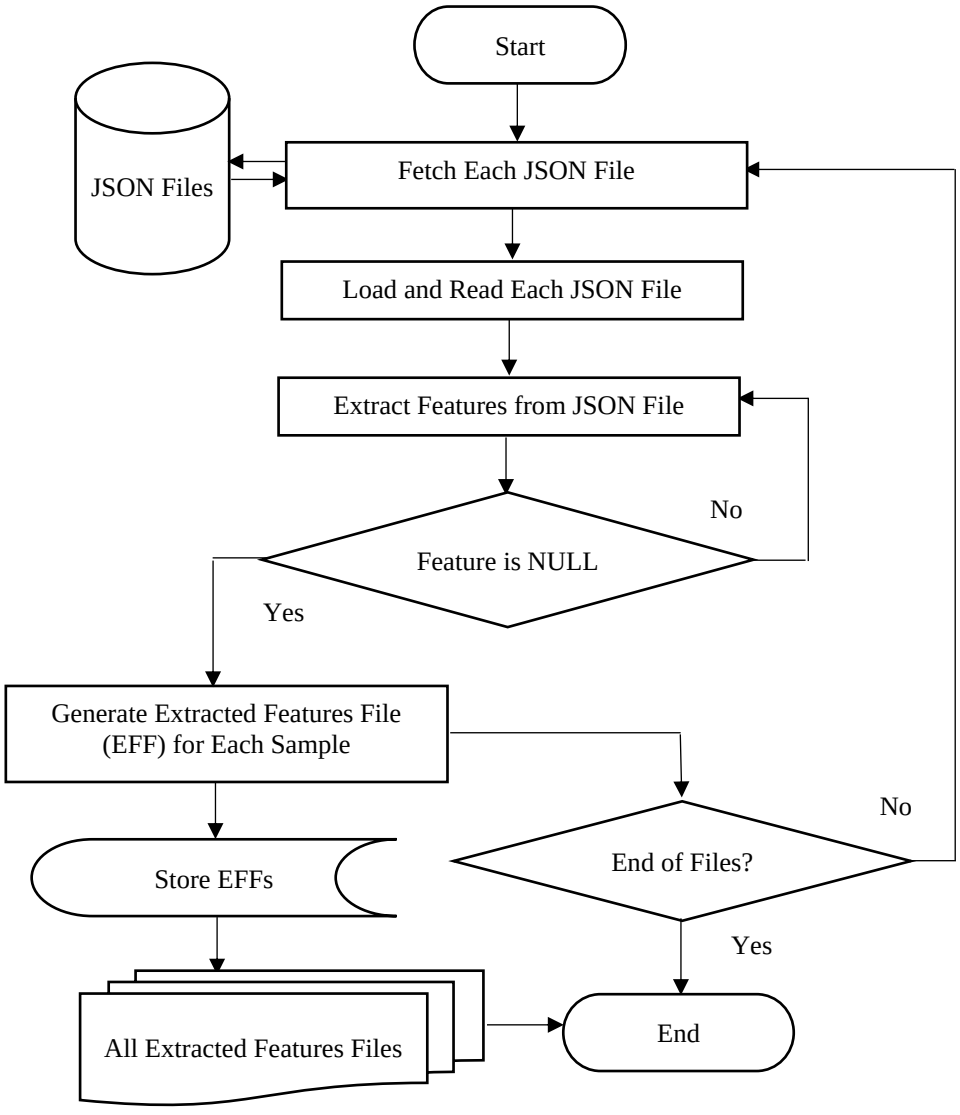

Figure 4.4 Benign and Malicious Feature Extraction Procedure

The feature extraction process is performed as long as the feature is existed in the report file. Moreover, the features extraction process is also performed

until the database of JSON report files is emptied. Then, these extracted features files (EFF) are kept for preparing the benign-malware dataset.

4.5 Postprocessing

In this section, the postprocessing means the steps of the process after performing the proposed feature extraction process. Then it contains three main parts such as data cleansing and discarding for extracted raw data, applying n-gram techniques, and representing and constructing the features dataset.

4.5.1 Cleansing Noise Data for Extracted Features Files

After collecting the extracted raw data, the next stage is the process of cleansing. The extracted raw data are naturally noisy and infected with some common issues, likewise missing values, incorrect or bad delimiters, inconsistent records, or inadequate parameters. And also, for some cases, the extracted raw data cannot be mended and so it must be removed but for other cases, it can be automatically or manually corrected. The process of removing noise data is important step for classification and detection systems in the machine learning field. The extracted raw features are extremely large and it is not suitable to handle the classification system. The step by step raw data cleansing processes are described below in detail and also illustrated in Figure 4.5.

1. Remove empty line if the extracted feature files have an empty line.
2. Remove noise data such as comma, colon, single code, double code, curly braces, and so on.
3. Remove duplicate feature by keeping the order of feature called by malware.
4. Discard the extracted feature files if the number of features does not have at least 10 features. If not, keep them in a database.

   The database contains all of the extracted feature files for all samples per family. The Extracted Features Files that keep in the database are defined as EFF in this system.
5. To create the malware/benign feature database, all files from the database are needed to combine into a single file.
6. Perform step 3 again to ensure there is no redundant feature in the database.
7. The final **B**enign-**M**alicious **F**eatures **D**ata**B**ase (BMFDB) has been developed after performing the previous steps.

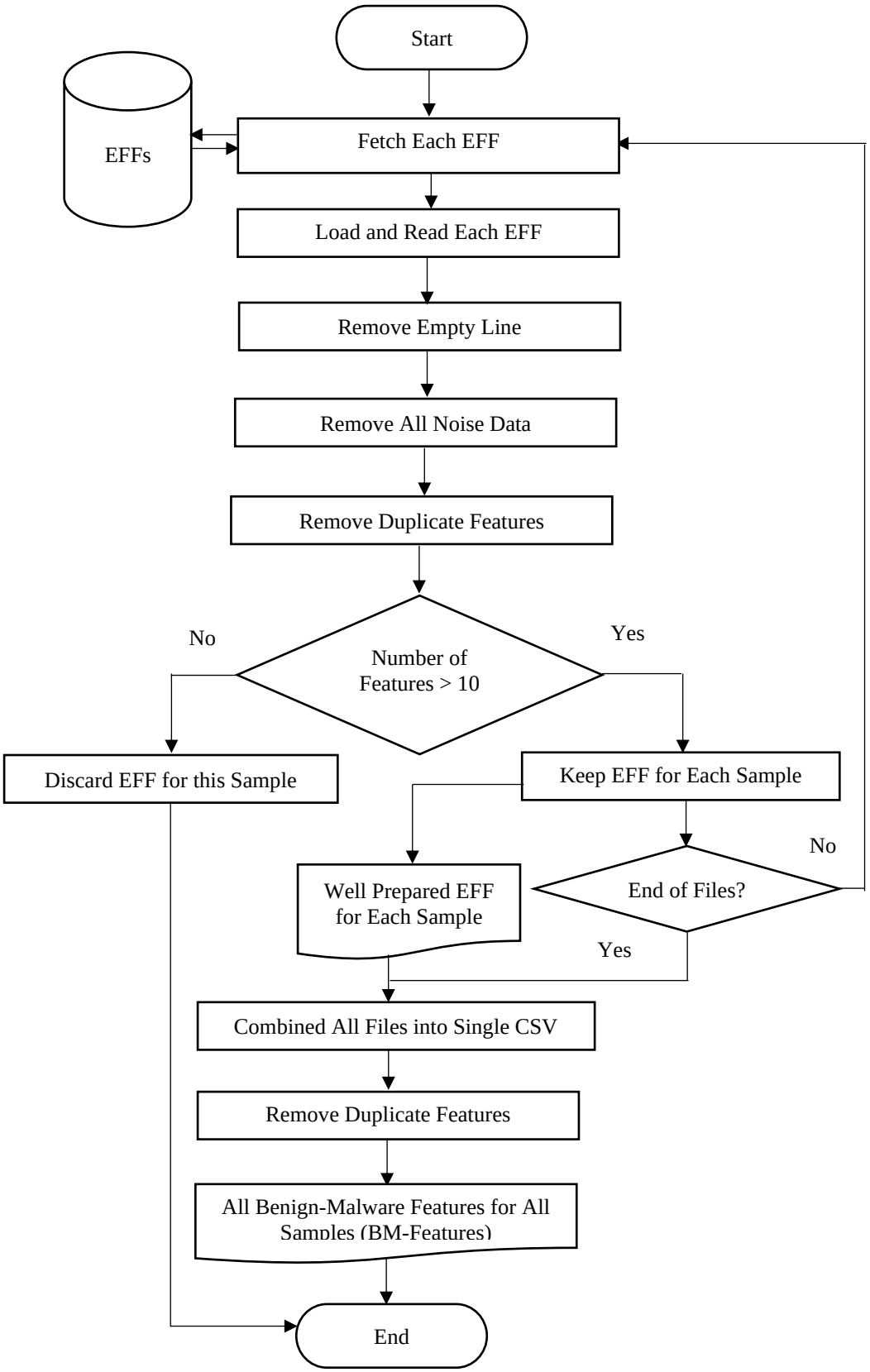

Figure 4.5 Extracted Raw Data Cleansing

The BMFDB is the database of all extracted features after the removing the noise data.

It has also been used as header for final malware classification dataset.

This section plays an important role in classification because the extracted features are quite large to handle the classification in terms of processing time and performance. Therefore, the system performs the data cleansing step for removing the noise and redundant features.

4.5.2 Applying N-gram

N-gram is an efficient method for text feature extraction, where n means the length of the feature. After processing the data cleansing steps described in the previous section, the N-gram method has been applied in this research. Because of the malicious features are correlated with each other sequentially according to their API categories e.g. NtClose (system), NtOpenKey (registry), NtQueryValueKey (registry), NtQueryAttributesFile (file). Therefore, the n-gram method is applied in this system to find the correlated API calls and to improve the classification performance. It is a continuous sequence of n^{th} items or grams from a given sequence. It is very useful for characterizing the sequences in Natural Language Processing, and DNA sequencing areas [2].

It has been adopted to extract the sequence of features in malware classification for both analyses. But the static is the one mostly used the n-gram such as opcode n-gram, byte-code n-gram, and API-call n-grams. The length determines the performance of the algorithm. As we concern about the low computational complexity and 2-gram provides the best accuracy, we use n-gram with (n = 1,2). Here, the term unigram and bigram will be used to represent the n-gram (n = 1,2) respectively for malware classification.

4.5.3 Representing and Preparing the Malware and Benign Dataset (MBDS)

The attributes representation process has been conducted after applying n-grams on extracted features. The process of attributes representation has been performed based on the presence and absence of features in the global feature database. The proposed system used the binary feature vector representation and described as follows:

$$API_i = \begin{cases} 1, & if \ \ API \ or \ Feature \ is \ in \ BMFDB \ File \\ 0, & otherwise \end{cases} \qquad \textbf{Equation 4.1}$$

The total global database BMFDB (Benign Malware Features DataBase) contains all

API features of malware and benign. For example, the sample F1 is the single malware instance feature representation and the last item 1 is the class label for malware family.

$$F1 = \{1,1,1,0,1,0,1,0,0,1, \ldots ,1\} \quad \textbf{Equation 4.2}$$

The attributes representation and dataset preparation process flow have been described in Figure 4.6.

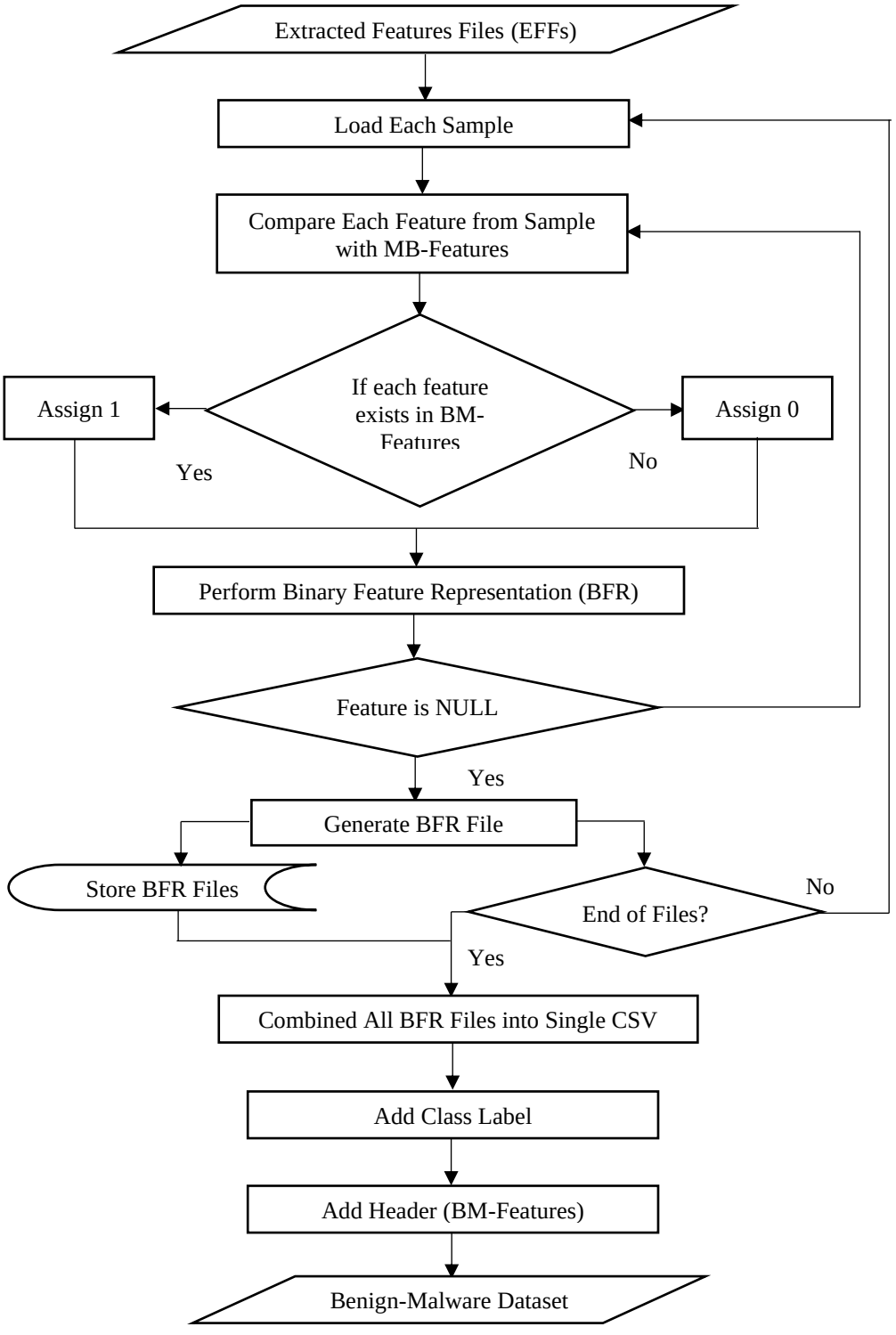

Figure 4.6 Feature Representation and Dataset Preparation

The database contains the Extracted Features Files (EFF) to perform the binary representation for the extracted features of executable files. If the extracted feature of a sample file exists in BMFDB, it will be denoted as 1. Otherwise, it will be denoted as 0. The Binary Feature Representation (BFR) of extracted feature file has been performed for each executable sample in this way. After processing this step, all BFR files are combined into a single CSV file for each family. Then, the Class label is added to every single instance per family, e.g. Adware as 1, Backdoor as 2, etc. This process flow has been performed repeatedly for other malware families as long as there is a family that exists. For example, there will be 5 MBDS if there are five families. Therefore, the MBDS combination has been performed to create the final dataset with multiple instances. The Benign Malware Family Classification (BMFC) Dataset and Benign- Malware Classification (BMC) Dataset have developed to perform the classification in this research.

4.6 Selecting Malicious and Benign Features

Selecting an important attribute is one of the essential techniques in data mining especially during the data processing. The purposes of applying the selection approaches are to select the appropriate features to the target class and to minimize the processing time. Feature or attribute selection methods are used for reducing the size of a feature dataset. The key role of the feature selection process is to improve the classification performance as well as improving the detection accuracy by choosing or transforming the feature set. Subsequently, the processing time for the classification process can speed up and improve the evaluation results since the feature number is reduced.

The main role of this phase is to improve the classification performance as well as improving the evaluation performance by selecting the effective and prominent attribute and removing the irrelevant from the dataset. The aim of using feature selection methods in this system is to discover the best features which can provide the improve classification rate of malicious and benign executables and to decrease the FP and FN rate. To remove the irrelevance features in classification, this research investigates two different feature selection methods.

Among the three FS methods such as filter, wrapper, and embedded methods, most researchers commonly used the filter-based approach in malicious classification

and detection research areas. The filtering approach does not depend on any particular algorithm. It is very fast and computationally less expensive than the other two methods. It is easy to scale to very high dimension datasets [21]. So, the system applies the χ^2 method from the filtering approach and Principal Component Analysis (PCA) approach for reducing the features.

The efficiency of the classification system can be improved by applying the attribute selection techniques such as Chi-Square (χ^2) and Principal Component Analysis (PCA). Chi-square (χ^2), and Principal Component Analysis (PCA) methods are applied to select the distinguish features.

4.7 Malware Classification using Machine Learning Classifiers

Machine learning can be divided into supervised and unsupervised learning. And most of the real-world ML applications such as object recognition and detection, Natural Language Processing, products or goods recommendations are based on multi-class classification algorithms.

The task of classification in the machine learning field is binary, multi-class, multi-labeled and hierarchical classification. The BMFC dataset has the multiclass classification problem which deals with high dimensional datasets because the malware dataset contains different malware types or families.

The overview of Benign Malware Classification (BMC) architecture has been described in Figure 4.7. This system provides the binary classification system for benign and malicious by extracting the prominent malicious features from dynamic analysis using three different ML classifiers such as SVC, k-Nearest Neighbor and Random Forest classifiers. These are implemented with the used of the scikit-learn Python ML library. The number of samples and family are described details in Chapter 6.

For evaluation metrics, five different options are considered to prove the effectiveness of extracted malicious/benign features. TPR, FPR, and accuracy can be generated from the confusion matrix of a classifier. The ROC curve has been plotted by using the TPR and FPR values.

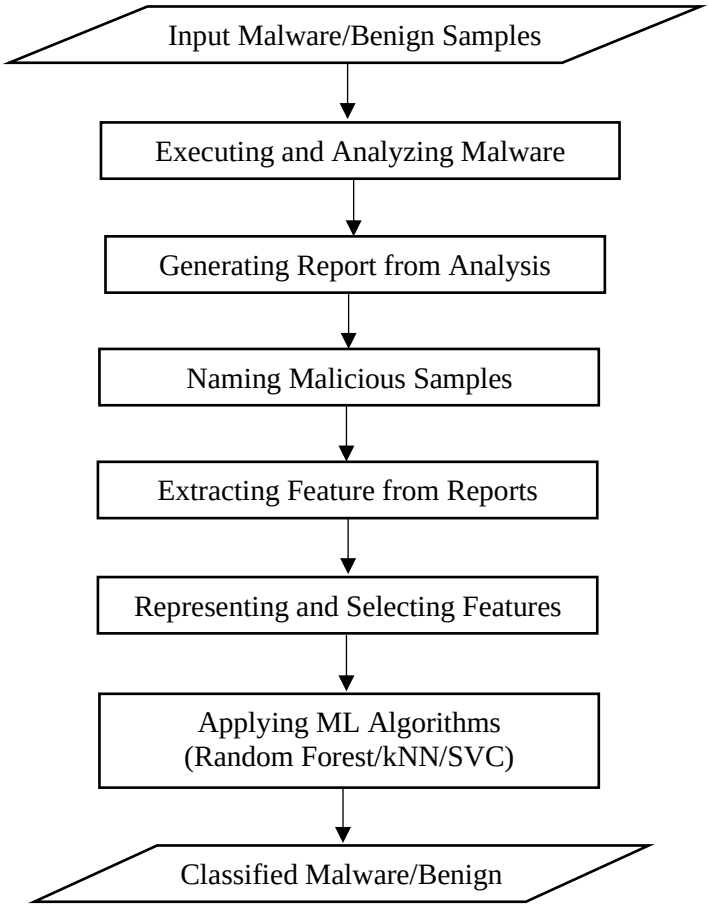

Figure 4.7 Overview Malicious-Benign Classification System

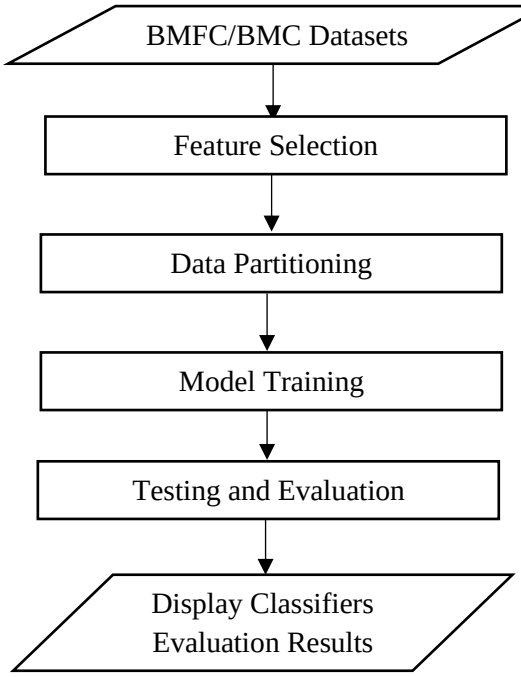

Figure 4.8 Performance Evaluation Metrics for BMFC and BMC

For classification purposes, three different classifiers are used in this proposed system such as SVC, k-Nearest Neighbor and Random Forest classifiers which can be used for multi-class classification in scikit-learn.

4.8 Summary

The proposed system performs dynamic malware and benign analysis using the sandbox called cuckoo. The sandbox implementation for analysis is carried out in the Ubuntu operating system. In this chapter, naming malicious samples and extracting malicious features from JSON reports are carried out by proposing the NMS_RE technique and MFEA approach. Therefore, this chapter describes the proposed effective malware and benign feature extraction technique and labeling or naming malicious samples for classification. Moreover, the data cleansing procedures have been conducted for the extracted feature and label files. Then the process of applying the N-gram technique to the extracted features is also performed in this proposed work due to the nature of malicious feature correlation. The binary feature or attribute representation procedure has been conducted to denote the presence or absence of extracted features by comparing the existence of MBFDB. Due to the problem of redundant and irrelevant features existing in high dimensional datasets, the performance of machine learning algorithms could be lower and unstable. Thus, this chapter also presents the feature selection methods that are applied in the malware classification system.

The implementation of the proposed malware classification system is described in Chapter 5, along with the step by step procedure and implementation. The performance evaluation metrics and discussion for the experiment are also discussed in Chapter 6.

CHAPTER 5

IMPLEMENTATION OF THE MALICIOUS SOTWARE ANALYSIS ARCHITECTURE

In the malicious programs and websites research fields, the researchers have proposed various techniques and methods to detect and identify anomalies and malicious behavior for various operating systems such as Windows, Unix, Linux, and Android, etc. A common massive behavioral detection for malware research is focused on the detection of an anomaly by the behavior learning process. This chapter delivers an implementation of the research experiment for malware classification. Before implementing the analysis environment, the analysts need to understand what is malicious software and what is malware analysis. Malicious software can be any software that can destruct or pose a risk to a computer system, information system, network system or users and they can exist various forms such as trojan, viruses, worms, adware, rootkits, backdoor, and spyware. The analysis of malicious software is the art of investigating the malware to know the following facts:

- how malware works or behaves,
- how to categorize, identify, and classify these malicious files, and
- how to detect or eliminate these suspicious files.

The different malicious files, systems, URLs analysis have been described in Chapter 3. The malicious features can be engineered based on different types of analysis. Section 5.1 describes why suspicious files analysis environment or analysis lab setup is needed in malware classification and detection fields. The next section 5.2 describes the step by step procedure to implement the analysis lab environment for suspicious software.

5.1 The Need for Malicious Software Analysis Lab

The malware analysis lab is an environment that setup securely for analyzing the suspicious files. This is generally a separated environment and it consists of various software and tools to analyze the malicious files for researchers and analysts. The analysis lab should be built that can resist the sudden attack of the modern new threats. The analysis lab can be regarded as a safeguard if the analysts or researchers run the

executable files accidentally or intentionally while performing the analysis. Different malware behaves differently based on the OS environment where they are being executed [53].

It is fairly simple to set up the analysis lab for analysts and researchers although it requires certain useful tools and hardware. As mentioned earlier, the isolated or separated safe environment is needed to create the perfect analysis lab. However, it is safe, but that is not enough by segregating the analysis lab from other computers within the network. Moreover, the authors from [53] suggested that the best way is isolating the analysis lab from the Internet. But, some malicious software, Trojan kinds of Remote Access Trojan and Botnets, need a connection to connect the Internet to accomplish their tasks fully. If the Internet connection is disabled, the malicious programs might raise connection error or not accomplish their tasks completely in this case.

The lab environment can generally be built into two ways such as physical and virtualization environments. It requires concern about the budget and time in building a physical lab environment and also space. On the other hand, these facts will be unconcerned by using the virtual lab environment. The analysts and researchers can restore the previous fresh virtual machine state like a snapshot by using the virtualization technique. The advantage of using the snapshot feature is no need to worry about malware infection. The analysts do not need to concern about infecting the Guest OS because it can easily restore the previous clean state.

The use of virtualization techniques in the automated analysis will save time and budget in analyzing malware samples. Virtualization plays a key role in automated malware analysis because of the cost effectiveness in resource utilization and consumption of CPU and hardware. Nevertheless, there are always pros and cons in both aspects. The malware writers can easily detect the use of virtualization technique and they can use the evasion methods for example packers, encryption, and code obfuscating techniques and so on. The sandboxing technique for automated analysis environment has been used in this research work. The implementation of the cuckoo sandbox is described in the next section.

5.2 The Implementation of Malware Analysis Lab Environment

The sandbox is used to run or execute the untrusted or untested code/programs from unreliable or unknown sources such as users, third-parties, websites, and suppliers. The cuckoo is an automated analysis tool that allows the analysts to perform suspicious files analysis. It is designed with multiple modules and it can be integrated by developing new module. It can analyze the DLL and executable files, various scripts such as PHP and VB, documents such as PDF, and Office, data compression archived files such as Zip and RAR, and HTML and URLs files.

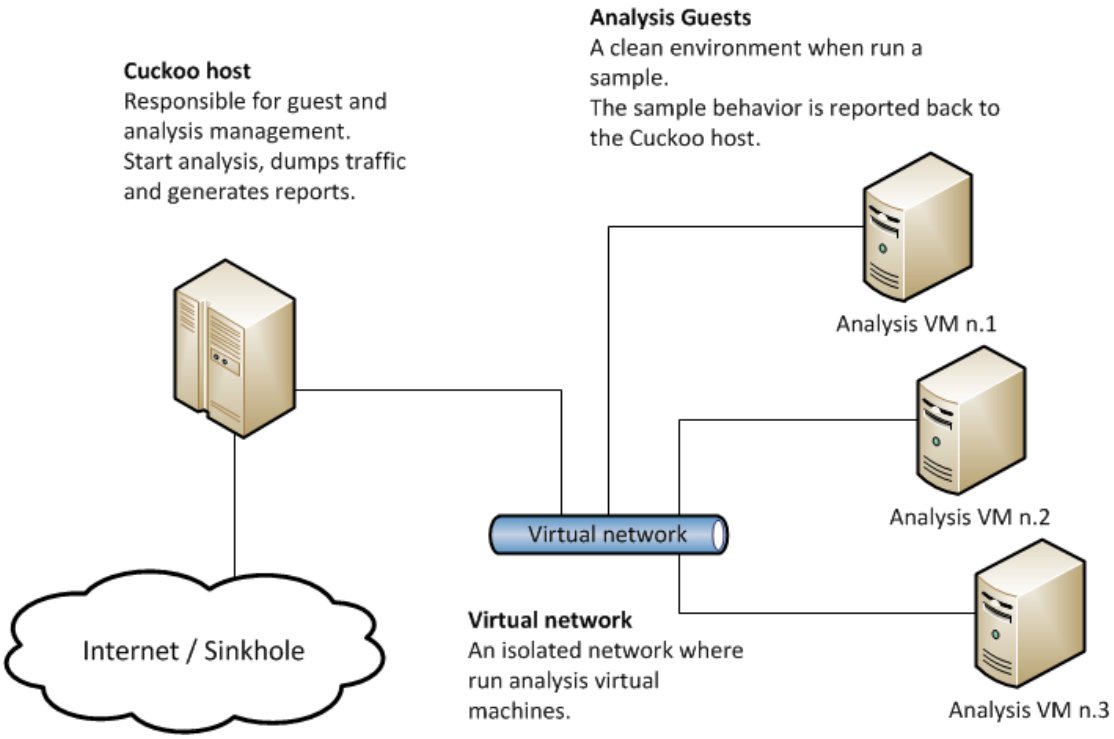

Figure 5.1 The Cuckoo Sandbox Architecture [19]

Figure 5.1 provides the main architecture of cuckoo sandbox architecture. It contains a central management software (CMS) that can handle the running and analysis of malicious programs. The key components of the cuckoo sandbox are CMS (Host machine) and one or more physical or virtual machines (Guest machines). The Host machine performs the whole process for analyzing and managing the executable files. The Guests/Virtual machines are the separated environments where the untrusted programs or software are securely executed and analyzed.

5.2.1 Setting Up the Analysis Lab Environment

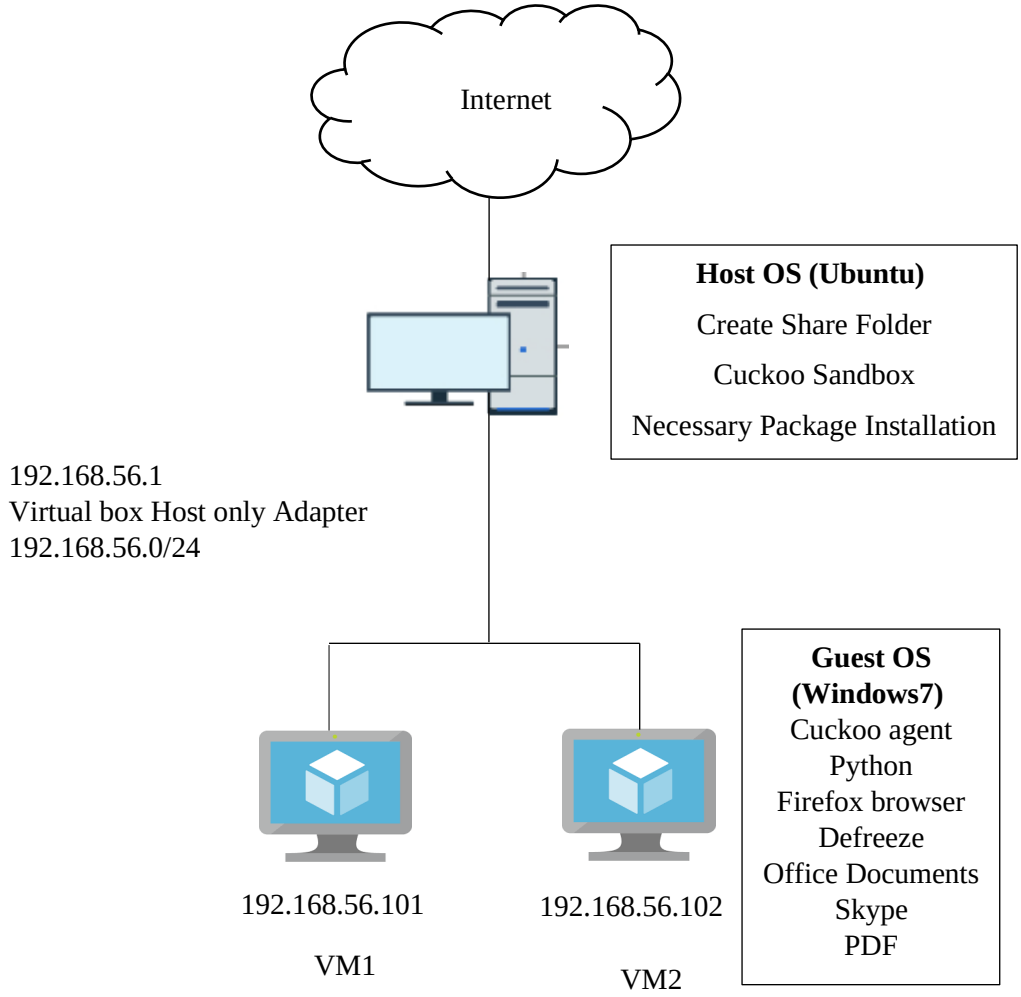

Figure 5.2 Analysis Lab Environmental Setup

Figure 5.2 illustrates the malicious lab environment set up for automated analysis. Windows-7 64 bit and Ubuntu 18.4 LTS-64-bit version have been used for analyzing the malicious samples as guest OS and host OS respectively. In Ubuntu 18.4, the sandbox namely cuckoo has been used for automated malware analysis system in this research work. The JSON (Java Script Object Notation) has been used to produce the analysis results. JSON report file is used in this system to extract the suspicious executable's features. The report file contains the name/label of the sample from many antiviruses using VirusTotal. The normal applications such as Browsers, Office Documents, Adobe Reader, data compression software, skype, etc. have been installed on virtual guest OS. Table 5.1 describes the system specification for analyzing the

malicious-benign samples. One Host Ubuntu OS contains two Guest Windows 7 OS and analyzes the samples simultaneously on two virtual machines using the headless function.

Table 5.1 System Specification for Analyzing Malicious-Benign Samples

Name	Specification
Host OS (2 machines)	Ubuntu 18. 4 LTS (64 bits)
Host Specification	Intel ® Core i5-3470 CPU @ 3.20GHz, 1TB Hard Disk, 4GB Memory
VM OS (4 machines)	Windows 7 64 bits
VMs Specification	1GB RAM, 100 GB Hard Disk

The installation steps for preparing the Host and Guest machines has been provided in the official website. After completing the malware-cleanware analysis lab environment setup, the analyzing process has been conducted for both suspicious and normal executable files.

5.2.2 Analyzing the Executable Samples

The step by step procedures of analyzing the executables and DLLs are described in Figure 5.3. After implementing the host and the guest machines completely, the snapshot of the guest OS's clean state needs to capture and restore before analyzing malicious samples.

Executing and analyzing the executables and DLLs files using a virtual machine (VirtualBox) have accomplished by performing the following steps:

1. Begin with a clean VirtualBox snapshot.
2. Start the Rooter of the sandbox.
3. Submit the samples that are being analyzed.
4. Run the cuckoo on Host machine.
5. Analyze the samples on the Guest machines.
6. Generate the analysis result on the CWD of the cuckoo.
7. Analyze the samples again until empty the submitted files by going to step 5.

The analysis process has been performed by conducting these steps described above. However, the interface of virtual machines needs to start up manually when it raises “unable to bind result server” error. It can be solved by starting and stopping the VM manually.

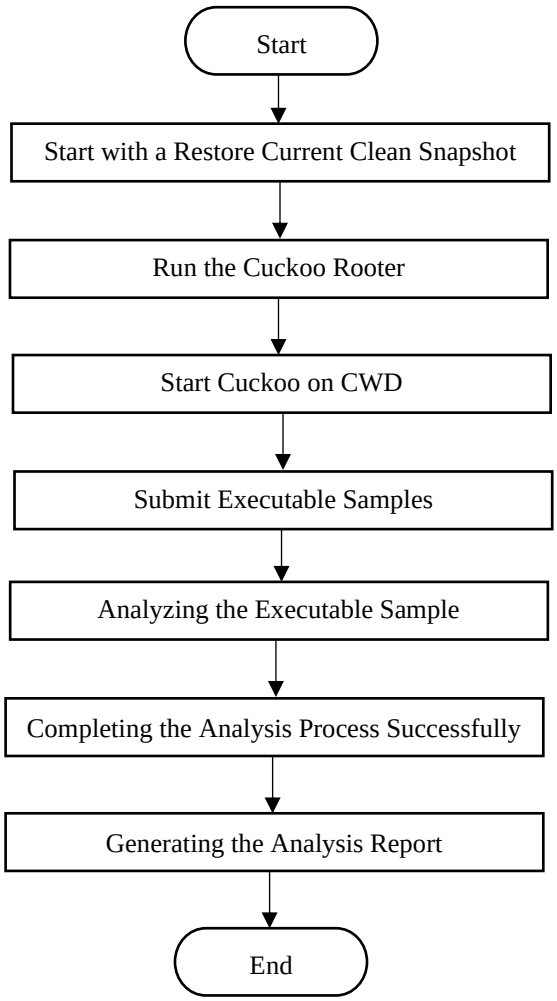

Figure 5.3 Process Flow of Analyzing the Benign and Malicious Executables

5.3 Experiment Environment of MBFE and BMFC

In this experiment, the family classification is developed with sklearn machine learning python libraries on Windows 10 OS with i7-7500U CPU @ 2.70GHz 2.90GHz, 12GB Memory, 1TB Hard Disk specifications. The specifications of the classification system and necessary software components of MBFE and BMFC are presented in Table 5.3. For classification purposes, the ML algorithms from sklearn or scikit-learn project python library have been used in this experiment. It is an open-source project and it contains up-to-date ML algorithms, and also details documentation

about the algorithm. Python has been used to implement the whole experiments in this research.

Table 5.2 System Specification for MBFE and BMFC

Name	Specification
OS	Windows 10 (64 bits)
OS Specification	Intel ® Core ™ i7-7500U CPU @ 2.70GHz 2.90GHz, 12GB Memory, 1TB Hard Disk
Software Components	- Sklearn - Scikitplot - Pandas - SciPy

5.4 Summary

This chapter provides the design and implementation of the proposed system. Section 5.1 explains the importance of the creation of the analysis lab environment and why do the analysts need it. Section 5.2 highlights the implementation of the analysis lab in detail with setting up the lab environment, analyzing the samples, and designing the Benign Malware Feature Extraction and Benign Malware Family Classification for experiments. Section 5.3 delivers the specification of the experiment environment for the BMFE and BMFC system.

CHAPTER 6

EXPERIMENTAL RESULTS AND DISCUSSIONS

In this chapter, the experimental results are discussed on performance evaluations of the proposed MFEA and NMS_RE method. All conducted experiments during this research will be discussed in this chapter. This chapter mainly focus on the evaluation of the robustness of proposed malicious feature extraction algorithm (MFEA) and naming malicious samples using Regular Expression (NMS_RE). For performance evaluation, this system tests on two different datasets with three different approaches along with unigram and bigram. Dataset-1 represents the classification between benign and four different malicious families as Benign-Malware Family Classification (BMFC) dataset. The dataset-2 represents the Benign-Malware Classification (BMC) dataset for malicious or not classification. Experimental results show the effectiveness of using the proposed MFEA and NMS_RE for both datasets. This chapter comprises four sections to support the executable sample description, features description, datasets description, evaluation results and discussions for experiments.

6.1 Executable Samples Description for Experiments

The executable samples have been collected from virusshare and tested over 40000 malwares and 8000 cleanware in the experiment. However, the total 21855 samples from different malware families including cleanware have been used to classify the malware samples. Due to the concern of imbalance class in multi-class classification problem, the samples are separated into two datasets. Therefore, the dataset-1 contains 18276 samples from 5 family including benign for malicious family classification with equal class label but not precisely.

Table 6.1 describes the different malware types and the total number of samples experimented in this research for malware family classification. However, these families are provided generally for each malware type but not specifically (e.g. sub-family of Trojan such as Trojan-Downloader, Trojan-Backdoor, etc. are not considered in this case).

Table 6.1 Executable Samples Description for BMFC Dataset

List	Family	# of samples for experiments
1	Clean	5642
2	Startpage	4014
3	Trojan	3200
4	Adware	2881
5	Zbot	2539
Total samples		**18276**

The dataset-2 contains 3579 malicious samples from 18 families and 3000 benign files for malicious or not classification. However, 3000 benign samples have been used to experiment the dataset-2 due to the concern of unbalance class. Table 6.2 describes the number of analyzed samples for the malware-benign classification. The malicious files are described in the following table together with family name and number of samples.

Table 6.2 Executable Samples Description for Benign-Malware Classification

List	Family	# of samples for experiments
1	Downloader	799
2	Backdoor	407
3	Virus	306
4	Worm	236
5	Domaiq	278
6	Installmonster	186
7	Qvod	50
8	Ransom	132
9	Loadmoney	100
10	Banker	116
11	Onlinegame	96
12	Bettersurf	89
13	Tool	87
14	Packer	234

15	Bitcoin	26
16	Swizzor	86
17	Hupigon	117
18	Unsafe	234
Total samples		**3579**

6.2 Features Description for Experiments

This research provides the malicious and benign features using three different feature types. Three different types of features which have been used to develop this research are API, API_DLL, API_DLL _PROCESS. The total number of features for each feature approach have been described in Table 6.3.

Table 6.3 Extracted Features Description for Experiments

List	Family	# of API	# of DLL	# of PROCESS
1	Clean	286	331	3066
2	Zbot	272	63	1433
3	Startpage	168	9	1097
4	Trojan	303	204	4360
5	Adware	276	37	1867
6	Downloader	243	32	496
7	Backdoor	260	49	409
8	Virus	259	72	313
9	Worm	244	36	286
10	Domaiq	161	4	239
11	Installmonster	176	16	212
12	Qvod	147	28	73
13	Ransom	218	22	99
14	Loadmoney	184	7	80
15	Banker	189	18	158
16	Onlinegame	114	13	56
17	Bettersurf	179	8	43
18	Tool	232	31	125

19	Packer	239	22	245
20	Bitcoin	159	17	45
21	Swizzor	81	10	77
22	Hupigon	200	20	115
23	Unsafe	250	49	185

6.3 Datasets Description for Experiments

In this research, two different datasets have been used to experiment the classification system. The approach 1 only considers the API features and approach2 considers the API and DLL features. The approach 3 reflects the API, DLL, and Process features in this research work. Table 6.4 shows the number of extracted features per family for each feature approach.

Table 6.4 Description of Dataset-1 for Experiments (unigram)

Class Label	Family	# of API	# of API_DLL	# of API_DLL_PROCESS
0	Clean	286	617	3683
1	Zbot	272	335	1768
2	Startpage	168	177	1274
3	Trojan	303	507	4867
4	Adware	276	313	2180

Table 6.5 supports the features description for dataset-1. This table also shows the number of features per family for each feature approach on bigram sequence.

Table 6.5 Description of Dataset-1 for Experiments (bigram)

Class Label	Family	# of API	# of API_DLL	# of API_DLL_PROCESS
0	Clean	13944	16098	23272
1	Zbot	4073	4463	7884
2	Startpage	712	727	3699
3	Trojan	14529	16232	27084
4	Adware	5938	6177	10766

Table 6.6 and Table 6.7 describe total number of features for malicious or not classification on unigram and bigram sequence. The dataset-2 contains the 3000 benign samples and 3579 malicious samples for malicious or not classification.

Table 6.6 Description of Dataset-2 for Experiments (unigram)

Class Label	Family	# of API	# of API_DLL	# of API_DLL_PROCESS
0	Clean	286	617	3683
1	Malicious	312	428	3039

Table 6.7 Description of Dataset-2 for Experiments (bigram)

Class Label	Family	# of API	# of API_DLL	# of API_DLL_PROCESS
0	Clean	11134	13015	16492
1	Malicious	11103	11943	17947

6.4 Evaluation Results and Discussions

To assess the performance of the model, the new test data will be used, which have labeled. An appropriate way is using the train-test-split to partition the dataset into a separate test and train data randomly. This is generally done by partitioning the labeled data set into train and test parts. The train data will be used to build the ML model and it is called the training data or training set. The rest of the data will be separated into validation set and testing set to check how well the model works in this research. This function extracts 80% of the data as the training set, 10% as validation set and the rest 10% as the test set. The table provides the number of instances for the experiments.

Table 6.8 Train-Validation-Test Shape for Experiments (Dataset-1)

Data Shape	# of instances
Train Shape (Train)	14803
Validation Shape (Val)	1645
Test Shape (Test)	1828

The number of features for each family are described in the previous section for three approaches and two datasets. Table 6.9 provides the total number of features for

all experiments.

Table 6.9 Total Number of Features for Experiments

Datasets	# of API (Approach1)	# of API_DLL (Approach2)	# of API_DLL_PROCESS (Approach3)
BMFC (Unigram)	324	746	12216
BMFC (Bigram)	21637	25039	53022
BMC (Unigram)	332	704	6284
BMC (Bigram)	16222	18568	27915

6.4.1 Evaluation on BMFC Dataset for Unigram

This section expresses the experimental results of dataset-1 on unigram features. The evaluation metrics are accuracy, confusion matrix (CM), TPR, FPR and ROC curve in this research. The subsections provide the accuracy scores, CM, and ROC curves for three approaches on unigram dataset.

6.4.1.1 Evaluation on Approach 1 (API)

Table 6.10 describes the accuracy of dataset-1 for unigram with approach 1 (API). The table showed that the selected features using Chi2 provide better accuracy than the PCA FS method. It also provides better accuracy than without using the feature selection method.

Table 6.10 Accuracy (%) on Approach 1 (API/unigram)

Classifiers	Chi2 FS (251 selected features)		PCA FS (251 selected features)		Without FS (324 features)	
	Val	**Test**	**Val**	**Test**	**Val**	**Test**
RF	97.44	**98.16**	97.3	97.89	97.39	97.87
kNN	96.57	97.35	96.42	97.17	96.64	97.11
SVC	96.96	97.39	96.74	97.35	97.08	97.28

RF classifier provides better accuracy than the other two classifiers with and without using feature selection methods. The SVC classifier provides the accuracy slightly better than kNN classifier. The last column shows the accuracy of the test set without performing the feature selection method. The accuracy from the experiment

shows that the effectiveness of extracted features using the proposed feature extraction method.

The following Figure 6.1 provides the CM of approach 1 for RF classifier. It describes that the Clean samples can correctly predict 570 samples and incorrectly predict 9 samples as Trojan and Adware. Thus, the True Positive (TP) number is 570 and the False Negative (FN) is 9. The False Positive (FP) number for Clean family is 25 since the Trojan and Adware incorrectly classify 22 and 3 samples as Clean family.

The True Negative (TN) is the combination of all numbers except the row and column of its own class. Thus, the TN number is 1224 for the Clean family.

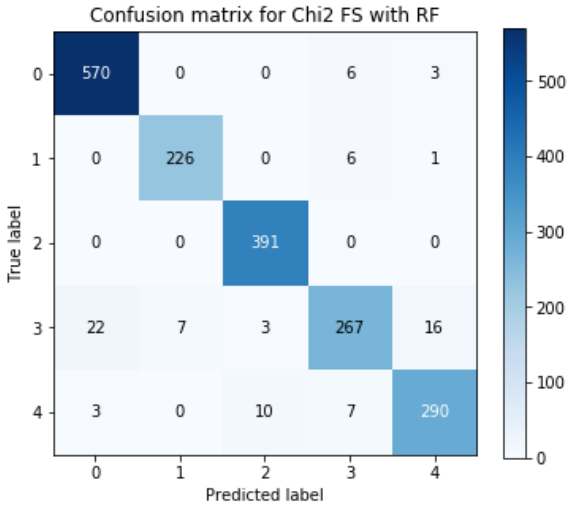

Figure 6.1 CM for Approach 1 using Chi2 FS with RF Classifier

The Zbot family provides 226 True Positive number and 7 False Negative number. Then the False Positive number is 7 since Trojan misclassifies 7 samples as Zbot. The Startpage family provides 391 TP number since there is not any incorrectly classified instances number. But the False Positive number is 13 since Trojan and Adware incorrectly predict 3 and 10 samples into Startpage.

However, the Trojan family incorrectly classifies 48 samples. Thus, the TP number is 267 and the FN number is 48 in the Trojan family. Then the FP number of Trojan family is 19 since the Clean, Zbot, and Adware families have the incorrect prediction. The Adware family provides 20 FP number and 20 FN number, respectively. The TP number for the Adware family is 290 samples. The combination of RF classifier with Chi2 FS method provides a better TP number than RF classifier with the PCA FS method. Also, RF classifier with full feature set (all features) provides a better TP number than using Chi2 and PCA feature selection methods in the Clean family.

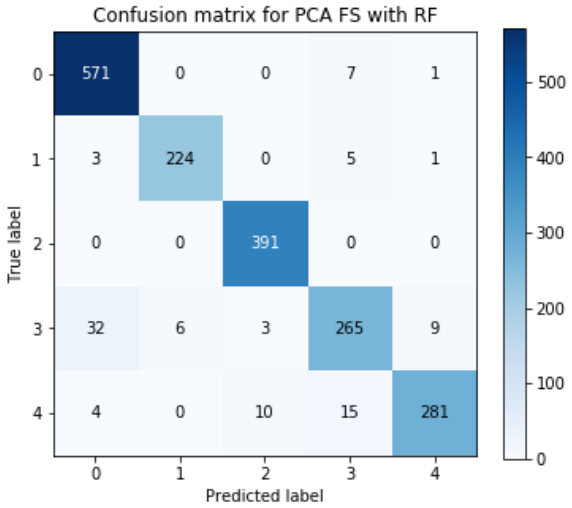

Figure 6.2 CM for Approach 1 using PCA FS with RF Classifier

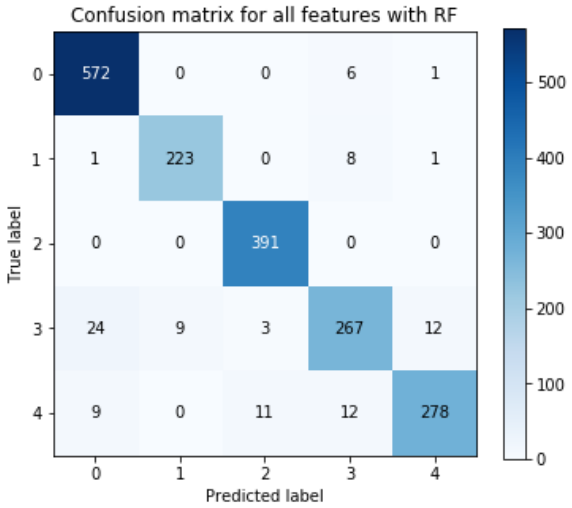

Figure 6.3 CM for Approach 1 using all features with RF Classifier

Table 6.11 provides the TPR and FPR for all classifiers for with and without performing the feature selection method. The Chi2 FS with RF classifier provides lower FPR than the PCA FS method. Moreover, the SVC classifier also provides lower FPR than the kNN classifier. The SVC classifier also supports higher TPR than the kNN classifier.

Table 6.11 TPR/FPR for Approach 1 on Test Set

Classifiers	**Chi2 FS TPR/FPR**	**PCA FS TPR/FPR**	**Without FS TPR/FPR**
RF	**0.94/ 0.011**	0.93/ 0.013	0.93/ 0.013
kNN	0.92/ 0.017	0.92/ 0.018	0.91/ 0.018
SVC	0.92/ 0.016	0.93/ 0.013	0.92/ 0.017

The following Figure 6.4 to 6.6 provide the ROC curves of RF classifier for with and without performing FS method.

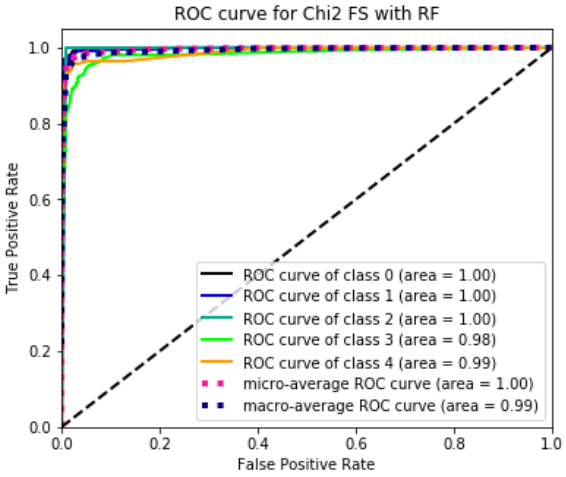

Figure 6.4 ROC curve for Approach 1 using Chi2 FS with RF Classifier

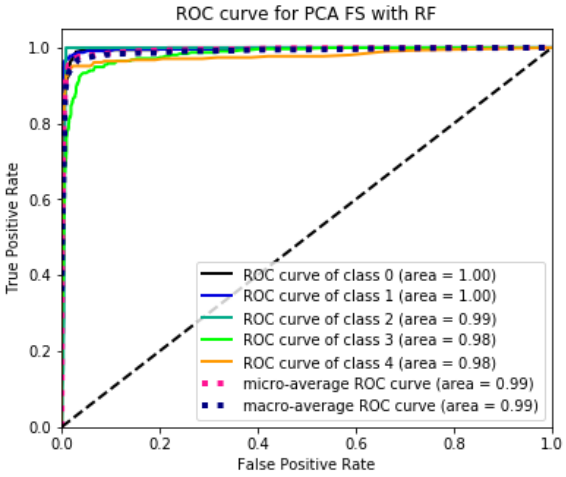

Figure 6.5 ROC curve for Approach 1 using PCA FS with RF Classifier

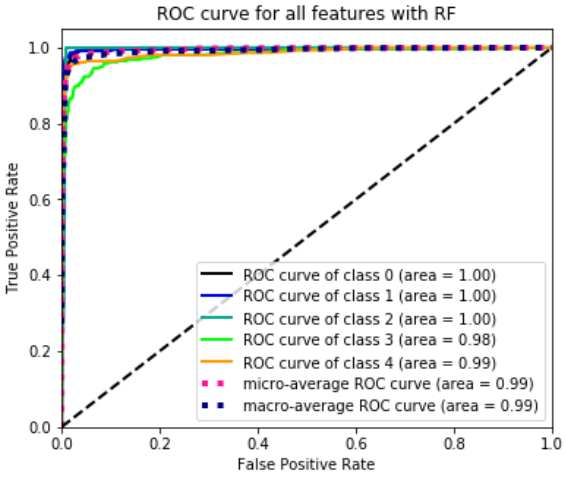

Figure 6.6 ROC curve for Approach 1 using all features with RF Classifier

Alternatively, ROC curve delivers an improved picture on how well the classifier behaves in general. The TPR and FPR are used to plot the ROC curve and the ROC area "1" mean it has low FPR and high TPR of a classifier. The ROC area of RF classifier is slightly better than others. The RF classifier with Chi2 FS provides lower FPR than the PCA FS. Besides, it provides better FPR than all features without using the features selection method. Thus, the micro-average ROC curve area differs while the ROC curve area remains the same per class.

According to the ROC curves of the RF classifier, the FPR of PCA FS with RF classifier is slightly higher than the Chi2 FS with RF classifier.

6.4.1.2 Evaluation on Approach 2 (API_DLL)

In this approach, the RF also outperforms than the other two classifiers before and after applying the FS methods. This approach provides more accuracy than approach 1. The use of the Chi2 feature selection method with the RF classifier provides the accuracy slightly better than without using the feature selection method. Moreover, it is better than the PCA feature selection method with the RF classifier. However, the kNN classifier provides classification accuracy equally for Chi2 FS and without using the FS method. The following Figure 6.7 to 6.9 provides the confusion matrix of all classifiers for API_DLL (approach2) on test data.

Table 6.12 Accuracy (%) on Approach 2 (API_DLL/unigram)

Classifiers	Chi2 FS (300 selected features)		PCA FS (300 selected features)		Without FS (746 features)	
	Val	Test	Val	Test	Val	Test
RF	97.64	**98.4**	97.34	98.07	97.54	98.07
kNN	96.88	97.41	96.71	97.11	96.57	97.41
SVC	97.1	97.59	97.22	97.54	97.27	97.52

The number of TP in all families of RF classifier provides higher than the other classifiers except Startpage family. The Startpage family provides same TP number in all classifiers.

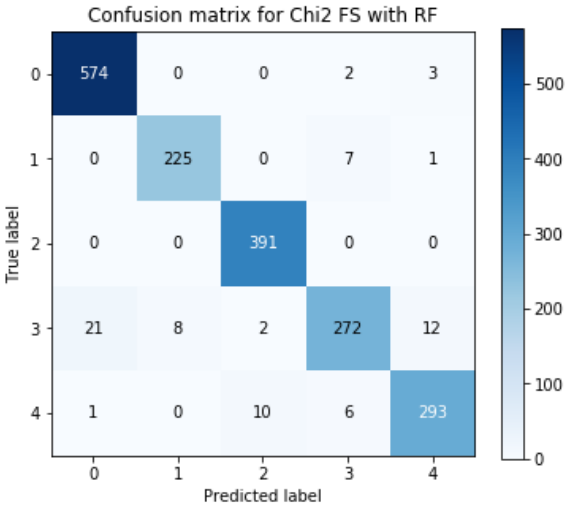

Figure 6.7 CM for Approach 2 using Chi2 FS with RF Classifier

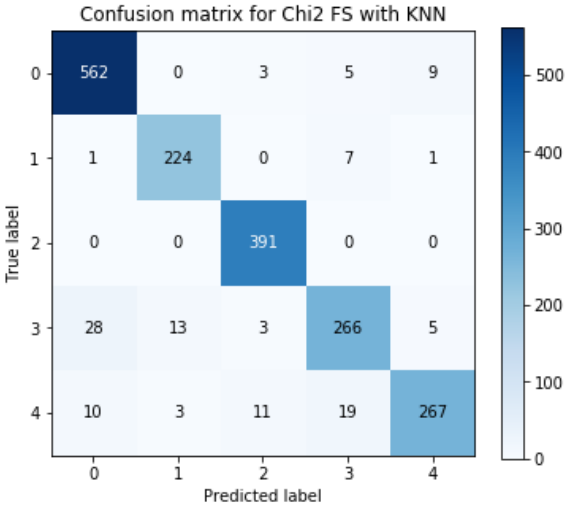

Figure 6.8 CM for Approach 2 using Chi2 FS with kNN Classifier

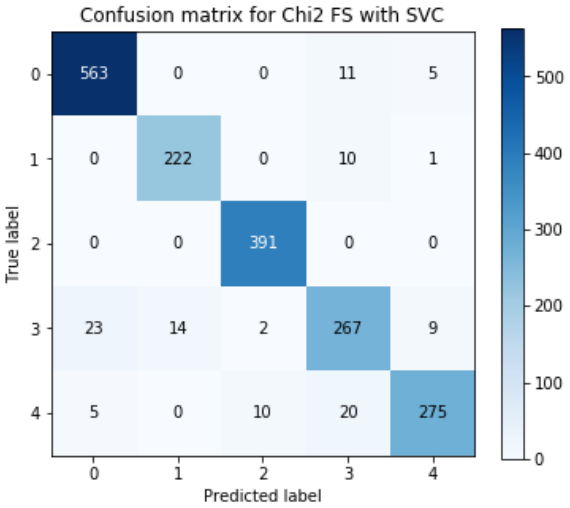

Figure 6.9 CM for Approach 2 using Chi2 FS with SVC Classifier

The results of kNN and SVC classifiers with Chi2 feature selection are slightly better than the PCA feature selection method. And it provides the classification result better than without using the feature selection method on the test set.

Table 6.13 describes the true positive rate and false positive rate for the API_DLL approach using three different classifiers. The Chi2 FS with RF classifier provides lower FPR than the PCA FS method.

The SVC classifier provides lower FPR than the kNN classifier. The SVC classifier also supports higher TPR than the kNN classifier before and after applying the feature selection method.

Table 6.13 TPR/FPR for Approach 2 on Test Set

Classifiers	Chi2 FS TPR/FPR	PCA FS TPR/FPR	Without FS TPR/FPR
RF	**0.95/0.01**	0.94/0.012	0.94/0.012
kNN	0.92/0.016	0.91/0.018	0.92/0.016
SVC	0.93/0.015	0.93/0.015	0.93/0.015

The following Figure 6.10 to 6.12 provides the ROC curves of Chi2 feature selection method with three classifiers.

The FPR in RF classifier is also lower than the other classifiers, and the SVC's FPR is lower than the kNN classifier.

However, the evaluation results of full feature sets with RF classifier provides a better true positive rate and false positive rate than other classifiers.

Therefore, the micro-average and macro-average ROC curve area of the RF classifier are better than those of kNN and SVC classifiers.

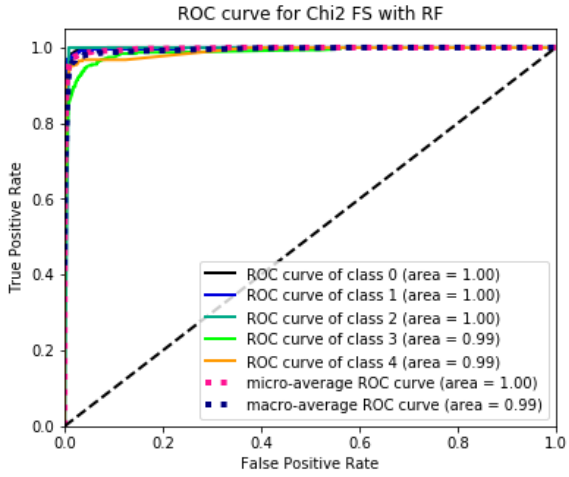

Figure 6.10 ROC curve for Approach 2 using Chi2 FS with RF Classifier

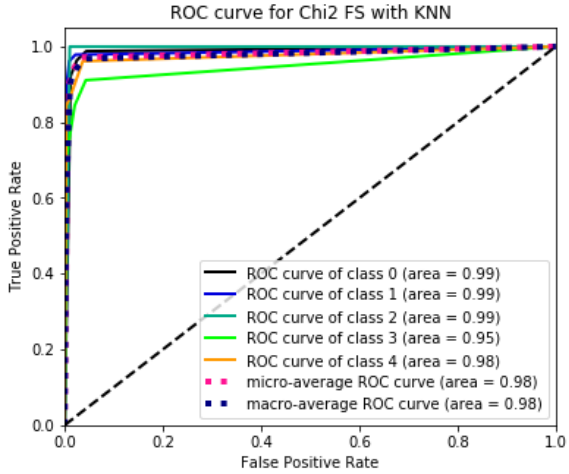

Figure 6.11 ROC curve for Approach 2 using Chi2 FS with kNN Classifier

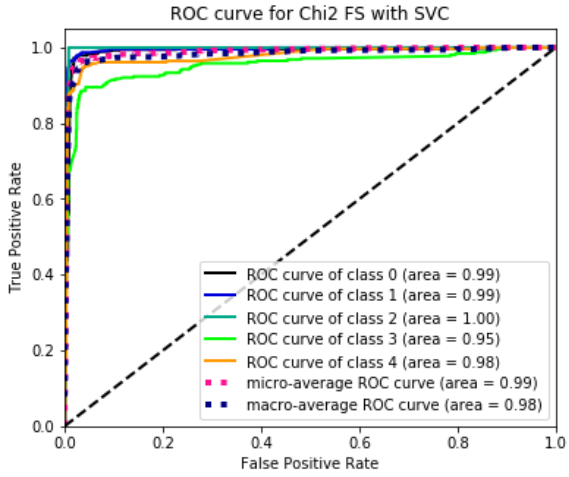

Figure 6.12 ROC curve for Approach 2 using Chi2 FS with SVC Classifier

6.4.1.3 Evaluation on Approach 3 (API_DLL_PROCESS)

This section shows the experiment of approach 3 for malware family classification by selecting the best 1000 features. Table 6.14 shows the comparison of accuracy on three different classifiers before and after applying the feature selection method.

Here, with the combination of Chi2 FS with all classifiers, the validation set and test set provide better accuracy than the PCA FS method. Besides it also provides better accuracy than without using the feature selection method.

Table 6.14 Accuracy (%) on Approach 3 (API_DLL_PROCESS/unigram)

Classifiers	Chi2 FS (1000 selected features)		PCA FS (1000 selected features)		Without FS (12216 features)	
	Val	Test	Val	Test	Val	Test
RF	97.56	**98.21**	97.51	98	97.51	98
kNN	96.93	97.35	96.79	97.33	96.79	97.46
SVC	97.15	97.39	97.1	97.52	97.64	98.11

Here, the validation set and test set of RF classifier with Chi2 FS provide better accuracy than other classifiers. The unigram (n-gram, where n =1) approach 3 (API_DLL_PROCESS features) provides high accuracy even without applying the feature selection method.

The last column, Without FS, is the experimental results of extracted features before applying the feature selection method. Hence, the experimental results show the effectiveness of extracted features using the proposed feature extraction method.

In this approach, the experimental results of the test set have been described by illustrating the confusion matrix on Chi2 feature selection with three different classifiers.

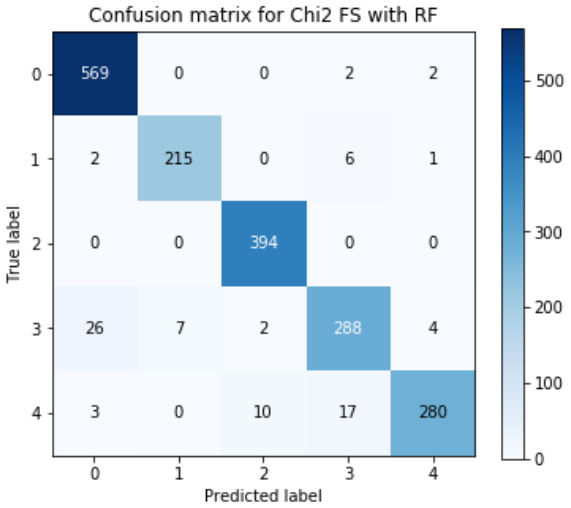

Figure 6.13 CM for Approach 3 using Chi2 FS with RF Classifier

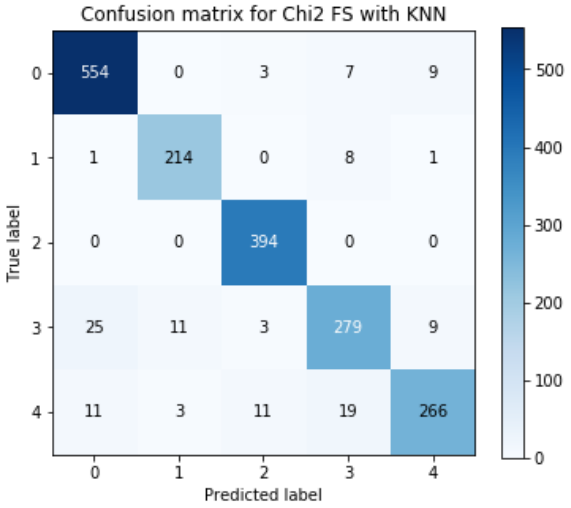

Figure 6.14 CM for Approach 3 using Chi2 FS with kNN Classifier

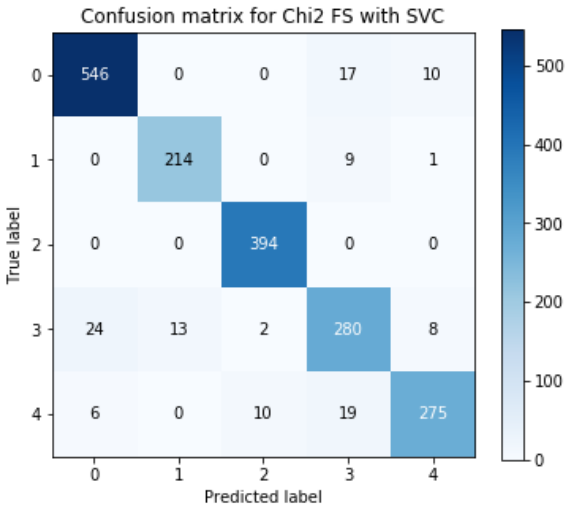

Figure 6. 15 CM for Approach 3 using Chi2 FS with SVC classifier

The TPR, FPR, and FNR can be calculated from the CM and these values are provided in the following Table 6.15. The RF classifier provides lower FPR on Chi2 FS than the PCA FS method.

Table 6.15 TPR/FPR for Approach 3 on Test Set

Classifiers	Chi2 FS TPR/FPR	PCA FS TPR/FPR	Without FS TPR/FPR
RF	**0.94/ 0.011**	0.94/0.0134	0.94/0.013
kNN	0.92/0.017	0.92/0.0172	0.93/0.016
SVC	0.93/0.016	0.93/0.0157	0.94/0.012

The SVC classifier also provides low FPR without applying the feature selection method. The SVC classifier also supports higher TPR than the kNN classifier. The following Figure 6.16 to 6.18 provide the ROC curves of Chi2 FS on three different classifiers.

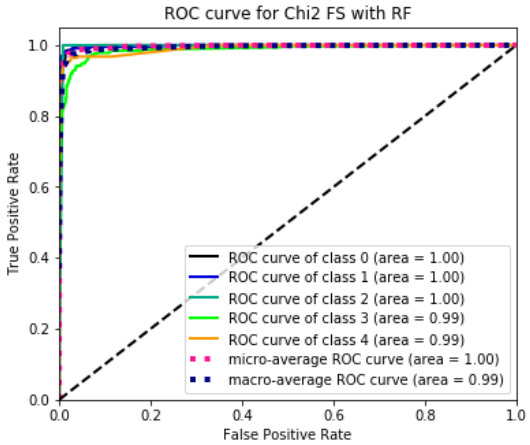

Figure 6.16 ROC curve for Approach 3 using Chi2 FS with RF Classifier

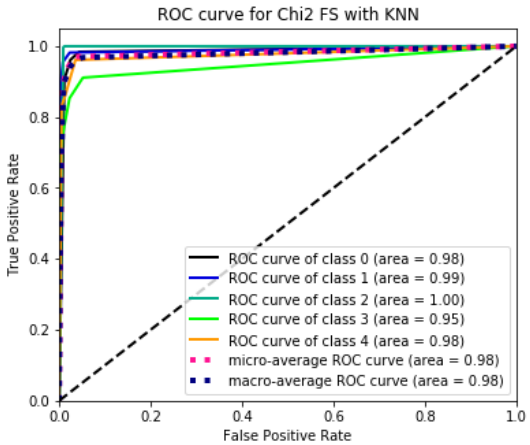

Figure 6.17 ROC curve for Approach 3 using Chi2 FS with kNN Classifier

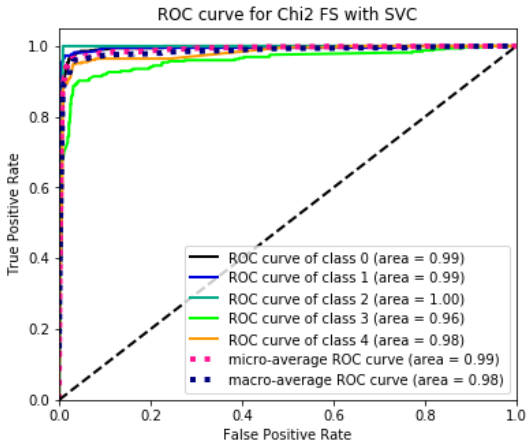

Figure 6.18 ROC curve for Approach 3 using Chi2 FS with SVC Classifier

According to the values of TPR/FPR in the table described above, the combination of RF classifier with the Chi2 feature selection method provides lower FPR and better ROC area scores than SVC and kNN classifiers. The SVC also provides lower FPR and better ROC area scores than the kNN classifier.

6.4.2 Evaluation on BMFC Dataset for Bigram

This section covers the experimental results of dataset-1 on bigram features. The evaluation metrics are accuracy, confusion matrix (CM), TPR, FPR and ROC curve in this experiment. The subsections present the accuracy, ROC curves, and CM for three approaches on the bigram dataset. In this case, approach 1 contains the 21637 features and 18276 instances. These instances are divided into three parts, train (14803), validation (1645), and test (1828) set. And then feature tuning is performed to get the best classification performance. The 1000 features provide the best accuracy for the BMFC dataset.

6.4.2.1 Evaluation on Approach 1 (API)

The following table describes the accuracy of dataset-1 for bigram with approach 1 (API). The table shows that the selected features using Chi2 provide better accuracy than the PCA feature selection method. However, the accuracy is not significantly increased whether or not applying the feature selection method. The combination of a 2-gram sequence in the API approach provides better accuracy than unigram (1-gram).

Table 6.16 Accuracy (%) on Approach 1 (API/ bigram)

Classifiers	Chi2 FS (1000 selected features)		PCA FS (1000 selected features)		Without FS (21637 features)	
	Val	Test	Val	Test	Val	Test
RF	97.83	**98.4**	97.47	98.29	97.49	98.11
kNN	96.96	97.26	96.83	97.3	96.54	96.76
SVC	97.25	97.57	97.27	98.11	97.03	97.5

This section provides the CM and ROC curve for the experiment results of RF with Chi2 (1000) selected features. The following figure illustrates the CM for the test set of Chi2 FS using the RF classifier.

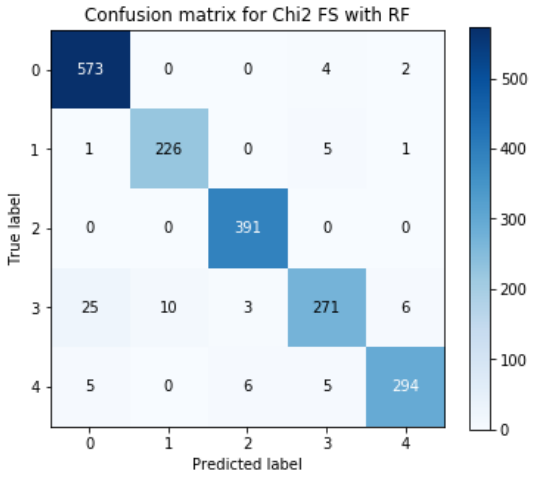

Figure 6.19 CM for Approach 1 using Chi2 FS with RF Classifier

Table 6.17 shows the TPR and FPR of approach 1 on the bigram BMFC dataset. The RF classifier provides better TPR and FPR than other classifiers.

Table 6.17 TPR/FPR for Approach 1 (bigram) on Test Set

Classifiers	Chi2 FS TPR/FPR	PCA FS TPR/FPR	Without FS TPR/FPR
RF	**0.95/0.0105**	0.949/0.0113	0.94/0.0125
kNN	0.92/0.0179	0.9261/0.017	0.91/ 0.0211
SVC	0.92/0.0157	0.946/0.0121	0.93/ 0.016

The ROC curve provides low FPR and high TPR for all classes. The ROC curve describes how much classifier is well-distinguished between classes. It typically uses

TPR on the Y axis, and FPR on the X axis. It means that the top left corner of the ROC curve is the ideal point - a FPR of zero, and a TPR of one.

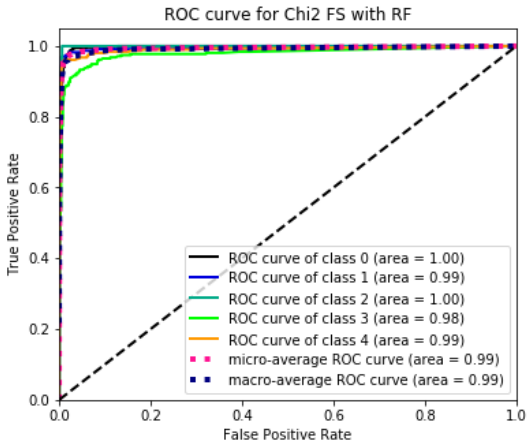

Figure 6.20 ROC curve for Approach 1 using Chi2 FS with RF Classifier

The ROC area score of Class 3, Trojan family, is less than the other classes due to the conflict between Clean (class 0), Zbot (class 1), and Adware (class 4). The RF classifier delivers better accuracy, TPR and FPR than those of kNN and SVC classifiers. And Chi2 FS method provides better experiment results than PCA FS method in all approaches.

6.4.2.2 Evaluation on Approach 2 (API_DLL)

This section describes about the experiments with approach 2 on bigram BMFC dataset. The following table shows the comparison of accuracy between two feature selection methods and also without using feature selection method on different classifiers. In approach 2 on bigram, the RF also outperforms than the other two classifiers before and after applying the FS methods. The accuracy of the SVC is slightly better than the kNN classifier on both Chi2 FS and PCA FS methods. The accuracy of SVC classifier is slightly better than the kNN on all features.

Table 6.18 Accuracy (%) on Approach 2 (API_DLL/ bigram)

Classifiers	Chi2 FS (1000 features)		PCA FS (1000 features)		Without FS (25039 features)	
	Val	Test	Val	Test	Val	Test
RF	97.86	**98.53**	97.51	98	97.51	98.27
kNN	97.15	97.46	97.2	97.52	96.74	97.24
SVC	97.11	97.63	97.44	98.14	97.05	97.43

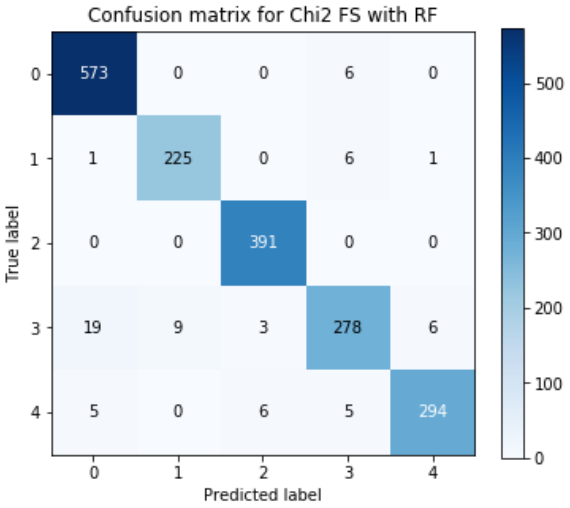

Figure 6.21 CM for Approach 2 using Chi2 FS with RF Classifier

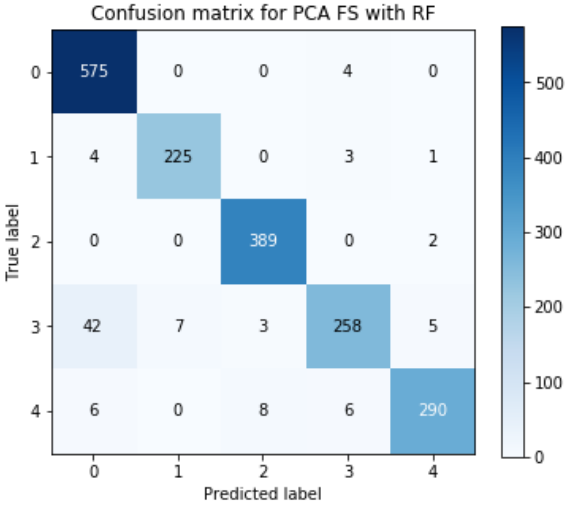

Figure 6.22 CM for Approach 2 using PCA FS with RF Classifier

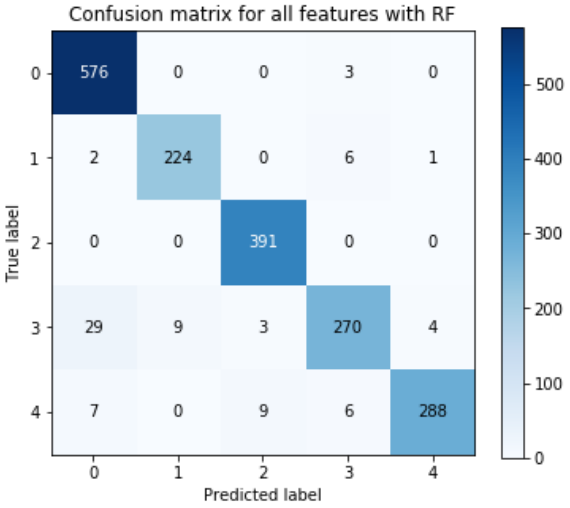

Figure 6.23 CM for Approach 2 using all features with RF Classifier

Figure 6.21 to 6.23 describe the CM of RF classifier for FS methods and without using

the FS method.

In the CM of Chi2 FS with RF classifier, the Zbot family (class 1) provides a lower FN number than the PCA FS method and without applying the FS method. In Chi2 FS, the Clean family (class 0) also has a lower FP than the PCA FS method and all features.

The Figure 6.22 provides the CM of PCA FS with RF classifier. The Clean family (class 0) provides a lower FN number than the Chi2 FS method. However, the Startpage family incorrectly predicts 2 instances as the Trojan family.

The below table describes the true positive rate and false positive rate for approach2 (bigram). The use of feature selection method with RF classifier provides lower FPR than the PCA feature selection method and without using the feature selection method. The SVC classifier provides lower FPR than the kNN classifier. The SVC classifier also supports higher TPR than the kNN classifier before and after applying the feature selection method.

Table 6.19 TPR/FPR for Approach 2 (bigram) on Test Set

Classifiers	Chi2 FS TPR/FPR	PCA FS TPR/FPR	Without FS TPR/FPR
RF	**0.95/0.009**	0.94/0.0132	0.94/0.011
kNN	0.92/0.0167	0.925/0.0174	0.92/0.018
SVC	0.93/0.0151	0.94/0.0124	0.93/0.0162

The RF classifier with Chi2 in approach 2 also provides better FPR and TPR than other classifiers. The RF classifier also produces low FPR in all features. In Chi2 FS, the SVC classifier offers FPR and TPR slightly better than the kNN classifier.

The combination of PCA FS with all classifiers, the SVC supports lower FPR than RF and kNN classifiers. However, the SVC and RF classifier have produced equal TPR in the PCA FS method. Without applying the FS method, the RF classifier supports better FPR and TPR than kNN and SVC classifiers. The TPR in RF classifier with Chi2 FS and without using the FS method provides slightly better than the PCA FS method.

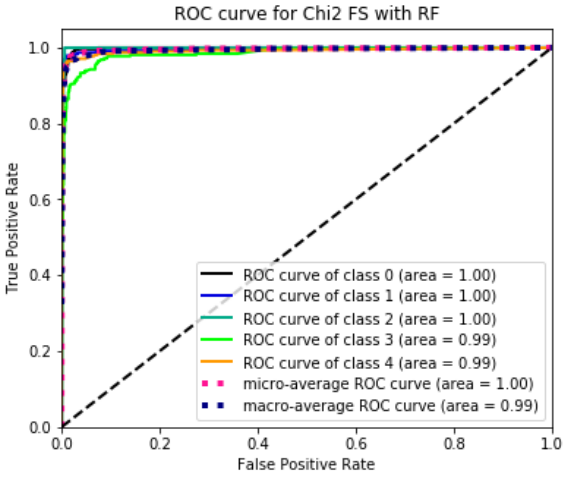

Figure 6.24 ROC curve for Approach 2 using Chi2 FS with RF Classifier

Thus, the ROC curve of RF classifier with Chi2 FS and all features provides better than the ROC curve of PCA FS with RF classifier.

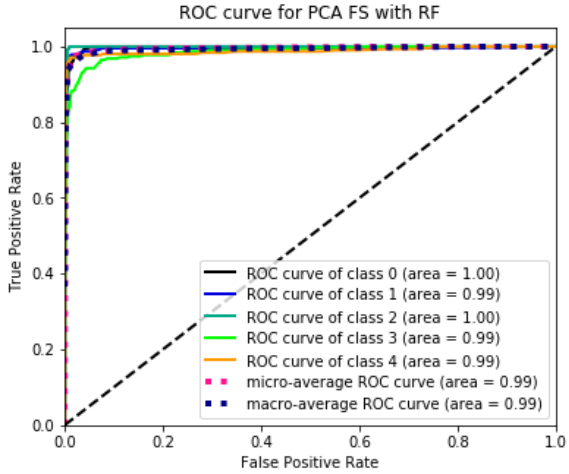

Figure 6.25 ROC curve for Approach 2 using PCA FS with RF Classifier

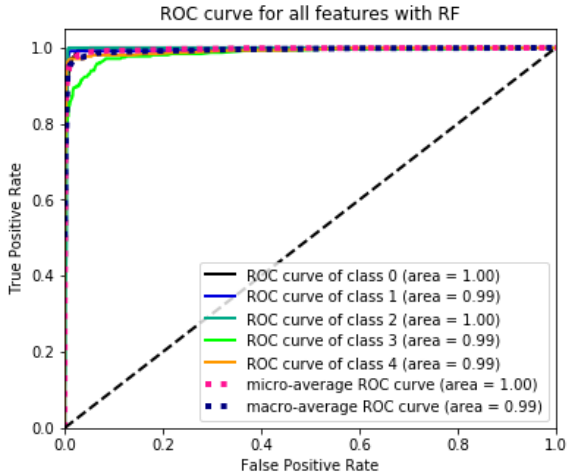

Figure 6.26 ROC curve for Approach 2 using all features with RF classifier

6.4.2.3 Evaluation on Approach 3 (API_DLL_PROCESS)

The following table shows the comparison of accuracy between validation and test set using different classifiers before and after applying the FS methods. In approach3 on bigram, the RF also outperforms than the other two classifiers before and after applying the FS methods.

Table 6.20 Accuracy (%) on Approach 3 (API_DLL_PROCESS/ bigram)

Classifiers	Chi2 FS (1000 selected features)		PCA FS (1000 selected features)		Without FS (53022 features)	
	Val	Test	Val	Test	Val	Test
RF	97.86	**98.49**	97.54	98.05	97.68	98.09
kNN	97.3	97.39	97.15	97.46	97	96.91
SVC	97.03	97.54	97.34	97.94	97.2	97.54

Moreover, approach 3 in bigram provides better accuracy than all approach on unigram and approach 1 on bigram. But, approach 2 on bigram outperforms than this approach on the BMFC dataset.

The bigram approach3 (API_DLL_PROCESS) provides high accuracy even without using the feature selection method. Therefore, the experimental results show the effectiveness of extracted features using the proposed feature extraction method.

Figure 6.27 to 6.29 provides the CM of three different classifier without applying the FS method on the test set. The Clean family (class 0) in the RF classifier provides a better TP number than other classifiers. The TP number and FN number of the Zbot family (class 1) in the RF classifier are equal to the SVC classifier.

Moreover, the RF classifier supports better TP number in Adware family (class 4) than kNN and SVC classifiers. However, the SVC classifier gives more TP numbers in Trojan family (class 3) than RF and kNN classifiers.

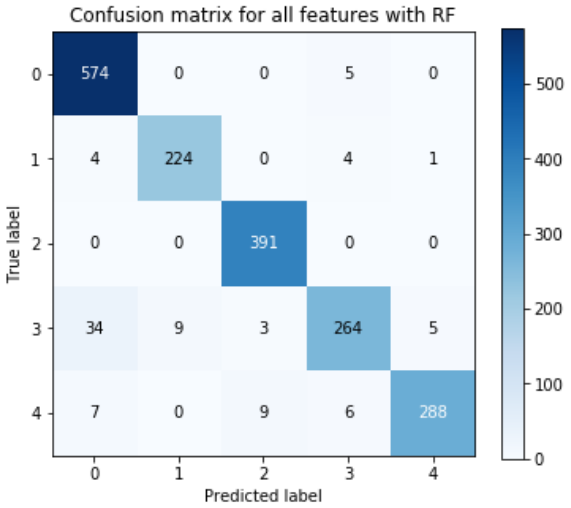

Figure 6.27 CM for Approach 3 using all features with RF Classifier

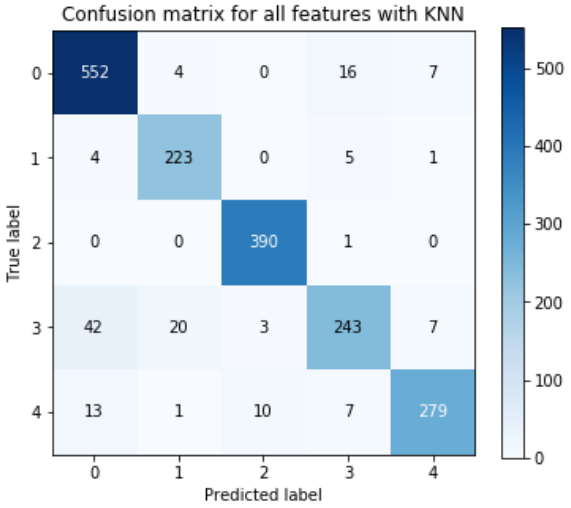

Figure 6.28 CM for Approach 3 using all features with kNN Classifier

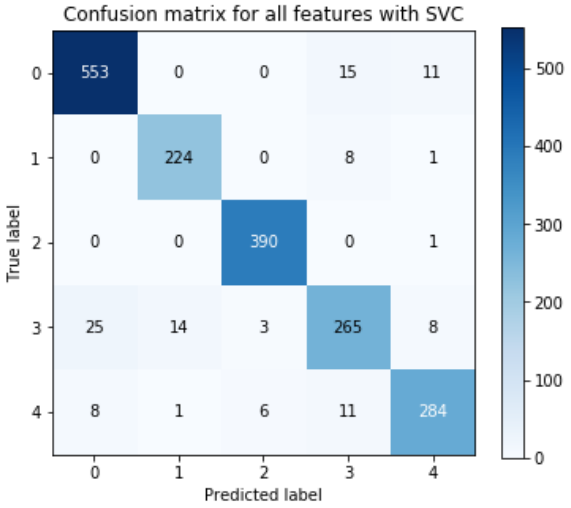

Figure 6.29 CM for Approach 3 using all features with SVC Classifier

Table 6.21 shows the TPR and FPR of approach 3 on the bigram BMFC dataset. The RF classifier provides better TPR and FPR than other classifiers. The SVC classifier provides higher TPR and lower FPR than the kNN classifier.

Table 6.21 TPR/FPR for Approach 3 (bigram) on Test Set

Classifiers	Chi2 FS TPR/FPR	PCA FS TPR/FPR	Without FS TPR/FPR
RF	**0.95/0.009**	0.94/0.0131	0.94/0.0127
kNN	0.92/0.017	0.92/0.0164	0.91/0.02
SVC	0.93/0.015	0.93/0.0131	0.93/0.0156

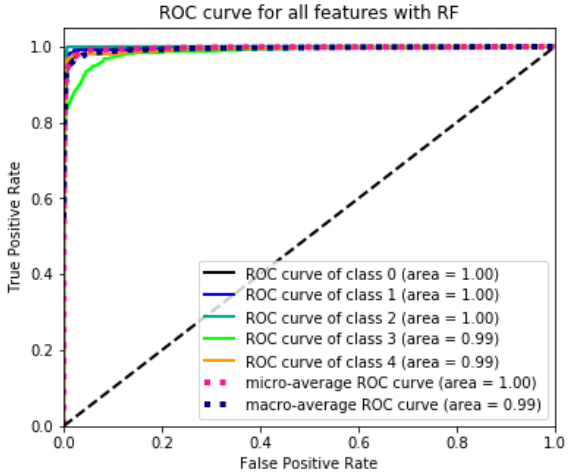

Figure 6.30 ROC curve for Approach 3 using all features with RF Classifier

The ROC curves have been described for three classifiers on test data without applying the FS method. The extracted features using the proposed feature extraction method provide low FPR on the test set of the BMFC dataset.

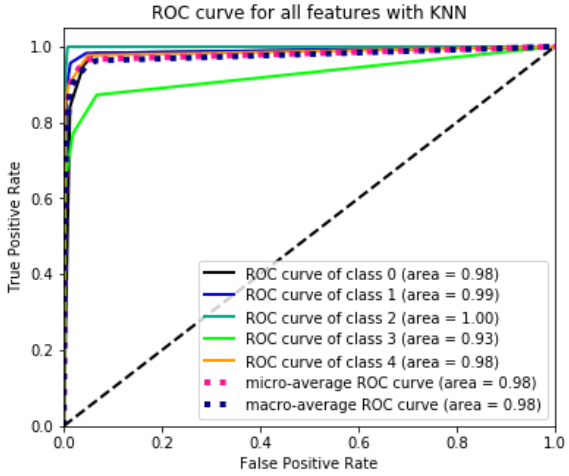

Figure 6.31 ROC curve for Approach 3 using all features with kNN Classifier

The SVC classifier delivers TPR and FPR better than the kNN classifier on all features without using the FS method. And the SVC also provides better accuracy than the kNN classifier.

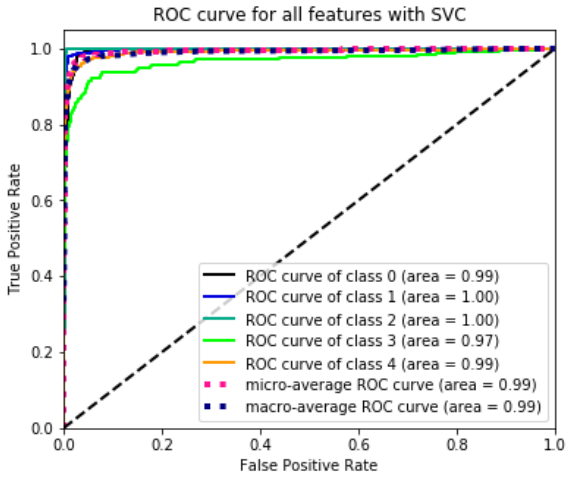

Figure 6.32 ROC curve for Approach 3 using all features with SVC Classifier

The ROC curves have been used to present the evaluation results as ROC Curves summarize the trade-off between the TPR and FPR for a predictive model using different probability thresholds. Therefore, the figures mentioned above describe the ROC curves of classifiers for selected Chi2 features on the test set.

6.4.3 Comparison of Evaluation Time on BMFC Dataset

The size of the bigram on approach 2 dataset is nearly 2Gigabytes in this experiment. The dataset loading time takes nearly an hour. The processing time of validation, and test set for all features with RF classifier is pleased to perform the classification. However, the processing time is quite long in all features for the kNN and SVC especially on unigram approach 3 and all approaches on bigram.

The following figures described the processing time for all features and selected features using three different classifiers. The combination of RF and Chi2 FS provides the best accuracy results and also for the processing time. It takes less time than PCA FS.

The combination of Chi2 and RF supports not only the best accuracy but also for the processing time. The below Figure 6.33 shows the comparison of evaluation time for approach 3(unigram). Here, it takes over one hour if feature selection is not applied.

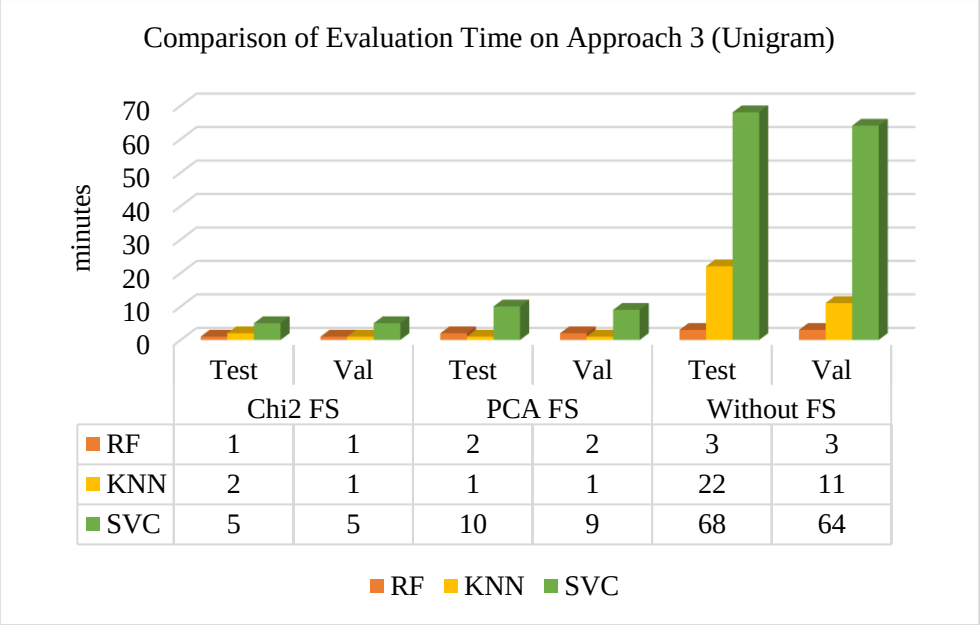

Figure 6.33 The Comparison of Evaluation Time for Approach 3 (unigram)

The below Figure 6.34 describes the evaluation time of approach 1 on bigram. The processing time of SVC classifier is more than RF and kNN classifiers.

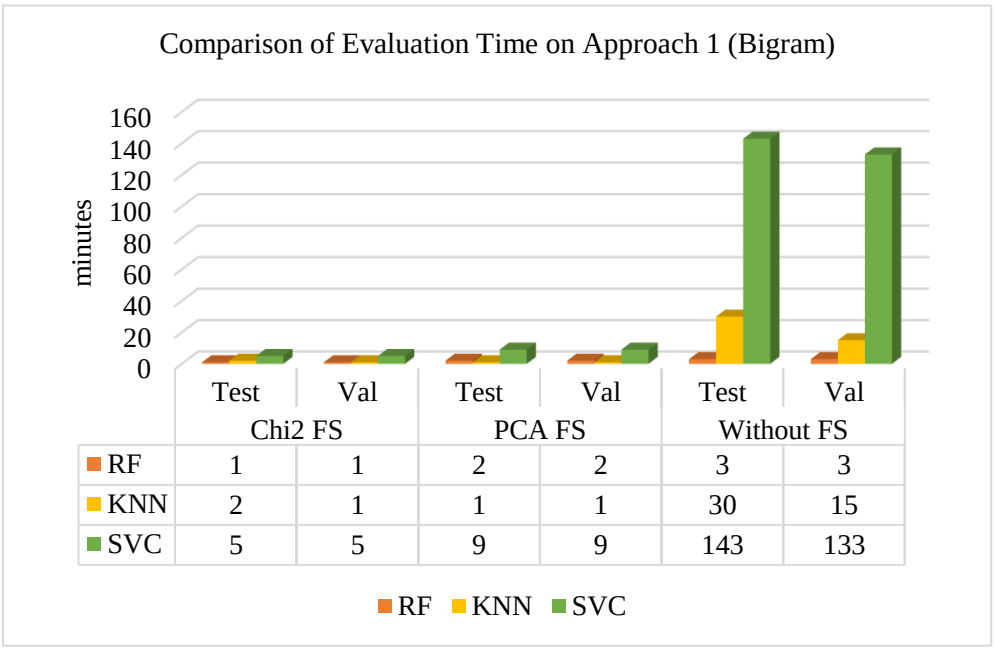

Figure 6.34 The Comparison of Evaluation Time for Approach 1 (bigram)

The evaluation time takes nearly three hours for approach 1 (API) and approach 2 (API_DLL) bigram.

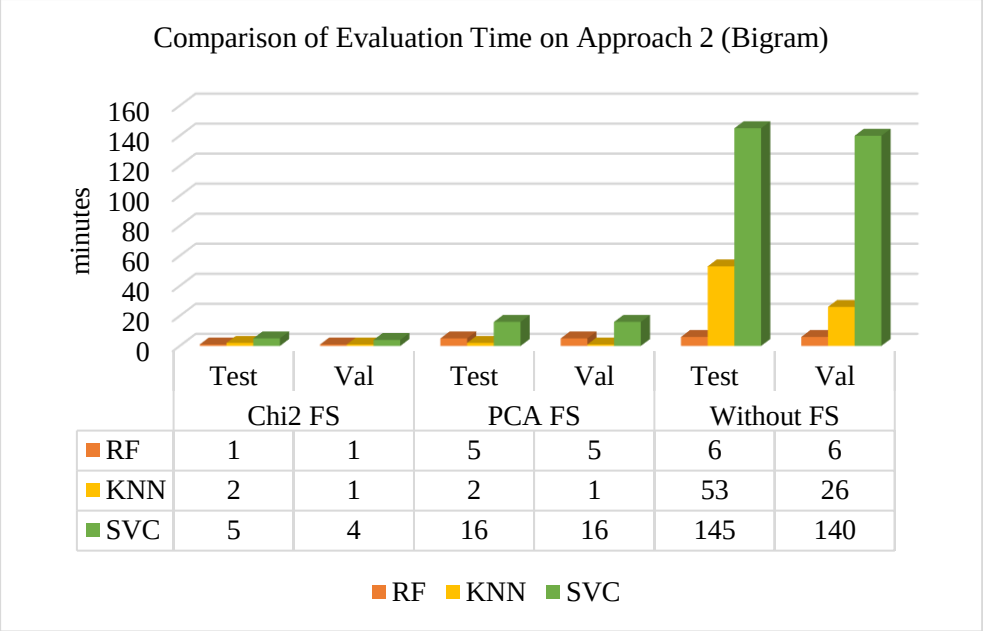

Figure 6.35 The Comparison of Evaluation Time for Approach 2 (bigram)

The below Figure 6.36 describes the processing time of approach 3 (bigram). With the aid of the feature selection method, the processing time totally decreases from hours to minutes in the SVC classifier. Moreover, the other two classifiers also decrease the processing time when feature selection is used. The use of RF and Chi2 together provides the fastest processing time not only for approach 2 (bigram) and also for approach 3 (bigram) in the BMFC dataset.

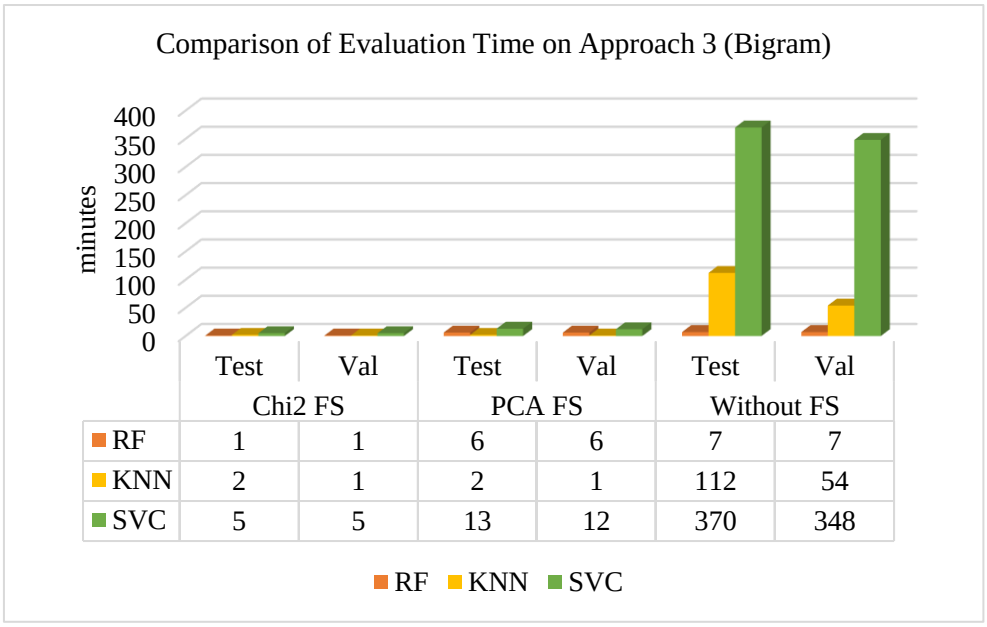

Figure 6.36 The Comparison of Evaluation Time for Approach 3 (bigram)

6.4.4 Evaluation on BMC Dataset for Unigram

In the Benign Malware Classification (MBC) dataset, it contains 3000 benign samples from dataset-1 and 3579 new malicious samples from 18 families. This section describes the experimental results of dataset-2 on unigram features. This research uses the accuracy, confusion matrix (CM), TPR, FPR and ROC curve as evaluation metrics.

Table 6.22 Train-Validation-Test Shape for Experiments

Data Shape	# of instances
Train Shape	5328
Validation Shape	593
Test Shape	658

6.4.4.1 Evaluation on Approach 1 (API)

Table 6.23 describes the comparison of accuracy on dataset-2 unigram with approach 1 (API). The table shows that the selected features using Chi2 with kNN provide better accuracy than the other two classifiers and without using the feature selection method in the test set. The RF classifier also achieves accuracy better than SVC with selected features.

Table 6.23 Accuracy (%) on Approach 1 (API/unigram)

Classifiers	Chi2 FS (300 selected features)		PCA FS (300 selected features)		Without FS (332 features)	
	Val	Test	Val	Test	Val	Test
RF	94.77	95.13	93.92	**96.35**	93.92	95.13
kNN	93.92	93.92	94.43	94.22	93.92	94.52
SVC	92.75	92.55	92.58	92.55	92.91	92.4

Besides accuracy, TPR, and FPR have been used to measure the classifier performance in this research. The section provides the ROC and CM for the experiment results of selected 300 features using PCA and Chi2. The following Figure 6.37 and Figure 6.38 illustrate the CM for RF classifier with Chi2 and PCA FS methods.

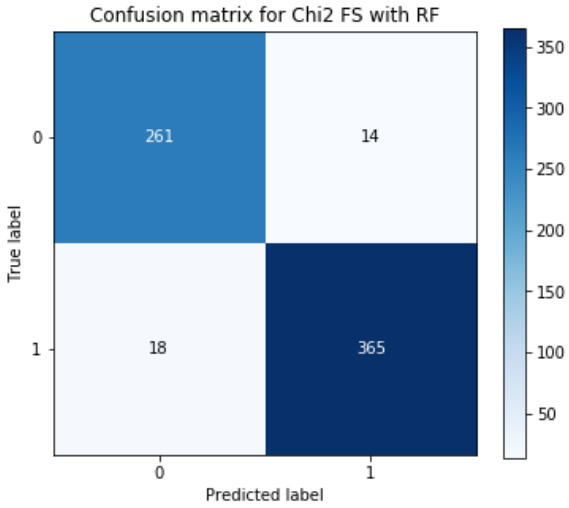

Figure 6.37 CM for Approach 1 using Chi2 FS with RF Classifier

The FN number of Clean family (class 0) is 14 and the FP number is 18 in Chi2 FS with RF classifier. Besides the CM of PCA provides 13 numbers of FN and 11 number of FP for approach 1. Therefore, the PCA feature selection method provides better FP and FN numbers than the Chi2 feature selection method in this approach. Thus, the accuracy of the PCA feature selection method in the RF classifier provides better than the Chi2 feature selection method.

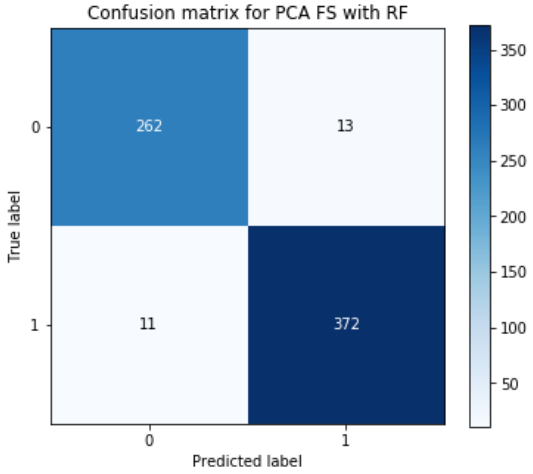

Figure 6.38 CM for Approach 1 using PCA FS with RF Classifier

According to the figures mentioned above, the PCA FS method provides better accuracy and FPR than the Chi2 FS method. The following Table 6.24 provides TPR and FPR on the test set for approach1. RF classifier with PCA FS provides lower FPR than Chi2 FS. The accuracy, TPR, and FPR of Chi2 FS are equaled to those of without using the feature selection approach in the last column.

Table 6.24 TPR/FPR for Approach 1 on Test Set

Classifiers	Chi2 FS TPR/FPR	PCA FS TPR/FPR	Without FS TPR/FPR
RF	0.95/ 0.0489	0.96/ 0.0379	0.95/ 0.0489
kNN	0.94/0.0599	0.94/ 0.0567	0.94/0.0552
SVC	0.92/ 0.0757	0.92/ 0.0757	0.92/ 0.077

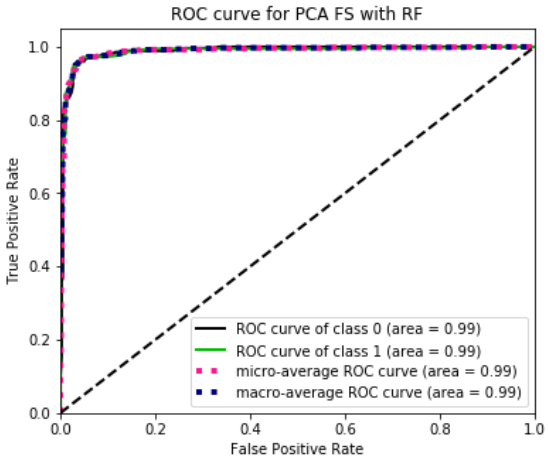

Figure 6.39 ROC curve for Approach 1 using PCA FS with RF Classifier

6.4.4.2 Evaluation on Approach 2 (API_DLL)

Table 6.25 shows the comparison of the accuracy of approach 2 on unigram for the MBC dataset. The approach 2 contains total of 704 features. And the 300 selected features provide the best accuracy and low FPR with this approach.

In this approach, the RF classifier also outperforms than the other two classifiers before and after applying the FS methods. The kNN leads to the second-best classifier for the API_DLL malicious or not classification dataset.

Table 6.25 Accuracy (%) on Approach 2 (API_DLL/unigram)

Classifiers	Chi2 FS (300 selected features)		PCA FS (300 selected features)		Without FS (704 features)	
	Val	Test	Val	Test	Val	Test
RF	95.11	**96.5**	94.94	95.13	94.6	96.35
kNN	94.44	94.98	94.09	94.83	94.09	94.37
SVC	92.58	91.64	94.26	92.55	93.59	92.4

The below Figure 6.40 provides the CM for API_DLL (approach2) on test data using RF classifier with Chi2 FS method.

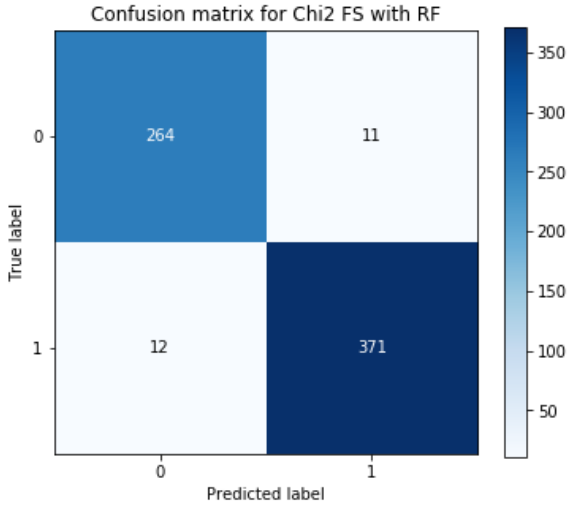

Figure 6.40 CM for Approach 2 using Chi2 FS with RF Classifier

The Chi2 FS method with RF classifier provides lower FP number than the all features with RF classifier. However, the TP number of Clean family (class 0) in all features is better than the Chi2 FS method with RF classifier.

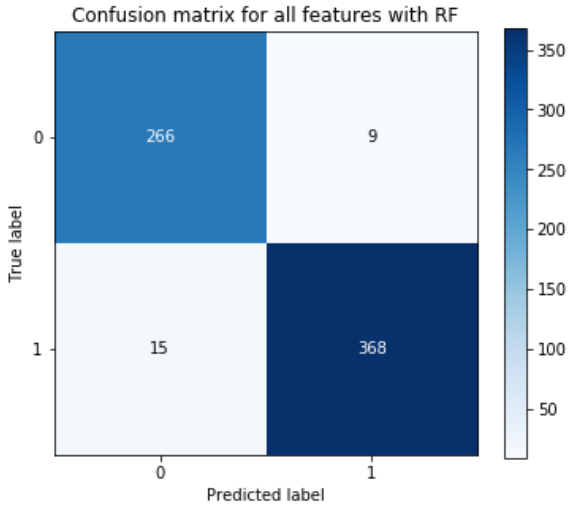

Figure 6.41 CM for Approach 2 on all features using RF Classifier

The applying of FS method in this approach provides lower FPR rather than without applying FS method. The FPR in SVC classifier is higher than the other classifiers but the RF classifier provides low FPR after applying the Chi2 FS method. Moreover, the RF classifier also provides lower FPR on extracted features before applying the feature selection method.

Table 6.26 TPR/FPR for Approach 2 on Test Set

Classifiers	Chi2 FS TPR/FPR	PCA FS TPR/FPR	Without FS TPR/FPR
RF	0.96/ 0.035	0.95/ 0.048	0.96/ 0.035
kNN	0.95/ 0.049	0.94/ 0.0515	0.94/ 0.056
SVC	0.915/ 0.085	0.92/ 0.075	0.92/ 0.078

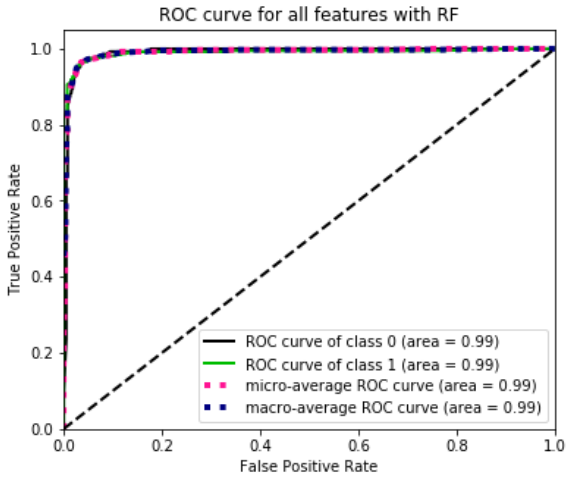

Figure 6.42 ROC curve for Approach 2 on all features using RF Classifier

Therefore, the extracted prominent features provide lower FPR and higher TPR according to the Table 6.26. The experimental results of RF classifier provide same TPR and FPR for Chi2 FS and all features without using FS. Thus, the scores of ROC area are not significantly changed in this approach.

The processing time does not take too much to train and evaluate the classifier as the number of total features are smaller than the approach 3 on unigram and all approach on bigram.

6.4.4.3 Evaluation on Approach 3 (API_DLL_PROCESS)

This section provides the experimental results of approach 3 on selected 1000 features. The following table shows the accuracy for API_DLL_PROCESS using three classifiers on unigram BMC dataset. Here, the validation set and test set of RF classifier with Chi2 FS provide better accuracy than other classifiers. According to the CM of experimental results, the correctly classified instance number in RF classifier is greater than the other classifiers.

Table 6.27 Accuracy (%) on Approach 3 (API_DLL_PROCESS /unigram)

Classifiers	Chi2 FS (1000 Selected features)		PCA FS (1000 Selected features)		Without FS (6284 features)	
	Val	**Test**	**Val**	**Test**	**Val**	**Test**
RF	95.78	**96.65**	95.44	94.37	94.94	95.44
kNN	94.6	94.22	93.92	94.98	93.92	94.52
SVC	94.43	93.16	94.09	93	93.59	93.46

The SVC classifier support higher FN and FP number than kNN classifier. Therefore, kNN classifier provides the second-best accuracy for malicious or not classification in this approach. The following figures have provided the comparison of CM for all classifiers using Chi2 FS method.

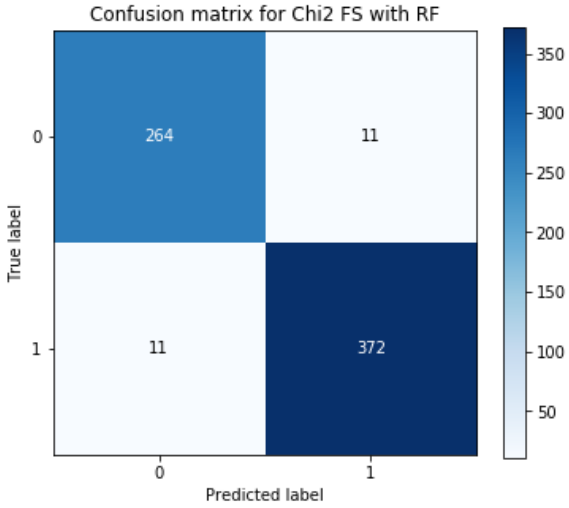

Figure 6.43 CM for Approach 3 using Chi2 FS with RF Classifier

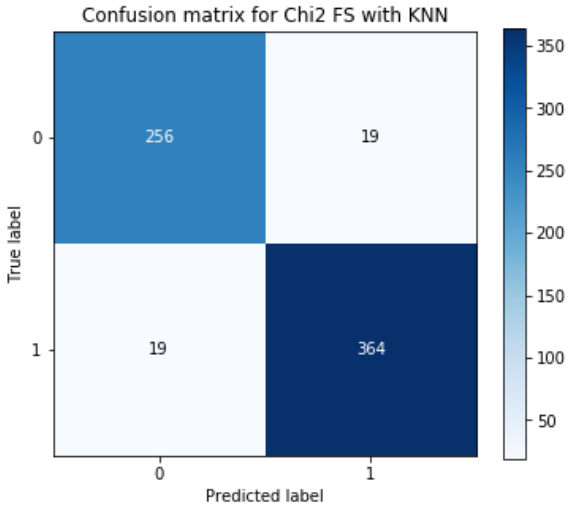

Figure 6.44 CM for Approach 3 using Chi2 FS with kNN Classifier

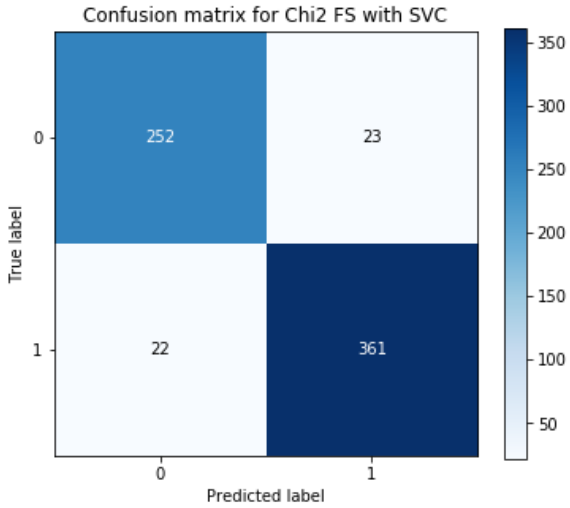

Figure 6.45 CM for Approach 3 using Chi2 FS with SVC Classifier

The RF classifier provides better TPR on both class label than others. The Chi2 feature selection with RF classifier provides the best scores in TPR and FPR than other classifiers.

Table 6.28 TPR/FPR for Approach 3 on Test Set

Classifiers	Chi2 FS TPR/FPR	PCA FS TPR/FPR	Without FS TPR/FPR
RF	**0.96/ 0.034**	0.94/ 0.058	0.95/ 0.045
kNN	0.94/ 0.059	0.94/ 0.05	0.94/ 0.054
SVC	0.92/ 0.07	0.92/ 0.071	0.93/ 0.067

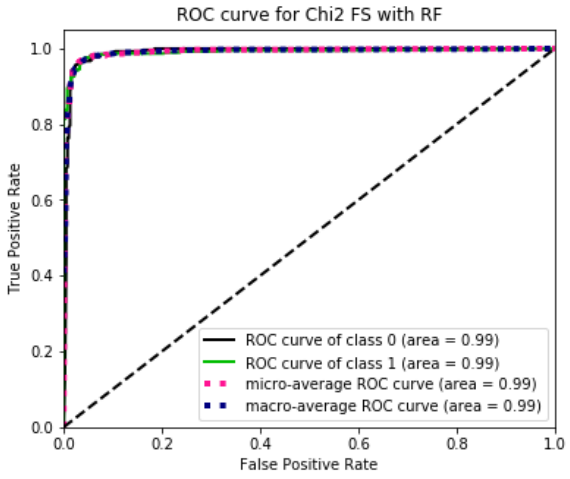

Figure 6.46 ROC curve for Approach 3 using Chi2 FS with RF Classifier

The kNN classifier supports TPR and FPR better than SVC classifier on all

features with and without using feature selection method. And the kNN also provides better accuracy than the SVC classifier. The following figure provides the ROC curves of RF classifiers using Chi2 feature selection method.

6.4.5 Evaluation on BMC Dataset for Bigram

This section covers the experimental results of dataset-2 on bigram features. The evaluation metrics are accuracy, ROC curve, and confusion matrix (CM) in this experiment. The combination of 2-gram sequence in API approach provides better accuracy than unigram (1-gram).

6.4.5.1 Evaluation on Approach 1 (API)

In this case, the approach 1 contains the 16222 features and 6579 instances. These instances are divided into three parts, train (5328), validation (593), and test (658) sets. The feature tuning is then performed to get the best performance. The 1000 features provide the best performance for the MBC dataset.

Table 6.29 Accuracy (%) on Approach 1 (API /bigram)

Classifiers	Chi2 FS (1000 selected features)		PCA FS (1000 selected features)		ALL Features (16222 features)	
	Val	Test	Val	Test	Val	Test
RF	95.27	**96.96**	93.08	93.61	94.6	94.98
kNN	94.43	93.92	92.74	93.16	92.24	92.24
SVC	92.91	94.37	93.42	94.07	93.76	92.85

The above table describes the accuracy of bigram on approach 1 (API). The table shows that the selected features using Chi2 FS method provides better accuracy than PCA FS method and without using feature selection method. The SVC classifier provides better accuracy than the kNN in this test dataset.

The following figures have provided the CM for all classifiers using Chi2 FS method. The RF classifier provides more correct prediction on both benign and malware than the other classifiers. The kNN classifier provides better correct prediction on benign samples than SVC classifier. However, the SVC classifier provides better prediction on malware samples than kNN classifier.

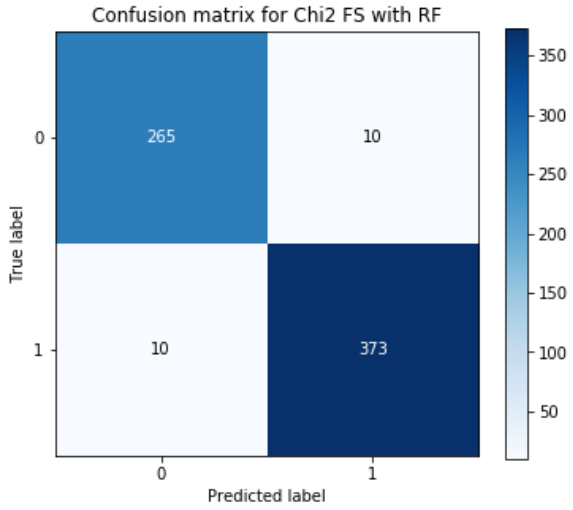

Figure 6.47 CM for Approach 1 using Chi2 FS with RF Classifier

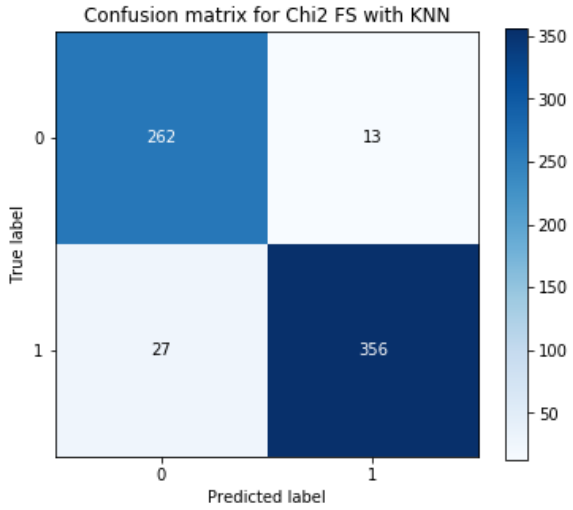

Figure 6.48 CM for Approach 1 using Chi2 FS with kNN Classifier

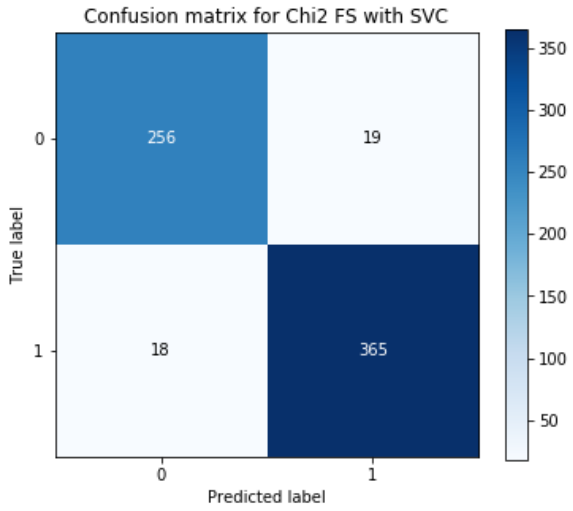

Figure 6.49 CM for Approach 1 using Chi2 FS with SVC Classifier

The following table describes the TPR and FPR of the experiment on Approach 2 using three different classifiers.

Table 6.30 TPR/FPR for Approach 1 on Test Set

Classifiers	Chi2 FS TPR/FPR	PCA FS TPR/FPR	Without FS TPR/FPR
RF	**0.96/ 0.031**	0.92/ 0.071	0.95/ 0.049
kNN	0.94/ 0.058	0.93/ 0.064	0.92/ 0.072
SVC	0.94/ 0.058	0.93/ 0.062	0.92/ 0.075

The RF classifier with Chi2 FS in approach 2 also provides lower FPR and higher TPR than other classifiers. The RF classifier also produces low FPR in all features.

The combination of PCA FS with all classifiers, the SVC supports lower FPR than RF and kNN classifiers. However, the SVC and kNN classifier have produced equal TPR in PCA FS method.

Without applying FS method, the RF classifier supports lower FPR and higher TPR than kNN and SVC classifiers.

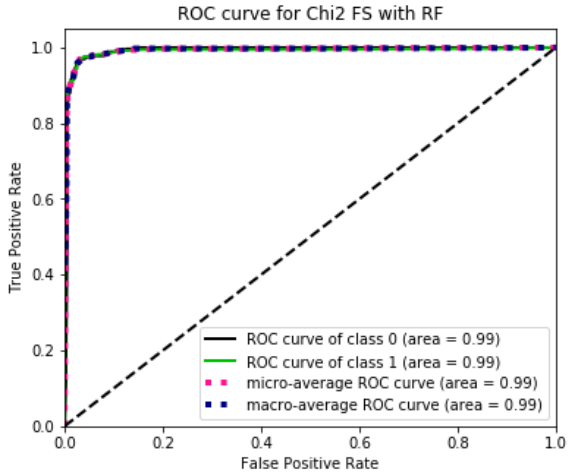

Figure 6.50 ROC curve for Approach 1 using Chi2 FS with RF Classifier

The RF classifier provides better ROC score than kNN and SVC classifiers with Chi2 FS method.

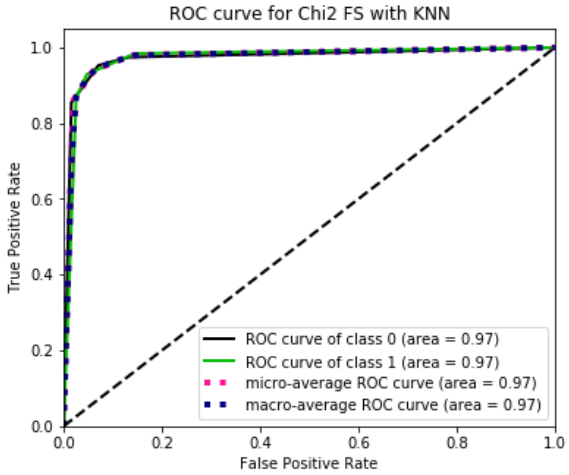

Figure 6.51 ROC curve for Approach 1 using Chi2 FS with kNN Classifier

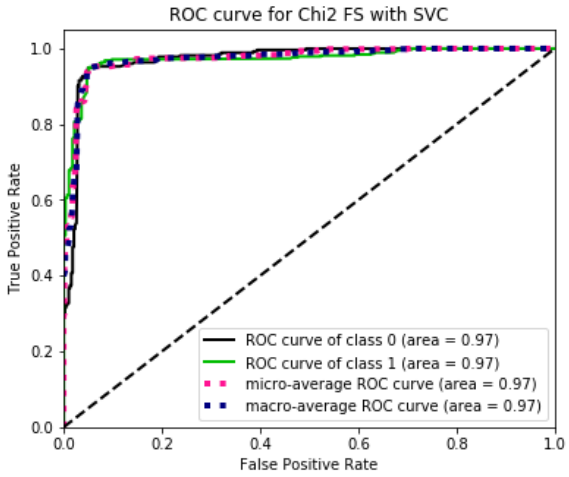

Figure 6.52 ROC curve for Approach 1 using Chi2 FS with SVC Classifier

The ROC curves provide same area score in SVC and kNN classifiers since the TPR and FPR are same in these classifiers.

6.4.5.2 Evaluation on Approach 2 (API_DLL)

This section describes the experiment of approach 2 on bigram MBC dataset. The following table shows the accuracy of API_DLL bigram MBC dataset using three different classifiers. In approach2 on bigram, the RF also outperforms than the other two classifiers before and after applying the FS methods.

The accuracy of SVC classifier is better than the kNN on all test sets in API_DLL bigram dataset. The accuracy of RF classifier on all features provides better than the PCA feature selection method on test dataset.

Table 6.31 Accuracy (%) on Approach 2 (API_DLL /bigram)

Classifiers	Chi2 FS (1000 selected features)		PCA FS (1000 selected features)		Without FS (18568 features)	
	Val	Test	Val	Test	Val	Test
RF	95.78	**96.96**	92.91	94.98	95.27	95.89
kNN	93.93	94.07	92.58	92.7	92.91	91.64
SVC	93.42	94.22	93.25	94.22	92.74	93.92

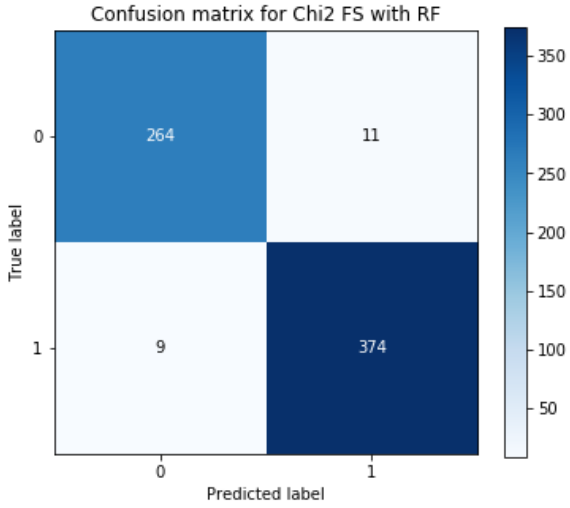

Figure 6.53 CM for Approach 2 using Chi2 FS with RF Classifier

The values of TPR and FPR have been described in the following table for approach 2 on bigram. The RF classifier with Chi2 FS method always outperforms in accuracy, TPR and FPR than other classifiers and PCA FS method.

Table 6.32 TPR/FPR for Approach 2 on Test Set

Classifiers	Chi2 FS TPR/FPR	PCA FS TPR/FPR	Without FS TPR/FPR
RF	0.96/ 0.031	0.94/ 0.057	0.95/ 0.04
kNN	0.94/ 0.057	0.93/ 0.069	0.91/ 0.08
SVC	0.94/ 0.059	0.94/ 0.059	0.93/ 0.064

The ROC curves have been described from the experiment of RF classifier with Chi2 FS method.

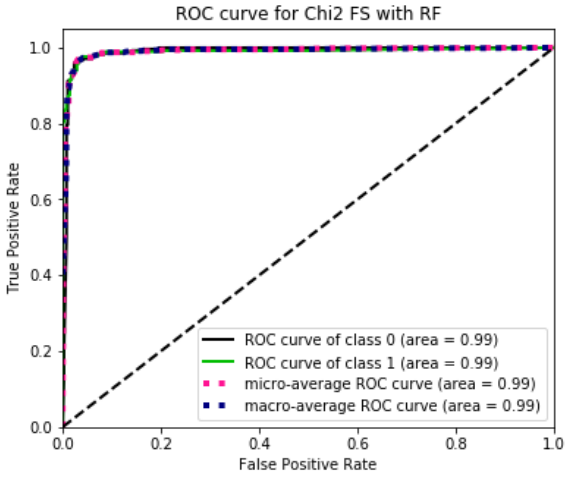

Figure 6.54 ROC curve for Approach 2 using Chi2 FS with RF Classifier

The experimental results from API_DLL provide the same accuracy as API in bigram. And the ROC score goes to the same as API and the total number of misclassified instance number also does in RF with Chi2 FS method.

6.4.5.3 Evaluation on Approach 3 (API_DLL_PROCESS)

The following table shows the accuracy between validation set and test set using three ML classifiers. The results are provided before and after applying the FS methods.

Table 6.33 Accuracy (%) on Approach 3 (API_DLL_PROCESS /bigram)

Classifiers	Chi2 FS (1000 selected features)		PCA FS (1000 selected features)		Without FS (27915 features)	
	Val	Test	Val	Test	Val	Test
RF	95.11	**96.5**	91.9	94.37	94.94	95.13
kNN	93.76	93.47	92.58	93.31	92.58	92.4
SVC	92.24	93.77	94	94.68	91.23	93.61

In approach3 on bigram, the Chi2 FS with RF also outperforms than the other two classifiers. The test set that used the PCA FS method provides less accuracy than all features with RF classifier. However, the approach 1 and 2 of bigram outperforms than this approach on MBC dataset.

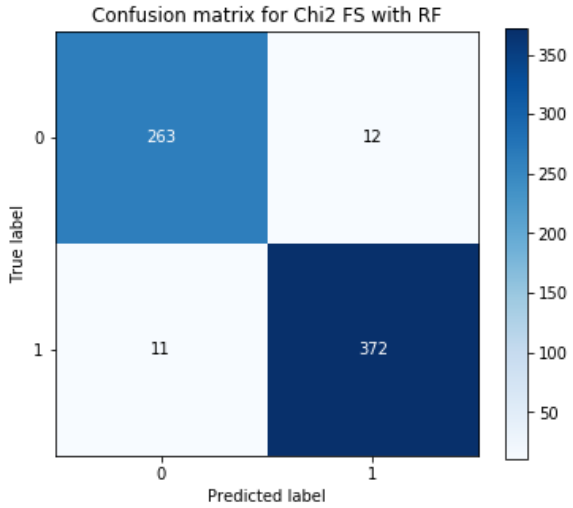

Figure 6.55 CM for Approach 3 using Chi2 FS with RF Classifier

The TPR and FPR of BMC on approach 3 have been described in the following table.

Table 6.34 TPR/FPR for Approach 3 on Test Set

Classifiers	Chi2 FS TPR/FPR	PCA FS TPR/FPR	Without FS TPR/FPR
RF	0.96/ 0.036	0.936/ 0.063	0.95/ 0.049
kNN	0.93/ 0.065	0.933/ 0.066	0.92/ 0.073
SVC	0.93/ 0.064	0.94/ 0.055	0.93/ 0.065

The RF classifier with Chi2 FS method continuously outperform in accuracy, TPR and FPR than other classifiers. The ROC curve has been described from the experiment of RF classifier with Chi2 FS method.

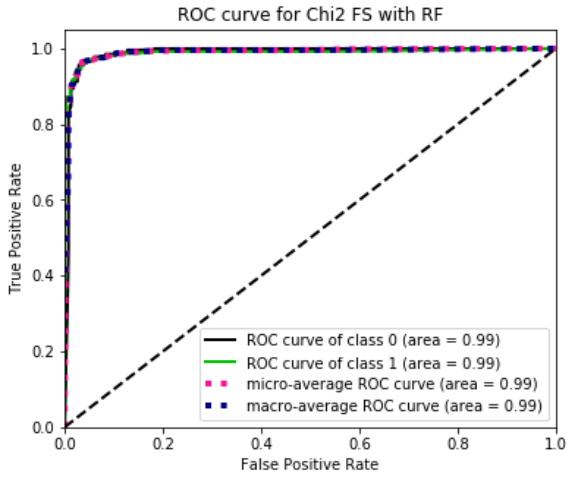

Figure 6.56 ROC curve for Approach 3 using Chi2 FS with RF Classifier

By differentiating the CM of all classifiers, the class label 0 (benign) of RF classifier provides more correct prediction than other classifiers. RF also provides better correct prediction on class label 1 (malware) than others. This section provides all experiment results of bigram API_DLL_PROCESS on MBC dataset.

6.5 Comparison of Accuracy for All Approaches

Table 6.35 shows the comparison of accuracy for RF classifier with Chi2 feature selection on all approaches.

Table 6.35 Accuracy (%) of All Approaches for Chi2 FS with RF Classifier

Approach / n-gram	Approach1 (API)	Approach2 (API_DLL)	Approach3 (API_DLL_PROCESS)
Unigram (BMFC)	98.16	**98.4**	98.21
Bigram (BMFC)	98.4	**98.53**	98.49
Unigram (BMC)	95.13	96.5	**96.65**
Bigram (BMC)	**96.96**	**96.96**	96.5

The RF classifier performs better accuracy than other classifiers. In BMFC dataset, the approach 2 on bigram provides the best classification accuracy. Although the accuracy is not improved significantly, the bigram is better than unigram sequence on all approaches. Moreover, the combination of API and DLL features support the best accuracy in BMFC dataset. In BMC dataset, the accuracy of bigram sequence is better than unigram sequence except approach 3.

6.6 Summary

This chapter provides the experimental results of the proposed malicious software classification system. The malicious or not classification has been provided by using the malware and benign samples to identify the incoming executable files is whether malicious or not. Then, the family classification has been performed by providing the five different families including benign samples. According to the experimental results, it has been discovered that the bigram sequence provides better accuracy than unigram on all approaches. Moreover, it has been provided the best accuracy on both datasets (BMFC and BMC) too. However, it is shown that the

approach 2 (API_DLL) on bigram is slightly better than the approach 3 (API_DLL_PROCESS) on bigram.

The findings from the experiments of dataset-1 (BMFC) and dataset-2 (BMC) are described as follows:

1. The accuracy is improved when the API features are combined with the DLL features. The API_DLL (Approach 2) features provide better results than others on both n-gram sequences (unigram & bigram) and both datasets (BMFC & BMC).
2. API_DLL_PROCESS (Approach 3) provides slightly better accuracy than API features only (Approach 1). However, the accuracy of API, DLL and Process features combination is not as good as the accuracy of API and DLL features combination.
3. The bigram sequence outperforms than the unigram sequence on all approaches and also for all datasets.
4. For dataset-2 (MBC) on bigram sequence, Approach 1 provides slightly better accuracy than the accuracy of Approach 3.
5. The combination of Chi2 FS with RF classifier supports the best results for all approaches and all datasets.
6. The processing time of the unigram sequence is quite faster than the bigram sequence for training, validation and testing set.
7. The processing time of approach 2 and 3 took nearly 3 and 6 hours respectively on the bigram of the BMFC dataset.
8. The Zbot (class 1) is the kind of Zeus malware and generally, it is also a kind of Trojan (class 3) type. Therefore, this kind of family does not provide accurate classification results completely since it is one of the parts of Trojan malware.

CHAPTER 7

CONCLUSION

The technology has increasingly changed much faster than ever before. Machine Security is the greatest importance security for all associations and society. To cope with the technology, the defense techniques are needed to build by using Machine Learning and Cyber Security approaches. There are multiple ways in which ML can be used in Cyber Security. The detection systems such as NIDS/HIDS, Email Detection, Malware Detection and Classification, Website Phishing Attack Detection, and other detection can be developed by using the Machine Learning in Cyber Security. The online advertising, consumer profiling, recommendation systems, and many other Internet-related businesses significantly depend on data analysis and the underlying methods of ML. It can extract useful information from unstructured and structured data. Regrettably, the evaluation of Internet, Smart Technologies and Smart Devices have also encouraged the cybercriminals to abuse and commit the sophisticated cyber-crimes. It enables cybercriminals to develop sustainable businesses that depend on the exploitation of security vulnerabilities. To evade the security detection systems, the cybercriminals use the new exploitation techniques. To build the development of suitable defenses, ML is used to extract information from extraordinary amounts of security data.

ML can be applied in many cyber security areas and it is an effective technique. It can also be effectively used to develop the authentication systems, to evaluate the protocol implementation, to assess the security of human interaction proofs for smart meter data profiling, etc. There are numerous opportunities in Information Security to apply ML to highlight various challenges in such complex domain.

The virus detection is one of the areas of cyber security together with ML techniques. However, advanced threats can easily bypass or evade traditional signature-based virus detection systems/tools. And the polymorphic, metamorphic malicious can bypass these traditional detection tools and systems very easily. The performance of the ML model will be improved with the use of more and latest datasets. However, ML can apply to create new malicious variants, target exact victims and generate sensitive data, exploit zero-day vulnerabilities and guard the attackers' own infrastructure.

Moreover, the organizations that applied the ML solutions can also become the targets of the criminals. The criminals or intruders can create, insert or manipulate the poisoned data into organizational data, or inappropriate decisions into beneficial systems. It is hard to say how and when the organizations that applied ML solutions will be victims by the attacker. Therefore, defense systems are essential to prevent cyber-attacks in any organization.

The key components of the proposed system compose of extracting the malware samples name from VirusTotal and naming and categorizing these samples by using regression regular expression theory, applying the n-gram sequence on extracted features in postprocessing and selecting the most important features from the n-gram sequences. Moreover, 2-gram API_DLL call sequences are enough to classify malicious behaviors using the Chi2 feature selection method. The proposed system provides the best performance metrics than previous research works.

The effectiveness of extracted features by using the proposed MFEA covers both benign-malware classification and benign-malware family classification datasets to classify the malicious or not classification and malware family classification. The correct classification between various malicious families and benign has been exposed that the correctness of the proposed NMS_RE technique on classification.

Moreover, the different experimental results have been discovered the effectiveness of BMC and BMFC with the used of proposed NMS_RE and MFEA on unigram and bigram sequence malware-benign classification. The experiment results show that classification on both datasets have been performed well and effective on both datasets without applying the feature selection approaches. Furthermore, the used of bigram sequence in malware family classification and malicious or not classification have been produced better performance than unigram sequence on all approaches and all datasets.

7.1 Thesis Summary

Due to the rapid growth of adopting new variant technologies to evade existing AV or anti-malware detection systems, further research is still needed to detect the serious targeted attacks. The classification and detection of malicious files are still ongoing research for security researchers and analysts because of the emerging of new suspicious variants. It is till never end game between attackers and analysts due to the

arising of new variants daily.

To study and learn the behavior and nature of malware patterns, this research has been explored by proposing the Malware Feature Extraction Algorithm (MFEA) in the malware classification research area. Therefore, the proposed system has developed an applicable BMFC and BMC datasets by contributing effective Malware Feature Extraction Algorithm (MFEA) for malware classification. In this research, a benign malware classification (BMC) and benign malware family classification (BMFC) datasets have been developed with the aim of distinguishing new malware variants and to provide new malicious patterns for further researches.

The efficiency of extracted features by using the proposed MFEA covers both datasets to handle the malicious or not and family classification. The correct classification between various families and benign has revealed the correctness of the proposed NMS_RE technique on classification.

Additionally, the different experimental results have discovered the effectiveness of BMFC and BMC with the use of proposed NMS_RE and MFEA on unigram and bigram sequence for malware classification. The experimental results show that both classification systems performed well and effective even without using the feature selection methods on both datasets.

Besides, the use of a bigram sequence in malware classification has provided better accuracy than the unigram sequence on all approaches and all datasets. But, the processing time of bigram in BMFC dataset takes more time than approach 1 and approach 2 of the unigram sequence. The size of the 2-gram feature dataset is very large to handle the classification. The approach 3 of unigram in BMFC also takes more processing time than others of unigram.

However, this problem can be solved by using feature selection methods. Therefore, the feature reduction methods are applied in this research because the feature reduction is one of the critical parts of the high dimension data in classification for performance and accuracy. This system has applied FS methods such as Chi2, PCA to select the dominant features. These main features help to identify the new unfamiliar variants of malicious behaviors in real-world applications. Two feature selection approaches, Chi2, and PCA, have been conducted to cut down the number of features, especially for all approaches in bigrams. Therefore, the selected features have provided better results for classification especially on bigram. There are two benefits of applying

the FS method in this research.

1. It is shown that the processing time has totally been reduced from hours to minutes after applying the FS method especially on approach 3 unigram and all approaches of bigram.
2. It is revealed that the accuracy of the classification system has also been improved in all classifiers after applying FS. The combination of Chi2 and RF has provided better performance than the others.

The usage of ML techniques in cyber security is increasing more and more ever before. The proposed system used the three ML methods for multi-class (four different malware types vs benign) and binary class (malware vs benign) classification. The proposed system evaluates the performance of the classifier by using Accuracy, Confusion Matrix (CM), True Positive Rate (PTR), False Positive Rate (FPR), and ROC curves. Moreover, the system also provides low FP and FN rates on malware and benign classification.

Therefore, the BMFC and BMC systems are able to classify the samples correctly. The BMFC system composes of 5 families including benign. The result shows that selected prominent features achieved the best accuracy of 98.53% on Approach 2 bigram (API_DLL) BMFC dataset and 96.96% on Approach 2 (API_DLL) bigram BMC dataset.

7.2 Advantages and Limitations of the Proposed System

This proposed BMFC and BMC datasets perform well and effective before and after applying the feature selection on three different approaches using three ML classifiers. The approaches on bigram perform better than the approaches on unigram as it has been revealed in the previous chapter.

In the aforementioned research articles in Chapter 2, APIs are mostly used to generate the features. The related works mentioned in Chapter 2 have some weaknesses in the small number of analyzed samples and high FPR in classification and detection system although some related works yield high accuracy. The proposed BMFC and BMC datasets are able to classify the samples correctly. And the system provides the best accuracy of 98.53% on Approach 2 of BMFC dataset and 96.96% on Approach 2 of the bigram MBC dataset.

The experimental results show that the bigram sequence outperforms than the

unigram sequence on all approaches and all datasets. Besides, it is proved that the processing time absolutely reduces from hours to minutes after conducting the FS method.

Additionally, the BMFC and BMC datasets have the ability of scalability by adding the new features that found from new malware analysis. Both datasets are simple to extend by updating recently found new features into the current datasets.

Although the use of virtualization techniques for analysis is great for budget concerned, there is a problem with using these techniques. New complex malware sometimes might try to detect VM and stop working on it. If the malicious file detects it is being run on a VM, it will not execute. When malware number is increased in the wild with million, the number of test samples in this work is not very large. However, the number of samples in this experiment provide larger than the recent related works.

7.3 Future Work

The benign-malware classification (dataset-2) has used from 18 different families for malicious or not classification but the number of samples is quite small. The benign-malware family classification (dataset-1) has used 5 different families including benign for family classification system due to the concerned of unbalanced class. Therefore, the sample number and family number seem quite small but it does not at all when comparing with the related works described in Chapter 2. Therefore, multiple malicious families like dataset-2 will also be used to classify the relevant families in the future.

The additional feature specification is needed for some malware families to correctly classify the malicious samples by adding the arguments of malicious features. The in-depth family classification is still needed for some family like Trojan, Adware, Worm, and Ransomware since malware analysis and classification are ongoing research to handle the incident response and variants of malware are increasing yearly.

By extending the current work, the unidentified malware and zero-day attack detection system will be developed for the detection system in future work. The unknown variant of malware types can be classified with the use of proposed benign-malware feature datasets by differentiating the malicious features of each family.

The better performance results will be generated by performing more data and more experiments in the future. Learning methods such as Machine Learning, Deep

Learning, and Big Data Analysis on Malware Classification and Detection System will be kept exploring.

AUTHOR'S PUBLICATIONS

[P1] Cho Cho San, Mie Mie Su Thwin "Proposed Applicable Framework for Extracting Rootkits Features and Clustering through Dynamic Analysis for Incident Handling Systems", 15th International Conference on Computer Applications (**ICCA, 2017**), Yangon, MYANMAR February 2017. **Page [342-350]**

[P2] Cho Cho San, Mie Mie Su Thwin "Effective Malicious Feature Extraction and Classification using Random Forest Machine Learning Classifier", 15th International Conference on Computer Applications (**ICCA, 2018**), Yangon, MYANMAR February 2018. **Page [265-272]**

[P3] Cho Cho San, Mie Mie Su Thwin, Naing Linn Htun "Malicious Software Family Classification using Machine Learning Multi-class Classifiers", In Computational Science and Technology: 5th ICCST 2018, August 2018. **Lecture Notes in Electrical Engineering (Scimago index), vol 481, Page [423-433]**

[P4] Cho Cho San, Mie Mie Su Thwin "Selecting Prominent API Calls and Labeling Malicious Samples for Effective Malware Family Classification", International Journal of Computer Science and Information Security (**IJCSIS, 2019**), Vol. 17 No. 5 MAY. **Page [89-105]** Pittsburgh, PA, 15213, USA. **(Scopus index)**

[P5] Cho Cho San, Mie Mie Su Thwin "Proposed Effective Feature Extraction and Selection for Malicious Software Classification", Advances in Biometrics, Springer Nature, 2019.

BIBLIOGRAPHY

[1] T. Abou-Assaleh, N. Cercone, V. Keselj, R. Sweidan, “N-gram based detection of new malicious code,” Proc. 28th Annu. Int. Comput. Softw. Appl. Conf. 2004. COMPSAC 2004., vol. 2, no. 1, pp. 41–42, 2004.

[2] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, G. Giacinto, “Novel feature extraction, selection and fusion for effective malware family classification”. In Proceedings of the sixth ACM conference on data and application security and privacy (pp. 183-194). ACM, March, 2016.

[3] B. A. AlAhmadi, I. Martinovic, “MalClassifier: Malware family classification using network flow sequence behavior”. In 2018 APWG Symposium on Electronic Crime Research (eCrime) (pp. 1-13). IEEE, May, 2018.

[4] T. Ambwani, “Multi Class support vector machine implementation to intrusion detection”, The international joint conference on neural networks, vol. 3, pp. 2300–2305, July 2003.

[5] U. Baldangombo, N. Jambaljav, S.-J. Horng, “A static malware detection system using data mining methods”. arXiv preprint arXiv:1308.2831, 4(4), 113–126, 2013.

[6] S. Banin, G. O. Dyrkolbotn, “Multinomial malware classification via low-level features”. Digital Investigation, 26, S107-S117, 2018.

[7] G. N. Barbosa, R. R. Branco, “Prevalent Characteristics in Modern Malware”. Black Hat USA 2014.

[8] M. Belaoued, S. Mazouzi, “A Chi-Square-Based Decision for Real-Time Malware Detection Using PE-File Features”. Journal of Information Processing Systems, 12(4), 2016.

[9] L. Breiman, “Random Forests”. Statistics Department, University of California, Berkeley, CA 94720

[10] C. Cepeda, D. L. C. Tien, & P. Ordóñez, “Feature selection and improving classification performance for malware detection”, In 2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and

Communications (SustainCom)(BDCloud-SocialCom-SustainCom) pp560-566, IEEE, October 2016.

[11] C. C. Chang, C. J. Lin, “LIBSVM: a library for support vector machines”, ACM transactions on intelligent systems and technology (TIST), Vol. 2, No. 3, pp27, 2011.

[12] J.Y.C. Cheng, T.S. Tsai, C.S. Yang, “An information retrieval approach for malware classification based on Windows API calls”. Machine Learning and Cybernetics (ICMLC), 2013 International Conference on. IEEE, 4: 1678-1683, 2013.

[13] C.I. Fan, H.W. Hsiao, C.H. Chou, Y.F. Tseng, “Malware Detection Systems Based on API Log Data Mining,” in 2015 IEEE 39th Annual Computer Software and Applications Conference, pp. 255–260, 2015.

[14] P. Faruki, V. Laxmi, M. S. Gaur, P. Vinod. “Mining control flow graph as api call-grams to detect portable executable malware”. In Proceedings of the Fifth International Conference on Security of Information and Networks. ACM, 2012.

[15] V. Ford, A. Siraj, “Applications of machine learning in cyber security”. In Proceedings of the 27th International Conference on Computer Applications in Industry and Engineering, October 2014.

[16] J. B. Fraley, J. Cannady, “The promise of machine learning in cybersecurity”. In SoutheastCon 2017 (pp. 1-6). IEEE, March, 2017.

[17] H. S. Galal, Y. B. Mahdy, M. A. Atiea, “Behavior-based features model for malware detection,” J. Comput. Virol. Hacking Tech., vol. 12, no. 2, pp. 59–67, 2016.

[18] E. Gandotra, D. Bansal, S. Sofat, “Malware analysis and classification: A survey”. Journal of Information Security, 5(02), p.56, April 2014.

[19] C. Guarnieri, A. Tanasi, J. Bremer, M. Schloesser, “The cuckoo sandbox”, 2012.

[20] G. Hackeling, “Mastering Machine Learning with scikit-learn”. Packt Publishing Ltd; Jul 24, 2017.

[21] K. S. Han, I. K. Kim, E. G. Im, “Malware classification methods using API sequence characteristics,” in Lecture Notes in Electrical Engineering, vol. 120 LNEE, pp. 613–626, 2012.

[22] O. Henchiri, N. Japkowicz, “A feature selection and evaluation scheme for computer virus detection”. In Sixth International Conference on Data Mining (ICDM'06) (pp. 891-895). IEEE, December, 2006.

[23] N. Horning, “Random Forests: An algorithm for image classification and generation of continuous fields data sets”. In Proceedings of the International Conference on Geoinformatics for Spatial Infrastructure Development in Earth and Allied Sciences, Osaka, Japan (Vol. 911), December, 2010.

[24] X. Hu, “Large-Scale Malware Analysis, Detection, and Signature Generation”. Diss. The University of Michigan, 2011.

[25] S. T. Ikram, A. K. Cherukuri, “Improving accuracy of intrusion detection model using PCA and optimized SVM”. Journal of computing and information technology, 24(2), 133-148, 2016.

[26] M. A. Jerlin, K. Marimuthu, “A new malware detection system using machine learning techniques for API call sequences”. Journal of Applied Security Research, 13(1), 45-62, 2018.

[27] A. Jerlin, C. Jayakumar, J. Prabhu, “EFE: Efficient Feature Extraction Algorithm for dynamic malware analysis in windows executables using API call sequence”, International Journal of Pharmacy and Technology, Vol 8, No (4), pp 25373-25383, 2016.

[28] I. T. Jolliffe, “Principle Component Analysis”, Springer-Verlag, New York, 1986.

[29] A. G. Kakisim, M. Nar, N. Carkaci, I. Sogukpinar, “Analysis and Evaluation of Dynamic Feature-Based Malware Detection Methods”. In International Conference on Security for Information Technology and Communications (pp. 247-258). Springer, Cham, November, 2018.

[30] N. Kaur, A. K. Bindal, “A Complete Dynamic Malware Analysis”. International Journal of Computer Applications (0975 – 8887), Volume 135, No.4, February 2016.

[31] K. Kendall, C. McMillan, “Practical malware analysis,” In Black Hat Conference, USA (p. 10), August, 2007.

[32] M. Khan, S. M. K. Quadri, “Effects of using filter based feature selection on the performance of machine learners using different datasets”. BVICA M's International Journal of Information Technology, 5(2), 597, 2013.

[33] Y. Ki, E. Kim, H. K. Kim, “A novel approach to detect malware based on API call sequence analysis,” Int. J. Distrib. Sens. Networks, vol. 2015, no. 6: 659101, pp. 1–9, 2015.

[34] B. Kolosnjaji, A. Zarras, G. Webster, C. Eckert, “Deep learning for classification of malware system call sequences”. In Australasian Joint Conference on Artificial Intelligence (pp. 137-149). Springer, Cham, (2016, December).

[35] J. Kolter and M. Maloof, “Learning to detect and classify malicious executables in the wild”. Journal of Machine Learning Research, 8(Dec):2755–2790, 2006.

[36] D. Komashinskiy, I. Kotenko, “Malware detection by data mining techniques based on positionally dependent features”, In 2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing, pp617-623, IEEE, February 2010.

[37] D. Kong, G.Yan, “Discriminant malware distance learning on structural information for automated malware classification”. In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 1357-1365). ACM, August 2013.

[38] A. Kumar, “A Framework for Malware Detection with Static Features using Machine Learning Algorithms”. (Doctoral dissertation, Department of Computer Science, Pondicherry University), (2017).

[39] J. Lee, K. Jeong, H. Lee, “Detecting metamorphic malwares using code graphs”. In Proceedings of the 2010 ACM Symposium on Applied Computing, SAC ’10. ACM, 2010.

[40] G. Liang, J. Pang, C. Dai, “A Behavior-Based Malware Variant Classification Technique,” Int. J. Inf. Educ. Technol., vol. 6, pp. 291–295, 2016.

[41] M. Ligh, S. Adair, B. Hartstein, M. Richard, “Malware analyst's cookbook and DVD: tools and techniques for fighting malicious code”. Wiley Publishing, 2010.

[42] C. T. Lin, N. J. Wang, H. Xiao, and C. Eckert, “Feature Selection and Extraction for Malware Classification”, J. Inf. Sci. Eng., 31(3), pp965-992, 2015.

[43] L. Liu, B. S. Wang, B. Yu, Q. X. Zhong, “Automatic malware classification

and new malware detection using machine learning". Frontiers of Information Technology & Electronic Engineering, 18(9), 1336-1347, 2016.

[44] H. Liu, J. Li, L. Wong, "A comparative study on feature selection and classification methods using gene expression profiles and proteomic patterns", Genome informatics, 13, pp51-60, 2002.

[45] R. W. Lo, K. N. Levitt, and R. A. Olsson. "MCF: a malicious code filter". Computers & Security, 14(6):541–566, 1995.

[46] M. M. Masud, L. Khan, B. Thuraisingham, "A hybrid model to detect malicious executables", In 2007 IEEE International Conference on Communications, pp.1443-1448, IEEE, June 2007.

[47] M. Mays, N. Drabinsky, S. Brandle, "Feature Selection for Malware Classification". In MAICS (pp. 165-170), 2017.

[48] V. V. Mieghem, "Detecting malicious behaviour using system calls", 2016.

[49] V. Moonsamy, R. Tian, L. Batten, "Feature reduction to speed up malware classification". In Nordic Conference on Secure IT Systems, pp176-188, Springer, Berlin, Heidelberg, October 2011.

[50] E. Moshiri, A. B. Abdullah, R. A. B. R. Mahmood, Z. Muda, "Malware Classification Framework for Dynamic Analysis using Information Theory". Indian Journal of Science and Technology,Vol 10 (21), June 2017.

[51] A. Ninyesiga, J. Ngubiri, "Malware Classification using API System Calls". International Journal of Technology and Management, 3(2), 9-9, 2018.

[52] M. Norouzi, A. Souri, M. Samad Zamini, "A data mining classification approach for behavioral malware detection". Journal of Computer Networks and Communications, 2016.

[53] D. Oktavianto, I. Muhardianto, "Cuckoo malware analysis". Packt Publishing Ltd, 2013.

[54] M. Palechor, F. Enrique, A. K. De La Hoz Manotas, E. De La Hoz Franco, P. P. Ariza Colpas, "Feature selection, learning metrics and dimension reduction in training and classification processes in intrusion detection systems", 2015.

[55] S. Parker, "A list of static analysis tools for Portable Executable (PE) files". Aug 31, 2017.

[56] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, J. Vanderplas, "Scikit-learn: Machine Learning in Python," Journal of

machine learning research, Vol. 12, pp2825–2830, 2011.

[57] A. Pektaş, “Behavior based malware classification using online machine learning” (Doctoral dissertation, Grenoble Alpes), 2015.

[58] R. S. Pirscoveanu, S. S. Hansen, T. M. Larsen, M. Stevanovic, J. M. Pedersen, & A. Czech, “Analysis of malware behavior: Type classification using machine learning”, In 2015 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA), pp.1-7, IEEE, June 2015.

[59] Y. Qi, “Random Forest for bioinformatics”.

[60] Y. Qiao, Y. Yang, L. Ji, J. He, “Analyzing malware by abstracting the frequent itemsets in API call sequences”. In 2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (pp. 265-270). IEEE, July, 2013.

[61] R. G. Ramani, S. S. Kumar, S. G. Jacob, “Rootkit (malicious code) prediction through data mining methods and techniques”. In 2013 IEEE International Conference on Computational Intelligence and Computing Research (pp. 1-5). IEEE, December, 2013.

[62] A. Ramaswamy, “Detecting kernel rootkits.” 2008.

[63] S. Ranveer, S. Hiray, “Comparative analysis of feature extraction methods of malware detection”. International Journal of Computer Applications, 120(5), 2015.

[64] S. Raschka, “Python machine learning”. Packt Publishing Ltd; Sep 23, 2015.

[65] W. Renkjumnong, “SVD and PCA in Image Processing”. Thesis, Georgia State University, 2007.

[66] I. A. Saeed, A. Selamat, A. M. Abuagoub, “A survey on malware and malware detection systems”. International Journal of Computer Applications, 67(16), 2013.

[67] Z. Salehi, A. Sami, M. Ghiasi, “Using feature generation from API calls for malware detection,” Comput. Fraud Secur., vol. 2014, no. 9, pp. 9–18, 2014.

[68] Z. Salehi, M. Ghiasi, A. Sami, “A miner for malware detection based on API function calls and their arguments”, In The 16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP 2012), pp563-568, IEEE, May 2012.

[69] A. Sami, B. Yadegari, H. Rahimi, N. Peiravian, S. Hashemi, A. Hamze. “Malware detection based on mining api calls”. In Proceedings of the 2010 ACM Symposium on Applied Computing, SAC ’10. ACM, 2010.

[70] I. Santos, J. Nieves, P. G. Bringas, “Semi-supervised learning for unknown malware detection”. In Proceedings of the 4th International Symposium on Distributed Computing and Artificial Intelligence (DCAI). 9th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS), 2011. in press.

[71] I. Santos, J. Devesa, F. Brezo, J. Nieves, P.G. Bringas, “Opem: A static-dynamic approach for machine-learning-based malware detection”. In International Joint Conference CISIS’12-ICEUTE 12-SOCO 12 Special Sessions (pp. 271-280). Springer, Berlin, Heidelberg, 2013.

[72] D. Sarkar, R. Bali, T. Sharma, Practical Machine Learning with Python: A Problem-Solver's Guide to Building Real-World Intelligent Systems. Apress, 2017.

[73] A. Sarode, “Machine Learning –Learning Cybersecurity”. 2018.

[74] M. G. Schultz, E. Eskin, E. Zadok, S. J. Stolfo, “Data mining methods for detection of new malicious executables,” in Proceedings of the IEEE Symposium on Security and Privacy, S&P, pp. 38–49, Oakland, Calif, USA, 2001.

[75] L. Seltzer, “Tools for analyzing static properties of suspicious files on windows”. SANS Digital Forensics and Incident Response Blog Last visited 18/11/2015, 2014.

[76] R. Sihwail, K. Omar, K. A. Z. Ariffin, “A Survey on Malware Analysis Techniques: Static, Dynamic, Hybrid and Memory Analysis”. International Journal on Advanced Science, Engineering and Information Technology, 8(4-2), 1662-1671, 2018.

[77] M. Sikorski, A. Honig, “Practical malware analysis: the hands-on guide to dissecting malicious software”. no starch press, 2012.

[78] M. Souppaya, K. Scarfone, “Guide to malware incident prevention and handling for desktops and laptops”. NIST Special Publication, 800, p.83, 2013.

[79] D. Swathigavaishnave, R. Sarala, “Detection of Malicious Code-Injection

Attack Using Two Phase Analysis Technique," Pondicherry Engineering College Puducherry, India. May 2012.

[80] G. Tahan, L. Rokach, Y. Shahar, "Automatic malware detection using common segment analysis and meta-features". Journal of Machine Learning Research, vol. 13, pp. 949–979, 2012.

[81] R. Tian, "An integrated malware detection and classification system" (No. Ph. D.). Deakin University, 2011.

[82] A. H. Toderici, M. Stamp, "Chi-squared distance and metamorphic virus detection". Journal of Computer Virology and Hacking Techniques, 9(1), 1-14, 2013.

[83] C. Wang, J. Pang, R. Zhao, X. Liu, "Using API sequence and Bayes algorithm to detect suspicious behavior". In 2009 International Conference on Communication Software and Networks (pp. 544-548). IEEE, February 2009.

[84] C. Wong, S. Bielski, J. M. McCune, C. Wang, "A study of mass-mailing worms". In Proceedings of the 2004 ACM workshop on Rapid malcode (pp. 1-10). ACM, October 2004.

[85] L. Xiaofeng, Z. Xiao, J. Fangshuo, Y. Shengwei, S. Jing, "ASSCA: API based Sequence and Statistics Features Combined Malware Detection Architecture". Procedia Computer Science, 129, 248-256, 2018.

[86] G. Yan, N. Brown, D. Kong, "Exploring discriminatory features for automated malware classification". In International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (pp. 41-61). Springer, Berlin, Heidelberg, July 2013.

[87] Y. Ye, T. Li, D. Adjeroh, S. S. Iyengar, "A Survey on Malware Detection Using Data Mining Techniques," ACM Comput. Surv., vol. 50, no. 3, pp. 1–40, 2017.

[88] Y. Ye, T. Li, Y. Chen, Q. Jing, "Automatic malware categorization using cluster ensemble". In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 95-104). ACM, July 2010.

[89] T. Ye, D. Wang, T. Li, D. Ye, Q. Jiang. "An intelligent pe-malware detection system based on association mining. Journal in computer virology", pages 323–334, 2008.

[90] A. Zacks, Malware Statistics, Trends and Facts in 2019.

[91] M. F. Zibran, “Chi-squared test of independence”. Department of Computer Science, University of Calgary, Alberta, Canada, 2007.

[92] “2019 State of Malware”. Malwarebytes Resources.

[93] “KoreLogicSecurity”, https://github.com/KoreLogicSecurity/mastiff.

[94] “Malware”. The Independent IT-Security Institute. https://www.av-test.org/en/statistics/malware/

[95] “MYTOOLZ”, https://aluigi.altervista.org/mytoolz.htm

[96] “Peframe”. https://github.com/guelfoweb/peframe

[97] “Timeline of computer viruses and worms”. https://en.wikipedia.org/wiki/Timeline_of_computer_viruses_and_worms

[98] V. Total, “Virustotal-free online virus, malware and url scanner”, Online: https://www. virustotal. com/en, 2012.

[99] VirusShare, [online] Available: http://tracker.virusshare.com:6969/

[100] What is a Trojan Virus?”. https://www.kaspersky.com/resource-center/threats/trojans

[101] “Dynamic-Link Libraries”, https://docs.microsoft.com/en-us/windows/win32/dlls/dynamic-link-libraries, 05/31/2018.

LIST OF ACRONYMS

API	Application Programming Interface
API CS	API Call-Sequences
API FBs	API Frequency Bins
APTs	Advanced Persistent Threats
BMC	Benign-Malware Classification
BMFC	Benign- Malware Family Classification
BSST	Best Subset Selection Technique
BYOD	Bring Your Own Device
C&C	Command and Control
CfS	Correlation-based feature selection
CM	Confusion Matrix
CPU	Central Processing Unit
CS	Cuckoo Sandbox
DL	Deep Learning
DLL	Dynamically Linked Libraries
EFF	Extracted Features Files
EXE	Executable
FE	Feature Extraction
FN	False Negative
FNR	False Negative Rate
FP	False Positive

FPR	False Positive Rate
FS	Feature Selection
FSST	Forward Stepwise Selection Technique
HFS	Hierarchical Feature Selection
HIDS	Host-Based Intrusion Detection Systems
IG	Information Gain
JSON	JavaScript Object Notation
LFC	Library Function Calls
NMS_RE	Naming Malicious Samples using the Regular Expression
LSTM	Long Short-Term Memory
M vs C	Malware versus Cleanware
MBFDB	Malicious/Benign Feature DataBase
MBFE	Malware-Benign Feature Extraction
MCD	Malware Classification and Detection
MCSF	Malware Classification System Framework
MFC	Malware Family Classification
MFEA	Malware Feature Extraction Algorithm
ML	Machine Learning
MMCD	Microsoft Malware Challenge Dataset
NIDS	Network Based Intrusion Detection System
NIPS	Network Based Intrusion Prevention System
NN	Neural Networks
PCAP	Packet Capture

PR	Precision-Recall
PSI	Printable String Information
RE	Regular Expressions
ROC	Receiver Operating Characteristic
SE	Social Engineering
SSL/TLS	Secure Sockets Layer/Transport Layer Security
TCP	Transmission Control Protocol
TNR	True Negative Rate
TPF	Technical PE Features
TPR	True Positive Rate
UDP	User Datagram Protocol
VT	VirusTotal
MSNE	Malicious Sample Name Extraction