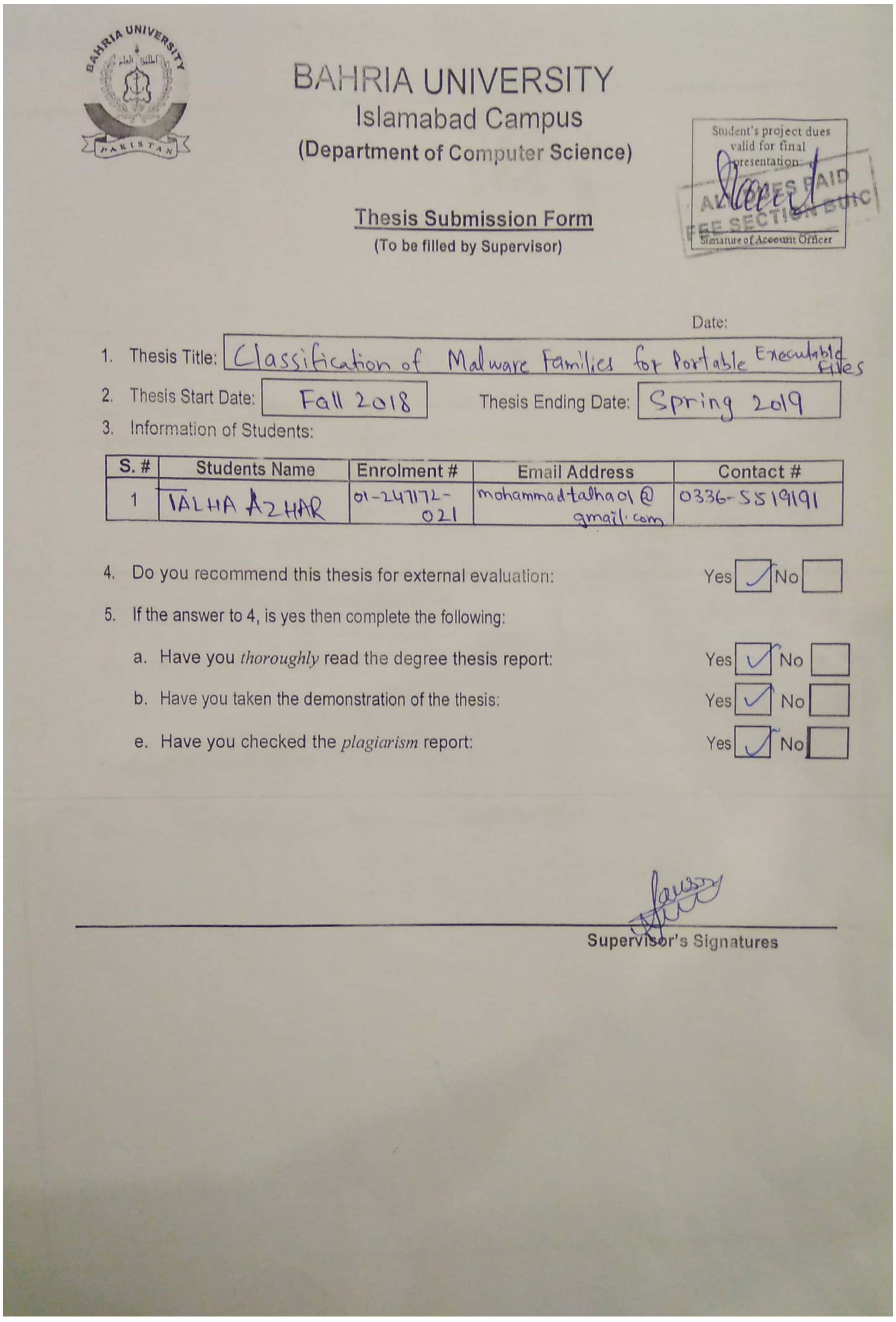
Classification of Malware Families for Portable Executable Files

Thesis Submitted By:
Talha Azhar
01-247172-021

Supervisor:
Dr.Sumaira Kausar
Co-Supervisor:
Dr.Faisal Bashir

A dissertation submitted to the Department of Computer Science, Bahria University, Islamabad as a partial fulfillment of the requirements for the award of the degree of Masters in Information Security.

Session (2017-2019)



BAHRIA UNIVERSITY
Islamabad Campus
(Department of Computer Science)

Student's project dues valid for final presentation: [illegible]

ALL DUES PAID
FEE SECTION BUIC

Signature of Account Officer

Thesis Submission Form
(To be filled by Supervisor)

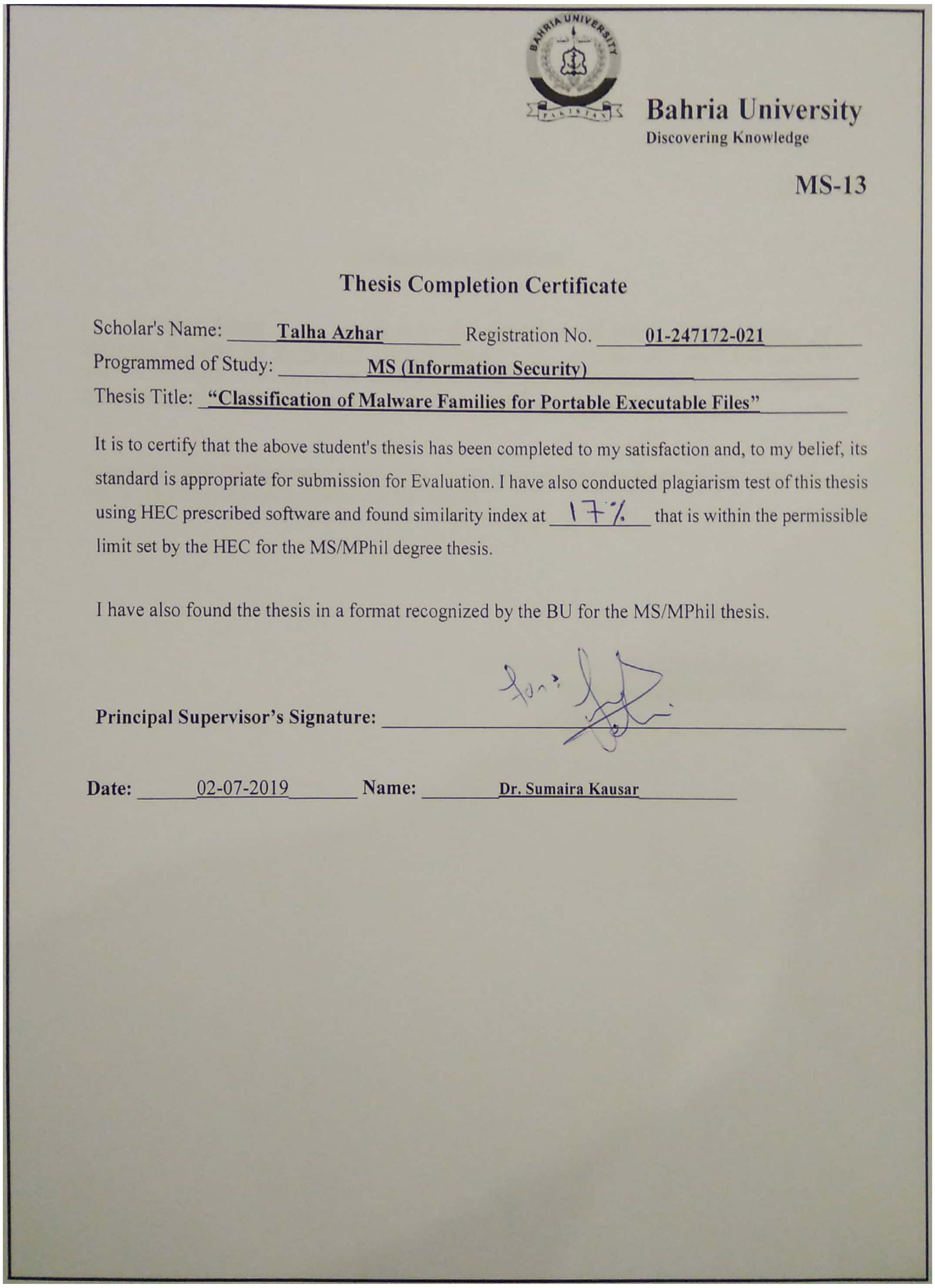
Date:

1. Thesis Title: Classification of Malware Families for Portable Executable Files
2. Thesis Start Date: Fall 2018 Thesis Ending Date: Spring 2019
3. Information of Students:

S. #	Students Name	Enrolment #	Email Address	Contact #
1	TALHA AZHAR	01-247172-021	mohammadtalha01@gmail.com	0336-5519191

4. Do you recommend this thesis for external evaluation: [✓ Yes] No
5. If the answer to 4, is yes then complete the following:
   a. Have you *thoroughly* read the degree thesis report: [✓ Yes] No
   b. Have you taken the demonstration of the thesis: [✓ Yes] No
   e. Have you checked the *plagiarism* report: [✓ Yes] No

[illegible]
Supervisor's Signatures



Bahria University
Discovering Knowledge

MS-13

Thesis Completion Certificate

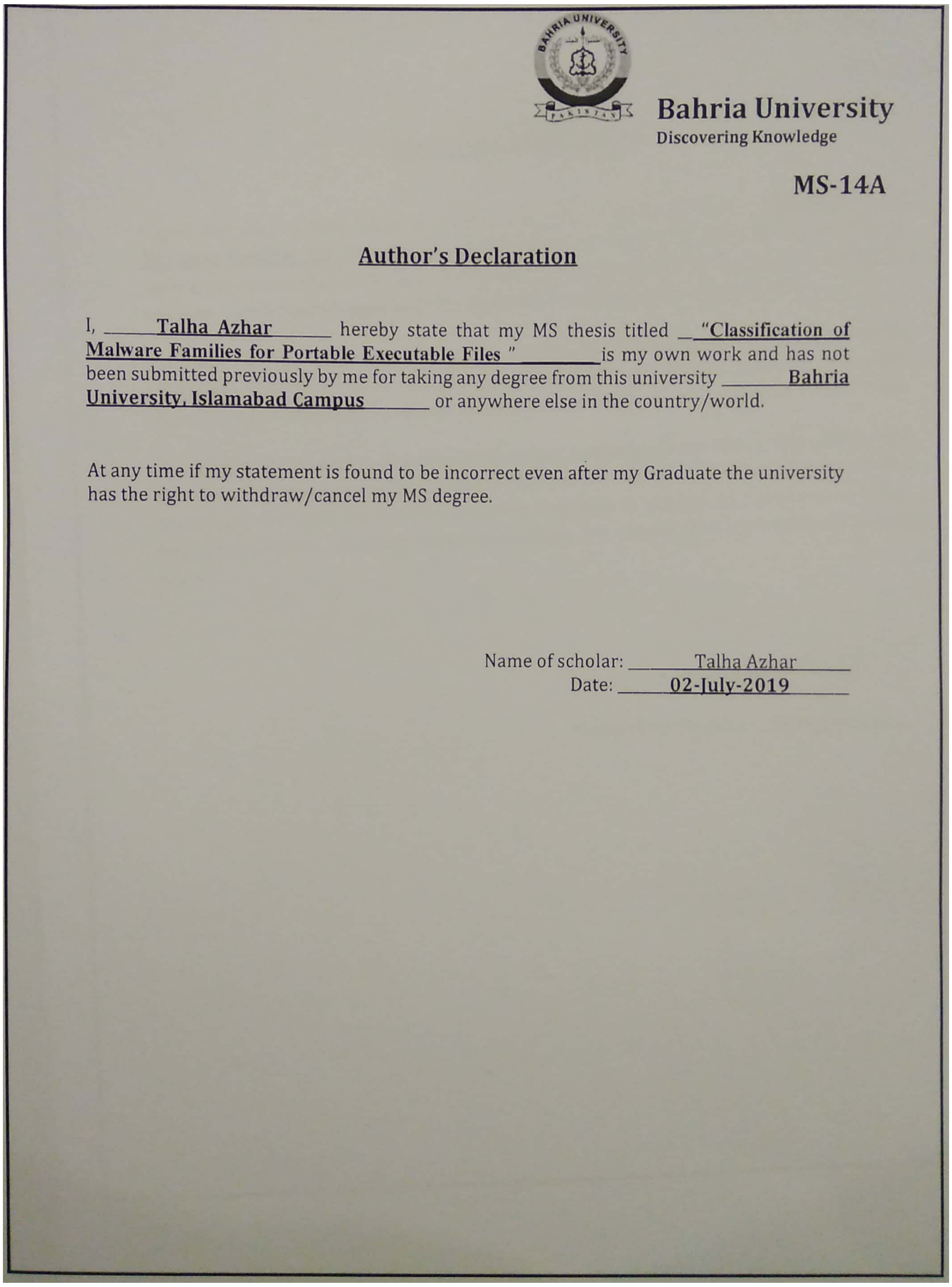
Scholar's Name: **Talha Azhar** Registration No. **01-247172-021**
Programmed of Study: **MS (Information Security)**
Thesis Title: **"Classification of Malware Families for Portable Executable Files"**

It is to certify that the above student's thesis has been completed to my satisfaction and, to my belief, its standard is appropriate for submission for Evaluation. I have also conducted plagiarism test of this thesis using HEC prescribed software and found similarity index at 17% that is within the permissible limit set by the HEC for the MS/MPhil degree thesis.

I have also found the thesis in a format recognized by the BU for the MS/MPhil thesis.

Principal Supervisor's Signature: [illegible]

Date: 02-07-2019 **Name:** **Dr. Sumaira Kausar**



Bahria University

Discovering Knowledge

MS-14A

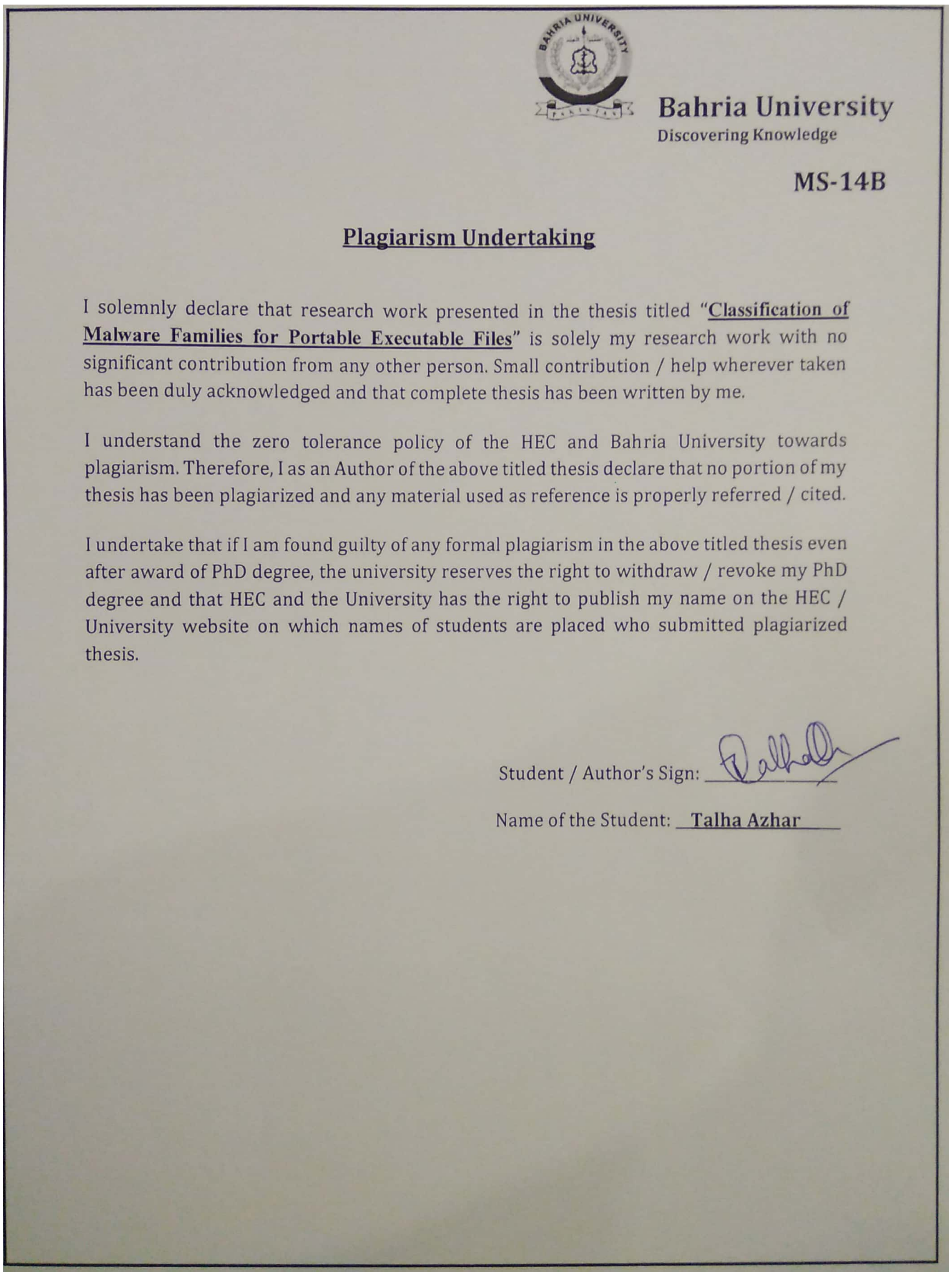
Author's Declaration

I, **Talha Azhar** hereby state that my MS thesis titled **"Classification of Malware Families for Portable Executable Files "** is my own work and has not been submitted previously by me for taking any degree from this university **Bahria University, Islamabad Campus** or anywhere else in the country/world.

At any time if my statement is found to be incorrect even after my Graduate the university has the right to withdraw/cancel my MS degree.

Name of scholar: Talha Azhar

Date: **02-July-2019**



Bahria University
Discovering Knowledge

MS-14B

Plagiarism Undertaking

I solemnly declare that research work presented in the thesis titled "**Classification of Malware Families for Portable Executable Files**" is solely my research work with no significant contribution from any other person. Small contribution / help wherever taken has been duly acknowledged and that complete thesis has been written by me.

I understand the zero tolerance policy of the HEC and Bahria University towards plagiarism. Therefore, I as an Author of the above titled thesis declare that no portion of my thesis has been plagiarized and any material used as reference is properly referred / cited.

I undertake that if I am found guilty of any formal plagiarism in the above titled thesis even after award of PhD degree, the university reserves the right to withdraw / revoke my PhD degree and that HEC and the University has the right to publish my name on the HEC / University website on which names of students are placed who submitted plagiarized thesis.

Student / Author's Sign: [illegible]

Name of the Student: **Talha Azhar**

Abstract

Cyber-attacks have been on the rise especially after the explosive widespread of social networking as it gives cyber criminals a way to break into other’ s computers and manipulate personal and sensitive data. Many different techniques have been used in the past to minimize the occurrences of cyber-attacks. These techniques focused primarily on traffic in order to look for malicious activity. This research proposes a methodology that can classify malware family on the basis of statistical features. To keep original features, we use Variance, ¾ quartile method to eliminate the un-important features. Forward selection wrapper method in feature selection is used to find out best features. To validate our proposed methodology, K Nearest Neighbor and Decision Tree is used as classifier and very promising results are achieved.

Acknowledgments

Up and above everything else, all praise to **Almighty Allah** alone, the Omnipotent, the Omnipresent, the Most Merciful and Compassionate, who blessed me with the sense of inquiry and potential for the successful accomplishment of this pieces of work.
Special praise to the final messenger, Prophet Muhammad (peace and blessings of Allah be upon him), who is the most perfect man ever lived and a torch of guidance for humanity forever.
The author expresses his profound and sincere appreciation and gratitude to the supervisor, Dr. Sumaira Kausar and Co-Supervisor,Prof. Dr. Faisal Bashir for their continuous guidance, inspiration, and patience throughout the thesis period. I am deeply beholden to Mr. Rizwan Ahmed, for his constant support and assistance. Without his supervision, constant help, and mentorship, this dissertation would not have been possible.
I find no words at my command to express my profound admiration to my affectionate parents, brothers and sisters for their moral support, encouragement and prayers throughout my academic career. Cordial indebtedness is documented for them, without their motivation and prayers; my present studies would have been a mere dream.
Moreover, the author is also indebted to NESCOM for funding this thesis under the RAC program and Computer Science department of Bahria University, Islamabad.

TALHA AZHAR
Bahria University Islamabad, Pakistan

July 2019

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

PE	Portable Executbale
SVM	Support Vector Machine
AV	Anti-Virus
ANN	Artificial Neural Network
DT	Decision Tree

Chapter 1

Introduction

1.1 Introduction

The malicious software which are being used by different types of cybercriminals nation states and hactivists are called "Malwares". These software are being used to steal professional and personal information by passing through an illegitimate access path. Moreover, these software are available in several different types which are: executable codes, scripts, ads, active content and several different types of other alternative software are also available for this purpose. Different classes of malware possess several different meanings in order to propagate the malware on the systems and as well as harming the system. Malware is a risky and a predictable danger for many users around the globe, malware has produced a large avoidance advancements.

The innovations of assailants keep on advancing the goal of security sellers in this field; due to which the existing security measures and methods are now restricting the procedures that are being used to transform the traditional arrangements of security into standard abilities. Malware also known as malicious software is a set of instructions that perform a fishy or a malicious activity on the targeted system. The piece of a code replicates itself and attached itself to the other executable programs in the system. There are many different types of malware available today; which include: viruses, worms, Trojans, root kits, backdoors, bots, spyware, adware, scareware and many more. These all types of malware behaves maliciously on the targeted systems and try to damage the software in several manners.

In order to protect the systems from these malicious attacks, there has been several different antivirus programs developed among which Norton, McAfee, Sophos, Kaspersky and clam antivirus programs are famous. These all types of antiviruses programs are made up of latest technologies which help the vendors to design the product in a manner to fight against different types of malware. Signature database is one of the primary tools being used by different developers in order to detect malware. This mechanism proved to be of the effective method against existing malware but at the same time for the newly discovered or not well known malware it may not be able to work properly. However, obfuscation, code displacement, compression and encryption are some of the techniques which can be used to dodge detection of malware based on signature. These software or techniques has made the malware writers to bypass signatures.

Therefore, the antivirus developers are nowadays trying their level best to detect the variants of several known variants so that they can develop strong antivirus software. There has been several different techniques which has been proposed and developed to data in order to design strong antivirus software which include heuristics, integrity, verification and sandboxing; however, these techniques are not effective while we talk about detection of new malware on the systems. Thus, it can be concluded that the virtual protection of the systems will remain there until and unless the signature mechanism will be extracted or deployed.

1.2 Classes of Malwares

The infection mechanism and behavior of the malware divides them into many different types including viruses, rootkits, worms, trojans, backdoors, spyware, adware and many more. Various types of malwares has been discussed below in Table 1.1; However, the table also explains infection mechanisms of these different types of malwares.

All together to recreate viruses regularly require human participation. This section portrays the most widely recognized strategies utilized by assailants to infect computer systems.

1.3 Malware Analysis

Here are two different types of technique with the help of which one can analyze the malicious program or activity. These two technique are categorized as static analysis and dynamic analysis of program, both of these are explained as below:

1.3.1 Static Analysis

The program can be statically analyzed when the malicious code has not been executed which means analyzing the program without running it on the system is known as static analysis of the program.

Class	Description	Infection Mechanisms
Virus	It recreates by joining itself to different programs running on the framework. All together to recreate viruses regularly require human participation. This section portrays the most widely recognized strategies utilized by assailants to infect computer systems.[1]	• Overwriting Technique • Appending Technique • Infecting Boot Sectors • Infecting Word Documents • Infecting Images.
Worms	It is a self-replicating computer program that spreads duplicates of itself to different computers over the system.[2]	• Buffer overflow exploits • Network file sharing exploit • Zero-day exploits
Backdoor	backdoor is a malicious program introduced by an attacker on an objective system to use it for remote . Cyber goons misuse different vulnerabilities on target machine to introduce backdoor.[3]	• Modifying startup files • Dropping a registry key
Trojans	A program that seems,by all accounts,to be normal however performs malicious functions is called Trojan.[4]	• Changing name of the file to real program's name • Changing the type of file • Modifying the source code of a genuine software utilizing polymorphic code

Table 1.1: Malware Classes, Description & Infection Mechanisms

There are many different detecting patterns being used for static analysis of the programs which are String Signature, Byte Sequence, Syntactic Library call, N-grams, Control flow graph(CFG), Operational code(op-code) frequency distribution.

These all above mentioned patterns decrypt the exe files first and then proceed with the static analysis of the program. Moreover, there are different tools which are being used for this type of analysis these tools include: IDA Pro, OllyDbg and many more. The two mentioned tools i.e. IDA Pro [5] and OllyDbg [6] helps in viewing the malware code in the form of assembly instruction of Intel x86 architecture. The assembly instruction view of the code helps in identifying the exact patterns or logs of the malware's action. On the other hand, LordPE [7] and OllyDump [8] are the tools which help to dump the memory in order to obtain the protected code located somewhere in the system. This dumping of memory helps in analyzing the exe packets in a more useful and beneficial manner.

Author in the study [9], stated different drawbacks of the static analysis and proposed a scheme which later on proved that static analysis of the system is not only the sufficient approach; one must need to implement dynamic analysis on the program in order to obfuscate the conversion of code in the vulnerable environment.

1.3.2 Dynamic Analysis

Interacting with the system and executing the program in order to analyze the behavior of the malicious program is known as dynamic analysis of the program. This type of analysis has to be done in a controlled environment which may include virtual machines, simulators, emulators, sandboxes and many other equipment like these. Following are some of the monitoring tools which monitors the process and detect changes in the system:

- Process monitor
- Capture BAT
- Process explorer
- Wireshark
- Regshot

Moreover, there are many different techniques that has been introduced and developed in order to perform dynamic analysis on the systems. These techniques monitor the process, collect the flow track, trace the instructions, monitor various function calls, analysis function parameters and do a lot more [10]. However, dynamic analysis has been proved to be the most effective technique as compared to static analysis as it discloses the natural behavior of the malware which static analysis can't do. In addition to this, it is to mention here that different automated tools has also been developed in order to automate the process of dynamic analysis of programs; these tools are as follows:

- Norman Sandbox
- CWSandbox
- Anubis
- TTAnalyzer
- Ether
- ThreatExpert

All these tools generate a report of analysis which helps in having the deep understanding the nature and behavior of the malware; as studying the behavior of malware can help in mitigating it in a more appropriate manner. Furthermore, several different AI techniques are also being used in analyzing the programs for malicious acts; among these techniques machine learning based techniques are the common one. Therefore, several different machine learning based analyzing techniques has been studied and reviewed for this research work.

1.4 Malware Detection Techniques

Malware detecting techniques identify and detect the malware on the systems and help the user to take proper countermeasures against these malware in order to keep the system and data secure in case of any malicious activity [11]. There are three broad categories of malware detection techniques which are as follows:

- Signature based detection
- Heuristic based detection

1.4.1 Signature Based detection

In Signature Based Detection, a signature is embedded in the malware in order to identify the family of malware from where it belongs. Nowadays, most of the antivirus programs are using signature based techniques [12]. String or pattern scanning is another name of signature based detection technique for malware. Moreover, this technique can be dynamic, static or can also be a hybrid detection of malware.

1.4.2 Heuristic based detection

As the name indicates that a heuristic approach detects the malware by analyzing both normal and abnormal behavior of the systems. In this type of detection the known and unknown both malware attacks can be detected; which can help the user to perform counter measures in order to mitigate malware from the system. Following are the two steps involved in heuristic based detection:

- Observing normal & abnormal behavior of the system both
- Watch the difference of normal and abnormal behavior to detect the family of malware

It is an efficient method of detecting malware but at the same time it is limited to several constraints and resources which results in high false positive rate. Proactive technique is another name for heuristic based detection.

Table 1.2 illustrated the advantages and disadvantages of malware detection techniques:

	Advantages	Disadvantages
Signature based	-Known malwares can be detected easily -Used less resources as compared to other techniques	-Unknown malwares cannot be detected.
Heuristic based	-Known and unknown new malware can be detected	-Data need to be updated regarding new and unknown malwares. -Need more resources in terms of time and space -level of false positive is high.

Table 1.2: Comparison of Detection Techniques

1.5 Motivation and Problem Description

There has been much work done on the classification of Benign and Malware Files but there is nothing to be seen for the classification of further malware families.

1.6 Research Contribution

This research work primarily focuses on classification of malware families. There are certain limitations and a defined scope for this research work. The scope defines that this research work only focuses on three classes of malware i.e. worms, viruses and Trojans. Table 1.3 below highlighted key features of this research work:

• The toughest part of this research work was to collect data as the data at most of the malware archives has special terms and conditions which made it difficult for authors to collect the required data.
• The dataset has been created from the scratch level as the data sets used earlier for this type of research work were discontinued. Moreover, the parser was used in order to parse the PE file headers and to store information.
• 75 different AVs labels from Virustital has been collected, it has been also a tough task because there was a certain limit of 4 requests per minute on Virustotal which required a lot of time.
• Classifying different malwares based on their features in different families. As early work was performed to distinguish between malware and benign files, we have just addressed the malware files and categorized them in respective families.

Table 1.3: Contribution for the Proposed Research Work

1.7 Thesis Organization

The rest of the thesis is structured as follows:

- In Chapter 2, the past techniques and other related work that is relevant to our area and scope of research is reviewed.
- In Chapter 3, the proposed methodology is explained briefly. The implementation details along with the system specifications and design requirements are outlined.
- In Chapter 4, the results of the testing phase and the subsequent analysis of the overall approach is presented.
- In Chapter 5, an overview of the research work along with the discussion regarding the limitations is done and the discussion regarding future research directions is carried out.

Chapter 2

Literature Review

2.1 Techniques used for Malware Detection

Malwares are overall scourge and malware detection techniques fill in as first line of barrier against them. The viability of a tool used for malware detection depends on the strategies/ techniques it uses. Malware detection techniques can be divided into three classes shown in Figure 2.1.

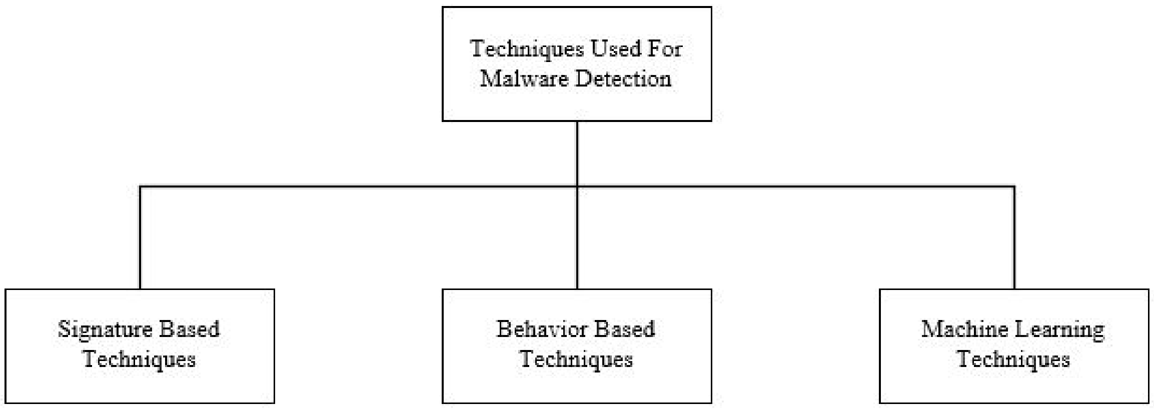

Techniques Used For Malware Detection
Signature Based Techniques
Behavior Based Techniques
Machine Learning Techniques

Figure 2.1: Techniques for Malware Detection

2.1.1 Signature Based Techniques

This techniques is generally used to identify known malware. In spite of the fact that it is exceptionally powerful, yet turned out to be ineffective if there is even a little change in the code, which thus changes its signature. Besides, this technique requires regular update of the signature database in order to detect the new malware.

2.1.2 Behavior Based Techniques

The behavior based technique continuously check the behavior of the program to decide if it is unsafe or not. In this way, this technique is used to recognize the obscure malware. Besides this, the adequacy of this technique isn't yet demonstrated. According to the experiments that are conducted on this technique shows that it consist of high false positive ratio. Also this technique take additional time in the process of detection. Heuristic Techniques utilize most part of data mining and machine learning methods to recognize the running program behavior. The significant techniques that have utilized so far incorporate (NB) naive bayes, (NN) neural network and (HM) hidden markov model.

2.1.3 Machine Learning Techniques

Clustering mainly depends on classification methods which is used to categorize the observations and the data elements. It is one of the well-known data analysis and data mining technique in which the main goal is to learn something from the given data. By utilizing different algorithms clustering analysis can be done. These algorithms vary in the techniques which determine that what basically belongs to the particular cluster and how efficiently it find the related cluster. (i.e few algorithms are used in order to check the performance of clustering which is based on calculating the distance between the cluster members, where some uses Statistical distribution of cluster for determining the clusters). Grouping is the main part in cluster analysis, the classification measurements is based on: 1. Goodness-of-fit to a postulated model. 2. Through analysis clustering (natural grouping) is revealed. As expected, the elements which belongs to a valid cluster are more likely to relate with each other, then the elements belongs to different clusters. Many different techniques are used to perform clustering in different ways.[13]

- **Intrinsic Approach** It is an unsupervised learning. Raw data is used in this approach, the data in which the labels of the classes are missing and we do not have information regarding the data.

- **Extrinsic approach** It is a supervised learning approach. In this approach raw data is used, the data in which every class has a label. Clustering can be of two types, divisive or agglomerate. An agglomerate approach is the approach, in which at the start we have many clusters and then we converge it to the less large clusters. So, it can be said that it is a bottom up approach. A divisive approach is basically the inverse of agglomerate approach. This approach starts with the less no of large clusters and it then divide them to the more clusters. So, it can be used said that it is a top down approach.

2.2 Selecting Features to Classify Malware

In existing literature, many researchers tried to answer this question with their approaches , experiments and solutions that which features should be select, how to distinguish useful and useless features.

The features are divided into three main categories:

1. Relevant
2. Irrelevant
3. Redundant

Relevant features are the key features, which have influence on overall decision where as irrelevant features are those features which are either leading to wrong decision or they don't have any good/bad influence on dataset. The values which are static throughout the dataset are usually irrelevant features. If one feature's behavior is very similar to any other feature, then this is called redundant features. Redundant features can lead to negative decision. Therefore this is essential to pick most important features which have higher impact and are unbiased for final decision. Feature selection is important phase of this research as well. The some key benefits of using feature selection algorithm are they make good categorization, lead to right decision, lower the complexity of system.[14]

2.3 Techniques for Classifying Malware

There are various well known machine learning techniques such as SVM, decision tree, Naïve Bayes and various other clustering techniques for malware detection and classification. In this section we are going to review the existing approaches and experiments for malware classification.

2.3.1 Artificial Neural Network (ANN)

This technique is used to predict the correct class. It learns the behavior of learns to predict the behavior attributes of users and daemons in the system. ANN is quite well known, reputable and expandable classifier. It consist of interconnected nodes and an output layer. ANN have potential to learn lot of features whether they are in form of text or image, It outstand the previous techniques in many cases. It has ability to learn from features and make predictions. It automatically adjust the weights and coefficients though optimizers play key role. So that ANN can work fine in both unsupervised and supervised learning problems [15].

2.3.2 Decision Tree

DTs are powerful straightforward, powerful and understandable classifier. It is graph like tree which has sequence of information. It has nodes which are labeled with feature value and leaf has class label. It works fine in supervised and reinforcement learning.[15]

2.3.3 Support Vector Machine (SVM)

SVM is a great supervised classifier, which is commonly used in machine learning problems, It also works great with CNNs. There are two main reasons of using SVM. 1) it handles very high dimensional space. 2) It works with small dataset. Like other classifier, it also has two phases training and testing, in training phase it finds the optimal hyperplane. In testing phase, it evaluates the performance of classifier. SVM has shown quite good results in other fields such as Digital Image Processing (DIP), Pattern Recognition (PR), Natural Language Processing (NLP) and etc. SVM have recently been applied to identify and counter network DoS attacks showing very high accuracy [16].

2.4 Static Malware Techniques

In [17], authors used data mining approaches for malware determination using three static features. It used bytes sequence extracted from PE files. The dataset has of 4266 files. Which has 3265 malicious and 1001 benign programs. Another approach used in [18], was applied to recognize patterns in DLL data, the authors [18], used Naïve Bayes learning algorithm for bytes sequences recognition. It gave the great accuracy 97%. According to researchers the data mining technique works better than signature based technique. In [19], authors used n-gram and data mining approach to detect malicious executables. Their experiment was couple of classifiers such as SVM, decision trees, and their advanced versions. In [20], authors used image processing techniques to visualize and classify malwares. They used gray scale image to visualize malware binaries and K-nearest neighbor for malware classification but later on researchers find some limitations, hackers can easily beat this system. In [21], authors presented the comparison between binary texture based techniques and dynamic approaches. They concluded that static methods got higher accuracy than dynamic methods. But it also limitation that one who has knowledge of static features can manipulate their code to tackle texture analysis.

In [22], authors presented an automated malware classifier which is based on structural information, the clustering is done by distance matrix. In [23], authors used number of bytes for determination of Trojans. They observed that adding few more features can improve classification accuracy.

In [24], authors used algorithms of WEKA repository for determination of virus. In [25], Santos et al. concluded that supervised learning require good , unbiased and large dataset which has both malicious and benign classes , they also stated that semi supervised learning techniques can be used for unknown malware determination such as LLGC (Learning with Local and Global Consistency). LLGC is a semi supervised approach which consider both labeled and unlabeled data smoothly. It is found in literature [19], and [26], that supervised learning techniques outperformed the semi supervised approach. Supervised learning approaches for malware detection and determination led to above 90% accuracy. Moreover, in [27], authors made a great contribution and formalized a collective learning framework for unknown malware detection. They also presented optimization approach for unlabeled data. Collective learning frameworks use labeled and unlabeled instances. This approach is helpful when we have small size of labeled dataset. In [28], authors used non-uniform sequence of instruction length. They applied decision tree and random forest for classification.

2.5 Dynamic Malware Techniques

In [29], authors proposed a technique for malware behavior analysis. For the collection phase authors used Amun and HoneyClients tool. They analyzed the behavior on CWSand box [30], and Anubis [31], on VM platform. For now it is almost impossible to have manual analysis. In [32], authors proposed an automated mechanism for malware behavior evaluation and then they used clustering. In [33], authors used Ether framework [34]. In [33], malware detection, 2-gram and markov chain algorithm were applied. Markov chain is similar to graph, this approach was found computationally expensive. In [35], automatic execution traces based on Anubis was applied. And later, LH algorithm is used for clustering. In [35], authors formalized a better approach. In it they used sub linear approach to knn problem. The authors demonstrated that it is quite scale-able and computationally less expensive. In [36], authors used framework for API calls extraction and their pattern on virtual machine. They used 1368 malwares and 456 cleanwares in their work and used classification algorithms given in WEKA library. They claimed 97% accuracy. In [37], authors did research on similar features presented in antivirus and virus files. And then they presented a classification algorithm which stores malware behavior and system state changes. For this research they used half firewalled virtual environment. They analyzed performance of their system over 3700 malwares gathered in six months. In [38], authors proposed malware classification algorithm which is based on maximal component subgraph detection. They used sandboxed environment for dynamic features extraction. It is seen that this approach has some major drawbacks such as some malwares do not make system calls and ruin the analysis method.They [39], did similar investigation over various classifiers for example kNN, Naive Bayes, J48 Decision Tree, SVM and Multilayer Perceptron Neural Network (MLP) is done on a little data set of 220 malicious samples and 250 benign samples with and without feature selection. The got outcomes delineated that general best exhibition is accomplished by J48 decision tree with a recall of 95.9%, a false positive rate of 2.4%, a precision of 97.3%, and an accuracy of 96.8%.

2.6 Hybrid Malware Techniques

It is observed that sometimes static or dynamic is not sufficient for classification, and we then need to use hybrid approach. Hybrid techniques use combination of static and dynamic features simultaneously for higher accuracy. It is proposed in [40], that a malware detector (OPEM) use both static and dynamic analysis. And then used different classifiers like Decision tree, K-nearest neighbor, Bayesian network and support vector machine. It is found that hybrid approach improved the overall performance.

In [41], authors have also adopted hybrid approach for classification, they used length frequency, string information, API function names and API parameters. This experiment was done using 2939 exe files. They retrieved results using SVM, IB1, DT and RF. They also tested these approaches on static and dynamic features. And found hybrid techniques are better and stated that RF gave the best performance among all other classifiers. In [42], authors used multiple resources static binary, disassembled binary file, flow graph, dynamic instruction trace and system call. They also used file information feature vector. They used SVM for classification and used more than 700 malware and benign classes and got 98% accuracy. Data mining and ML techniques have played vital role in malware classification.

Chapter 3

Methodology

3.1 Introduction

Malwares are found attached in different files like Word, HTML document, APK files, EXE files and many more.We are interested only in the PE files.So for analyzing the PE file, first we have to understand about the structure of PE file. Fig 3.1 tells the overview of PE file structure.

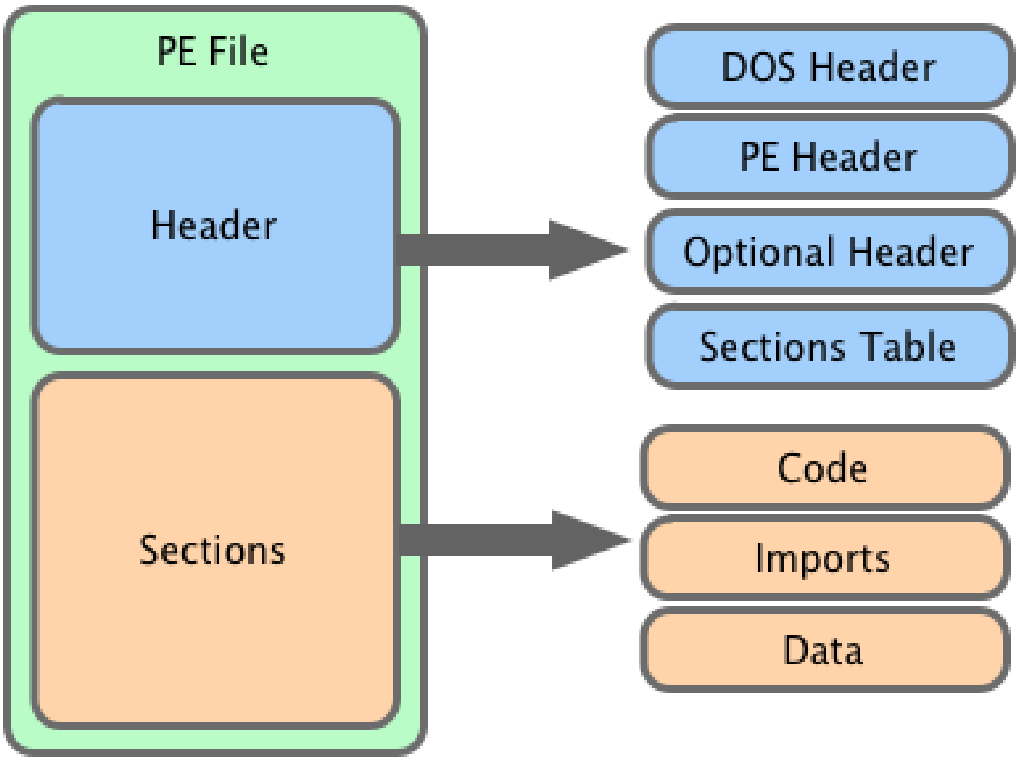


Figure 3.1: PE File Structure

3.1.1 PE File Format

The overview of PE format is showing as the following figure 3.1.

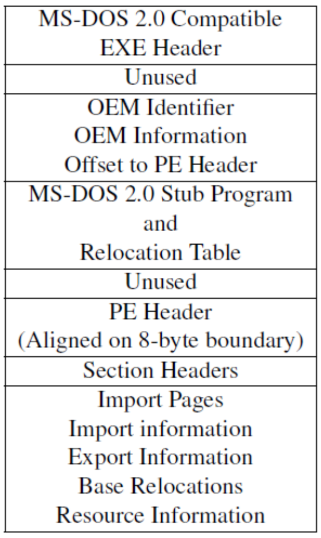


Figure 3.2: PE File Structure

The portable executable file header consists of the following parts:

- Portable Executable signature
- Microsoft-DOS stub
- Optional header

The PE file contains items in a specific sequence. The first item in the file is always the MS-DOS stub followed by a four-byte PE signature at offset 0x3c whose purpose is the identification of file as a Portable executable image file. There are other items in the list as well like COFF file header and optional file header whose function is to provide required data to the loader. The optional header is called optional because it may or may not be present in files, but in case of image files, this header is mandatory. The optional header is further subdivided into three parts which are:

1. Data Directory
2. Specific field
3. Standard field

The last item in the PE executable file is the section table. By default, every tuple of the section file is considered as the section header.

Based on the knowledge acquired from the literature and previous work, features from optional header and section header are extracted and utilized in the research.

3.2 Proposed Methodlogy

The proposed project carries out a multi-step approach. Figure 3.3 shows the methodology of the entire system.

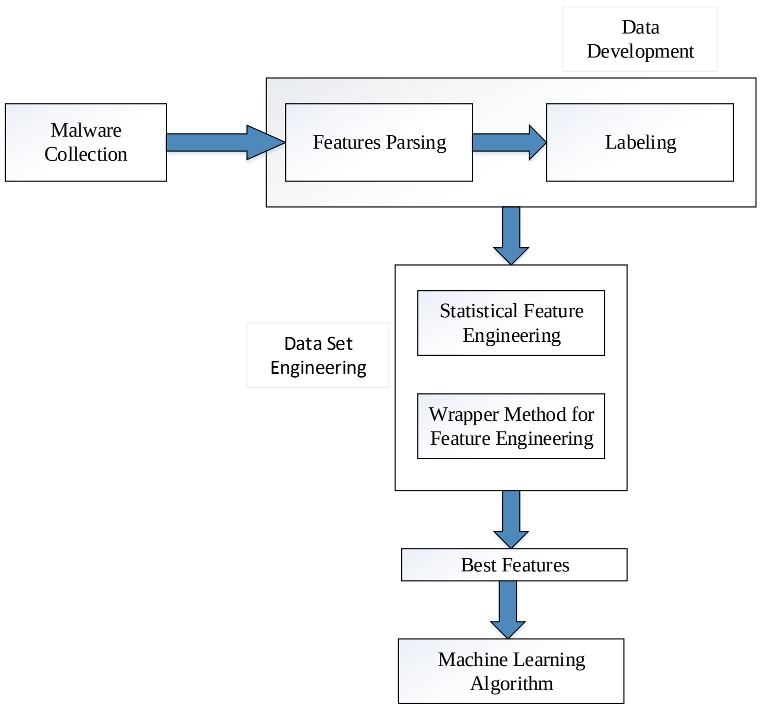

Data Development
Malware Collection
Features Parsing
Labeling
Data Set Engineering
Statistical Feature Engineering
Wrapper Method for Feature Engineering
Best Features
Machine Learning Algorithm

Figure 3.3: Methodology

1. Malware Collection
2. Features Parsing
3. Labeling
4. Statistical Feature Engineering
5. Wrapper Method for Feature Engineering
6. Best Features

3.2.1 Malware Collection

To create a solution for any problem, data is always required. In the proposed scheme, data collection is the first and most important step. Any kind of simple data is publicaly available but the data for Malwares like Viruses, Trojans etc is not easily available. Benign files can easily be found in the computer but the PE Malware files are difficult to find. Recent malware samples are not available over the internet easily as their occurrence is very low. Some security sector corporations keep brief malware repository, however they donot share their database with any other for the purpose of research.

[43] used the dataset from Malfease dataset (http://malfease.oarci.org). They used three antivirus softwares named as ClamAV, F-Prot and AVG to verify either these files are malicious or benign.

Malicia Project used to provide the dataset for the malware. Many researchers have used the Malicia-Project dataset but now it is discontinued.

Virusshare has been an important repository for the malware collection. They provide all kinds of malwares including HTML links, Executables, Word files and many more.

We have used different repositories for the malware collection including Nothink-malware repository,VirusTotal,VirusShare,VXHeaven. Table 3.1 shows the statistics of all the files we collected from different sources.

Source	No. of Files
VirusShare	1395
VirusTotal	1205
VXHeaven	950
Nothink	2093
	Total = 5643

Table 3.1: No of Files Collected from Sources

3.2.2 Features Parsing

Built-in data set already contains many features including headers of executable files. As we didn't have any malware dataset containing categories for Trojans,Viruses and Worms. A parser was made to parse the features from PE files. Code was written in python language and parsed different header values and write it to CSV file. PE file contains different headers i.e, DOS Header, Optional Header, File Header. Every header contains different features and every feature has its own value. Firstly, the parser imports pefile and parse all the files from the directory.Then the parser extracts features from DOS header,file header and optional header.

1. *DOS Header:* The DOS header contains 19 features: e_magic, e_cblp,e_lfanew etc. There are many features which contains same values for all of the files.
2. *File Header:* This header consists of the following features: Machine , NumberOfSections, TimeDateStamp, PointerToSymbolTable, NumberOfSymbols, SizeOfOptionalHeader, Characteristics

   Unfortunately, there is no such variant between malicious and legitimate files in each features.
3. *Optional Header:* Standard fields,windows specific fields and data directories are included in the Optional header.Windows specific fields have 21 features whereas standard field have 8 features.

During parsing of these files, headers of some files were missing.That files were not able to be parsed.Table 3.2 shows the errors we found during the parsing of PE files.

Total Number of Files	5643
'Invalid e_lfanew value, probably not a PE file'	15
'Invalid NT Headers signature.'	38
'Invalid NT Headers signature. Probably a NE file'	95
DOS Header Magic Not Found	250
Files Parsed with Header	5246

Table 3.2: Errors found during Parsing

3.2.3 Labeling

As there are more than 5000 files and they are still un-categorized.There is no information about these files that from which family do they belong.Some of them might be virus,some might be trojans and so on.So we took labels of these files from Virustotal.[44] Virustotal is an engine which provides us multiple functionalities. Virustotal provides the labels of 75 different Anti-Viruses(AVs) including AVG, Avast, Kaspersky, ESET NOD 32 and many others. Any file or file Hash (MD5 or SHA) can be uploaded on virustotal and it will provide all the details about that particular file including the labels,metadata of file,imports,exports and many others.

Virus Total is a free service that allows you to analyze files or URL addresses online. Many antivirus application engines and website scanners are used for analysis. Files considered to be harmful are analyzed individually in antivirus application engines. Each antivirus application engine creates an analysis report for the suspicious file [45]. The same analysis case is valid for URLs to be analyzed. The VirusTotal service includes a very large set of analyzes. In this way, a new scan can be performed, as well as previous analysis information can be obtained. Virus Total offers a service interface (VirusTotal Public API v2.0) to provide results without using a browser, as well as through a web browser. With this interface, files / URL addresses can be analyzed automatically. Virus Total Public API provides the results of the analysis as a JSON object. The results of each antivirus application engine and web browser analysis are obtained separately.

Uploading the file one by one on virustotal and copying labels of 75 different AVs and writing it to CSV file takes a lot of time.Virustotal provide APIs for different purposes like scanning,fetching reports,downloading and others.VT provides two versions of API.

- Public API
- Private API

For automation of that manual task,we computed the MD5 hash of all these files and write them in CSV file.Then we wrote a python script which uploaded the file hash from CSV file to virustotal,scanned the file hash and copied the labels of all AVs to the CSV file again.It took a lot of time to get and write the JSON reports of these files.

After getting complete labels for all the files, we named a file based on Majority Voting.This give us the following results.

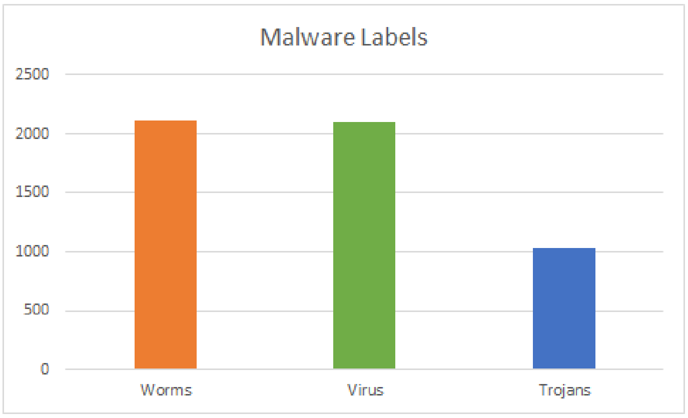


Figure 3.4: Graph After Labeling

3.2.4 Statistical Feature Engineering

Initially the dataset consists of 55 Features.Table 3.3 shows the features extracted from the file headers.

e_magic	e_cs	PointerToSymbol Table	BaseOfCode	SizeOfImage
e_cblp	e_lfarlc	NumberOfSymbols	BaseOfData	SizeOfHeaders
e_cp	e_ovno	SizeOfOptional Header	ImageBase	CheckSum
e_crlc	e_res	Characteristics	SectionAlignment	Subsystem
e_cparhdr	e_oemid	Magic	FileAlignment	DllCharacteristics
e_minalloc	e_oeminfo	MajorLinker Version	MajorOperating SystemVersion	SizeOfStack Reserve
e_maxalloc	e_res2	MinorLinker Version	MinorOperating SystemVersion	SizeOfStack Commit
e_ss	e_lfanew	SizeOfCode	MajorImage Version	SizeOfHeap Reserve
e_sp	Machine	SizeOfInitialized Data	MinorImage Version	SizeOfHeap Commit
e_csum	NumberOf Sections	SizeOfUninitialized Data	MajorSubsystem Version	LoaderFlags
e_ip	CreationYear	AddressOfEntry Point	MinorSubsystem Version	NumberOf RvaAndSizes

Table 3.3: Features of PE File

Now we have to eliminate the unimportant features from our dataset. There are many different methods used for data pre-processing but we have used the following two methods.

- **Variance Method**

  Some values are not changed throughout the data. These values are fixed for those features. The absence or presence of these values have no impact on the dataset because they are same overall. These features can be removed directly. In our data set, there were 4 features whose values remain same throughout the dataset.

  - e_magic
  - e_res
  - e_res2
  - Magic

- **3/4 Quartile Method** This is the second method which is used to remove the unnecessary features from the remaining dataset. If the occurrence of one value for one feature, exceeds more than 75% that feature should be dropped as it is an outlier and it will bias the single feature. In our dataset, there were many features which have same value occurring more than 75%. We eliminated about 30 features on this criteria. The features which were removed from this method are listed in the table 3.4.

e_crlc	e_cparhdr
e_maxalloc	e_css
e_sp	e_csum
e_ip	e_cs
e_lfarlc	e_oemid
e_oeminfo	Machine
PointerToSymbolTable	NoOfSymbols
SizeOfOptionalHeader	SectionAlignment
FileAlignment	MinorOperatingSystemVersion
MinorImageVersion	MajorSubsystemVersion
MinorSubsystemVersion	Checksum
SubSystem	DllCharacteristics
SizeOfStackReverse	SizeOfHeapReverse
SizeOfHeapCommit	LoaderFlag
NumberofRVAandSizes	CreationYear

Table 3.4: Deleted Features

After applying the two methods of data cleaning, we are left only with 21 features which are listed in table 3.5.

e_cblp	e_cp	e_minalloc	e_ovno
e_lfanew	NumberOfSections	Characteristics	MajorLinkerVersion
MinorLinkerVersion	SizeOfCode	SizeOfInitialized Data	SizeOfUninitialized Data
AddressOfEntryPoint	BaseOfCode	BaseOfData	ImageBase
MajorOperatingSystem Version	MajorImage Version	SizeOfImage	SizeOfHeaders
SizeOfStackCommit			

Table 3.5: Selected 21 Features

3.2.5 Wrapper Method for Feature Engineering

Feature selection [46, 47, 48] is considered an important method in classification of any problem. There are two main reasons for this:

- Computational complexity is reduced.
- It improves the generalization capability of a classifier.

This reason is obvious because if there is any data with large vector space or many features,it will require heavy resources and a very high cost of computation. On the other side a low-dimensional representation reduces the risk of overfitting . Features selection methods help us to reduce the dimensionality of the data and gives us a very helpful and a good subset from the available original features which helps us in building a powerful classifier to solve the problems[47].

Applying wrapper method will give us the best combination of features with maximum accuracy on the given dataset. Those features will be further used for the machine learning algorithms.

3.2.6 Machine Learning Algorithms

There are basically two different approaches.

1. **Supervised Learning**: which has label assigned to it.
2. **Unsupervised Learning**: which doesn't have any label with the data.

Both of the methods have their own algorithms. As our data is labeled, so we will be looking only to the supervised learning techniques. In this proposed solution, we will be implementing two different classifiers on our dataset and performing different experiments. Following are the two algorithms which will be used further.

- Decision Tree
- KNearest Neighbor (KNN)

Chapter 4

Experiments and Results

4.1 Experimental Setup

We carried out our experiments on system with following specifications.

- Intel Core i5 8th Generation
- 12 GB of RAM
- No Firewall
- 500GB of Harddrive

We will be using PyCharm and Anaconda framework.For coding, we will carry out our experiments in Jupyter Notebook Python 3.6 including many different libraries like Scikit-learn, pandas, Numpy and others.

On the system,we had to turn off the Firewall and disable the antivirus as we had to play with the viruses. If the firewall remained turned on,the system detect the file as virus and remove the file. As we had limited data,so we had to save maximum number of files for parsing.

After parsing and data cleaning,we were left with 21 features. Now we had to perform different experiments and start getting the results.

We applied two algorithms i.e Decision Tree and K-Nearest Neighbor(KNN).Varying the parameters, we obtained different different results and plotted them on a graph.

4.2 Results and Analysis

During the implementation phase, one of the most crucial steps is the choice of machine learning technique for each family. The choice is made based on the performance measures of the algorithms. The two performance measures that are being incorporated in the decision making process is Accuracy and Confusion Matrix, and explained as follow:

4.2.1 Accuracy

Accuracy is the ratio of number of correctly classified instances and total examined instances in the dataset. To select an algorithm for each attack category, each of the three algorithms’ accuracy is compared against the dataset. The measure of accuracy is further broken down into three different types: training accuracy, test accuracy and cross validation accuracy.

- ***Training Accuracy:*** Training accuracy is the measure of how well an algorithm is classifying when being applied to the data instances that were a part of the training dataset. In this research, the training accuracy is taken as an average of multiple runs of the algorithm on the training data in order to get a more reliable value for the training accuracy.

- ***Testing Accuracy:*** Test accuracy is the measure of accuracy in case of the algorithm’s application to the data instances that are previously unseen by the algorithm. In other words, it is the extent to which the algorithm correctly classifies the new data instances. Similar to the training accuracy, an average value of accuracy is taken based on multiple measurements.

4.2.2 Confusion Matrix

In addition to the use of accuracy as a performance measure, there is another simple yet powerful measure to describe the performance of a classification algorithm known as Confusion Matrix. It is an extension of accuracy measure in the sense that it also keeps track of how many data instances are classified incorrectly in addition to the correct ones. A confusion matrix has two rows and two columns of the form:

Predicted Values			
		Predicted : 0	**Predicted : 1**
Actual Values	**Actual : 0**	True Positive	False Negative
	Actual : 1	False Positive	True Negative

Table 4.1: Confusion Matrix

In the context of our attack prediction scenario, the four possible values based on the classification statistics of the algorithm after it is run on the test data instances are:

- **True Positive:**The data instance actually belongs to one class and the algorithm correctly classifies it as the same class.
- **False Negative:**The algorithm predicted the data instance is not from 1st class whereas it actually belongs to 1st class.
- **False Positive:**The data instance doesnot actually belong to 1st class but the algorithm incorrectly classified it in 1st class.
- **True Negative:**The algorithm predicted the data instance doesnot belong to 1st class and it actually belongs to the other class.

First of all we performed the experiments with the complete dataset including 21 features. Now starting with the experiments,there are many different experiments and their results are discussed at the end.70% of the dataset was used as training and the testing was performed on remaining 30

4.3 Experiments

We divided our experiments in two main categories.

1. Experiments with 21 Features
2. Experiments with 9 Features

4.3.1 Experiments with 21 Features

We created 4 different CSV files according to following statistics.

1. File containing instances of only Virus and Worms.
2. File containing instances of only Trojans and Worms.
3. File containing instances of only Virus and Trojans.
4. File containing instances of Trojans, Virus and Worms.

4.3.1.1 File containing instances of only Virus and Worms

In this file there were a total of 4218 instances consisting of 2101 viruses and 2116 worms. We applied decision tree algorithm on 70% of the training data. A total of 2951 different instances were trained including 1461 virus instances and 1490 instances of worms. As the classifier was trained, we performed our test on remaining 30% of the data.

The model predicted 605 instances out of 640 correctly as Virus and 591 instances out 626 correctly as Worms giving us the accuracy of 94.4%.

Then we applied the KNN classifier on the same number of instances and observed the results. The KNN classifier after testing the data gave the accuracy of 94.5%.

The confusion matrix of this experiment is given below in Table 4.2.

Models		**Predicted Labels**			
		N(1266)	Virus	Worms	Accuracy
Decision Tree	**Actual Labels**	Virus	605	35	94.4%
		Worms	35	591	
K-Nearest Neighbor		Virus	615	25	94.6%
		Worms	44	582	

Table 4.2: Confusion Matrix of Virus and Worms

The comparative results of the above table are shown graphically in Figure4.1.

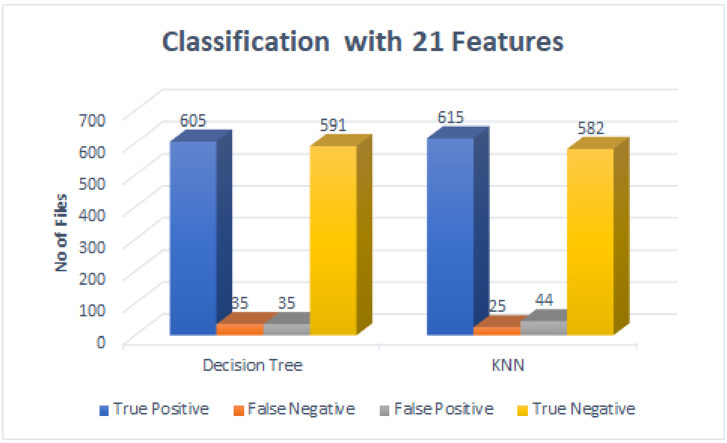


Figure 4.1: Confusion Matrix Statistics for Virus and Worms

4.3.1.2 File containing instances of only Virus and Trojans

In this file there were a total of 3130 instances consisting of 2101 viruses and 1028 trojans. We applied decision tree algorithm on 70% of the training data. A total of 2197 different instances were trained including 1487 virus instances and 703 instances of trojans. As the classifier was trained, we performed our test on remaining 30% of the data.

The model predicted 527 instances out of 614 correctly as Virus and 211 instances out 325 correctly as trojans giving us the accuracy of 78.6%.

Then we applied the KNN classifier on the same number of instances and observed the results. The KNN classifier after testing the data gave the accuracy of 76.01%.

The confusion matrix of this experiment is given below in Table 4.3.

Models		**Predicted Labels**			
		N(939)	Trojans	Virus	Accuracy
Decision Tree	**Actual Labels**	Trojan	211	114	78.6%%
		Virus	87	527	
K-Nearest Neighbor		Trojan	192	133	76.01%
		Virus	93	521	

Table 4.3: Confusion Matrix of Virus and Trojans

The comparative results of the above table are shown graphically in Figure4.2.

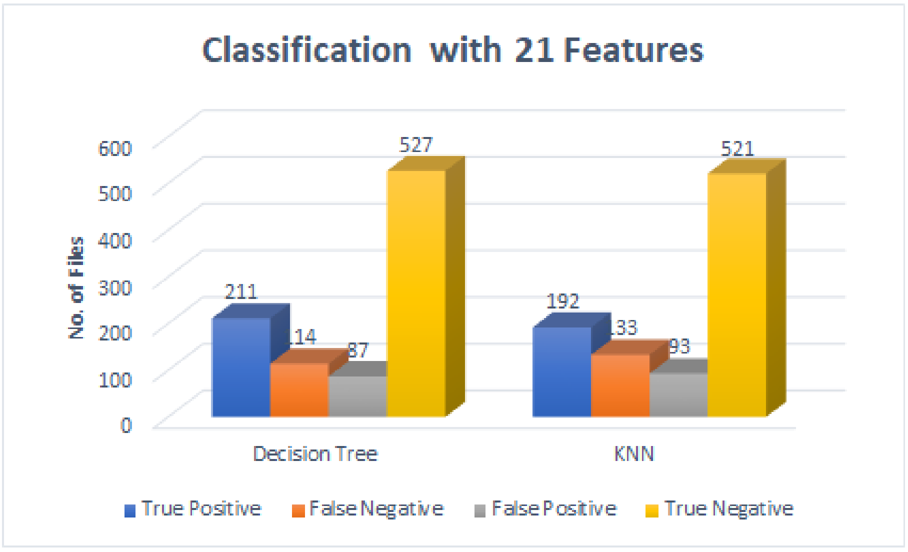


Figure 4.2: Confusion Matrix Statistics for Virus and Trojans

4.3.1.3 File containing instances of only Worms and Trojans

In this file there were a total of 3145 instances consisting of 2117 worms and 1028 trojans. We applied decision tree algorithm on 70% of the training data. A total of 2201 different instances were trained including 1471 worms instances and 731 instances of trojans. As the classifier was trained, we performed our test on remaining 30% of the data.

The model predicted 573 instances out of 647 correctly as worms and 286 instances out 297 correctly as trojans giving us the accuracy of 90.99%.

Then we applied the KNN classifier on the same number of instances and observed the results. The KNN classifier after testing the data gave the accuracy of 90.46%.

The confusion matrix of this experiment is given below in Table 4.4.

Models		**Predicted Labels**			
		N(944)	Trojans	Worms	Accuracy
Decision Tree	**Actual Labels**	Trojan	286	11	90.99%
		Worms	74	573	
K-Nearest Neighbor		Trojan	273	24	90.46%
		Worms	66	581	

Table 4.4: Confusion Matrix of Worms and Trojans

The comparative results of the above table are shown graphically in Figure4.2.

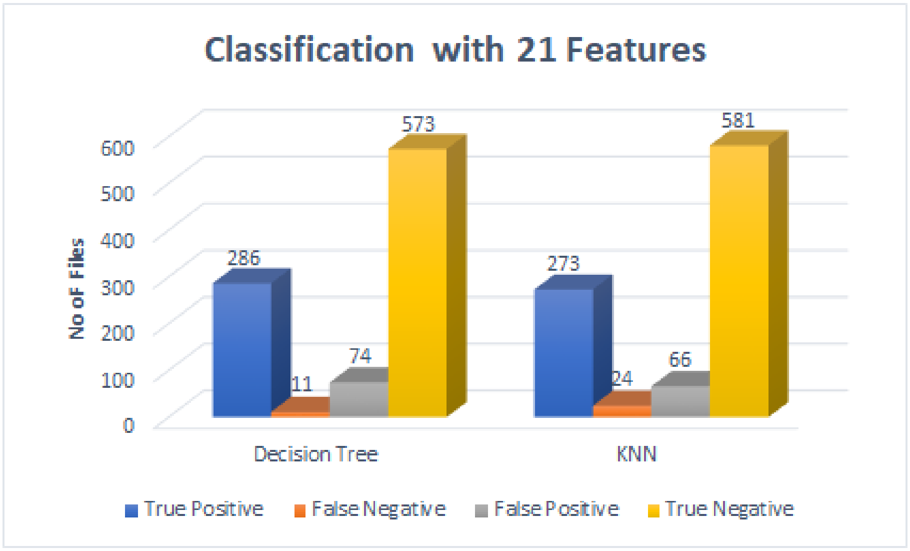


Figure 4.3: Confusion Matrix for Worms and Trojans

4.3.1.4 File containing instances of Virus, Worms and Trojans

Our first dataset comprises of 21 features as discussed in Table 3.5. We applied the decision tree algorithm by using default parameters and obtained the results. But that was just the one experiment and we had to train our dataset so that it gives us the best results. There was no depth control and no criteria defined in the first experiment.

There are total 5246 instances consisting of 1028 trojans instances, 2101 instances of virus and 2116 instances of worms. A total of 3672 instances were trained in which 1470 were viruses, 737 were trojans and 1464 were worms which concludes a total of 70%.As it is a multi-class problem,so there will be a confusion matrix of 3x3. After training upon these instances, we will test the data over the training model.30% of the data include 1574 files of different categories. 197 instances out of 291 were correctly labeled as Trojans,516 instances out of 631 were correctly identifies as Viruses and 584 out of 652 were identified correctly as Worms giving us the accuracy rate of 82.4% on decision tree with *max_depth=9*.

We are not done here with our result. We have just applied decision tree on this dataset. Lets apply KNN on the same dataset and observe the difference of results.The KNN provided the accuracy rate of 80.87%.

The confusion matrix of these two classifiers is given below in Table 4.5.

Models		**Predicted Labels**				
		N(1574)	Trojans	Virus	Worms	Accuracy
Decision Tree	**Actual Labels**	Trojan	197	78	16	82.4%
		Virus	100	516	15	
		Worms	46	22	584	
K-Nearest Neighbor		Trojan	179	107	5	80.87%
		Virus	111	516	4	
		Worms	44	30	578	

Table 4.5: Confusion Matrix of Worms,Trojans and Virus

If we perform the classifier without any criteria, it gives us the accuracy of 78.97%. The depth of the tree improves its accuracy and if we calculate the entropy of each node. As we start from minimum, selecting the *max_depth=3* and going on higher,lets compute accuracy level by level.The increment of depth and change of accuracy can be seen in the Table 4.6.

Depth Level	Accuracy
3	0.7668
4	0.7782
5	0.7795
6	0.805
7	0.809
8	0.817
9	0.824
10	0.815
11	0.813
12	0.802
No Depth	0.7897

Table 4.6: Decision Tree Results by increasing max_depth

The graphical representation of varying the max_depth by increasing or decreasing is shown in the Figure 4.4.

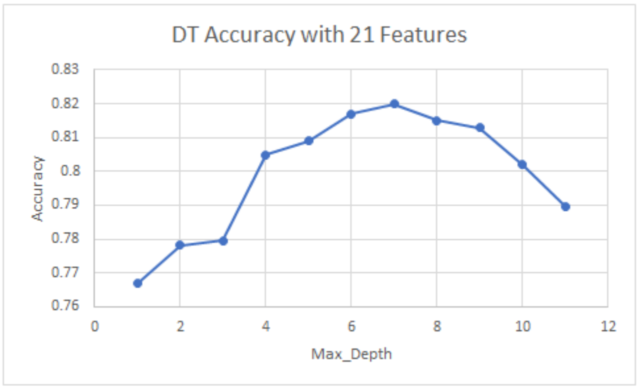


Figure 4.4: Accuracy of the Classifier

In the graph,we observed that as the depth level increases, the accuracy increases but at certain limit. As the depth level reaches 9, it reaches its peak. After this point,the accuracy started to decrease. This happens because the presence of some features are reducing the accuracy. So we have to remove those unimportant features. As we cannot delete any feature randomly, as it will affect its accuracy. So we have to compute the features importance based on their labels.

We computed the importance of each feature corresponding its label using decision tree classifier.Table 4.7 shows the importance of these features.

S.No	Features	Importance
1	e_ovno	0.000825
2	e_minalloc	0.001115
3	MinorLinkerVersion	0.00212
4	e_cp	0.002739
5	e_cblp	0.003723
6	SizeOfUninitializedData	0.004106
7	SizeOfStackCommit	0.005187
8	BaseOfCode	0.00607
9	Characteristics	0.011478
10	SizeOfHeaders	0.016447
11	MajorLinkerVersion	0.01656
12	MajorImageVersion	0.021837
13	NumberOfSections	0.027855
14	BaseOfData	0.030372
15	SizeOfCode	0.047802
16	SizeOfInitializedData	0.049985
17	SizeOfImage	0.052247
18	e_lfanew	0.061553
19	MajorOperatingSystemVersion	0.070637
20	AddressOfEntryPoint	0.088897
21	ImageBase	0.481629

Table 4.7: Features Importance

As we can see clearly, the feature 1,2,3,4 have quite low value. We set the threshold at 0.0035 and neglected the others having value lower than the selected threshold. By doing this , our 4 features were removed including e_ovno , e_minalloc , MinorLinkerVersion and e_cp.

Now at 17 features, we can apply the Wrapper method. We used the forward wrapper method. Forward Wrapper method is an iterative method which has no feature in start and then it adds feature one by one which best improves the model. Although the dataset contains 17 features, so number of combinations can be calculated by the following formula.

$$No.of combinations = 2^{n-1}$$

where 'n' is the number of features in the dataset.

The wrapper method took about 3-4 days for computing 1,31,072 different combinations providing different accuracy with different features. After complete execution of wrapper method, it provided us with following 9 features shown in Table 4.8.

S.No	Features
1	SizeOfUninitializedData
2	BaseOfCode
3	Characteristics
4	MajorLinkerVersion
5	MajorImageVersion
6	NumberOfSections
7	e_lfanew
8	MajorOperatingSystemVersion
9	ImageBase

Table 4.8: Features after Wrapper Method

We removed the remaining features from the data set and selected the top 9 features.Now we have a dataset with reduced features.We will try new experiments with these new features and test the data again.

4.3.2 Experiments with 9 Features

Just like we created different files in previous experiments,we will create the same files but now with 9 features. All the other features will be removed from the file and then we will start our experiments.

We created 4 different CSV files according to following statistics.

1. File containing instances of only Virus and Worms.
2. File containing instances of only Trojans and Worms.
3. File containing instances of only Virus and Trojans.
4. File containing instances of Trojans, Virus and Worms.

4.3.2.1 File containing instances of only Virus and Worms.

Just like previously, we separated the file containing the instances of only Virus and worms. There were a total of 4218 instances containing 2102 viruses and 2116 worms.Again, we repeated the same procedure but this time our model is being trained on 9 features instead of 21 features. Training was done on 70% of the data which makes a total of 2951 instances containing 1456 virus instances and 1496 instances of worms.

30% of the data will be used for testing. Our model predicted 640 instances out of 646 correctly as viruses and 555 instances out of 620 correctly as worms.The accuracy achieved by this model was 94.4%.

Then we applied KNN on the same dataset.The KNN classifier gave the accuracy of 93.44%. The confusion matrix of this experiment is shown in the Table 4.9

Models		**Predicted Labels**			
		N(1266)	Virus	Worms	Accuracy
Decision Tree	**Actual Labels**	Virus	640	6	94.39%
		Worms	65	555	
K-Nearest Neighbor		Virus	614	32	93.44%
		Worms	51	569	

Table 4.9: Confusion Matrix of Worms and Virus

The above results are also shown in graphical form in the Figure 4.5.

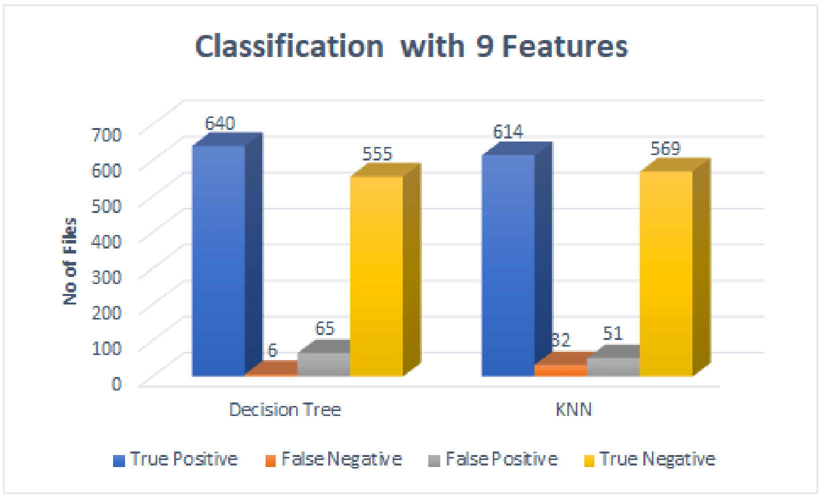


Figure 4.5: Confusion Matrix Statistics for Worms and Virus

4.3.2.2 File containing instances of only Trojans and Worms.

In this file there were a total of 3145 instances consisting of 2117 worms and 1028 trojans. We applied decision tree algorithm on 70% of the training data. A total of 2291 different instances were trained including 1474 worm instances and 727 instances of trojans. As the classifier was trained, we performed our test on remaining 30% of the data.

The model predicted 295 instances out of 301 correctly as Trojans and 567 instances out 643 correctly as Worms giving us the accuracy of 91.31%.

Then we applied the KNN classifier on the same number of instances and observed the results. The KNN classifier after testing the data gave the accuracy of 89.83%.

The confusion matrix of this experiment is given below in Table 4.10.

Models		**Predicted Labels**			
		N(944)	Trojan	Worms	Accuracy
Decision Tree	**Actual Labels**	Trojan	295	6	91.31%
		Worms	76	567	
K-Nearest Neighbor		Trojan	276	25	89.83%
		Worms	71	572	

Table 4.10: Confusion Matrix of Worms and Trojans

The above results are also shown in graphical form in the Figure 4.6.

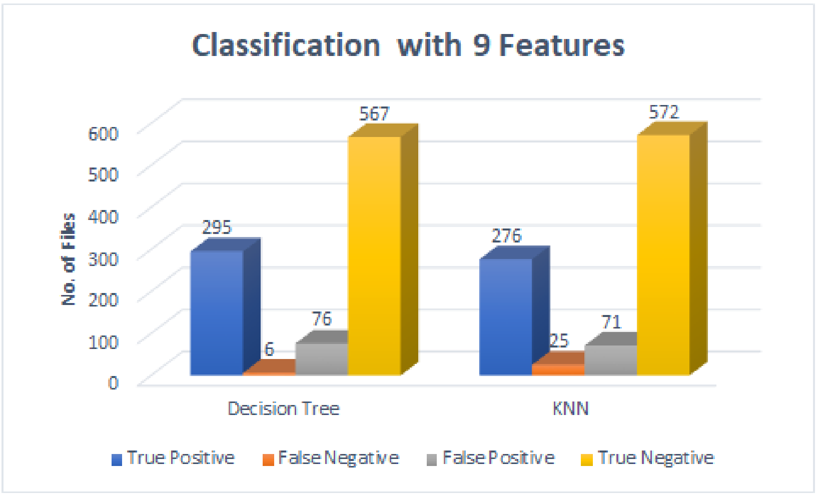


Figure 4.6: Confusion Matrix Statistics for Worm and Trojans

4.3.2.3 File containing instances of only Virus and Trojans

In this file there were a total of 3130 instances consisting of 2101 viruses and 1028 trojans. We applied decision tree algorithm on 70% of the training data. A total of 2197 different instances were trained including 1482 virus instances and 708 instances of trojans. As the classifier was trained, we performed our test on remaining 30% of the data.

The model predicted 241 instances out of 320 correctly as Trojans and 499 instances out 619 correctly as trojans giving us the accuracy of 78.9%.

Then we applied the KNN classifier on the same number of instances and observed the results. The KNN classifier after testing the data gave the accuracy of 77.10%.

The confusion matrix of this experiment is given below in Table 4.11.

Models		**Predicted Labels**			
		N(939)	Trojan	Virus	Accuracy
Decision Tree	**Actual Labels**	Trojan	241	79	78.8%
		Virus	120	499	
K-Nearest Neighbor		Trojan	178	142	77.10%
		Virus	73	546	

Table 4.11: Confusion Matrix of Virus and Trojans

The above results are also shown in graphical form in the Figure 4.5.

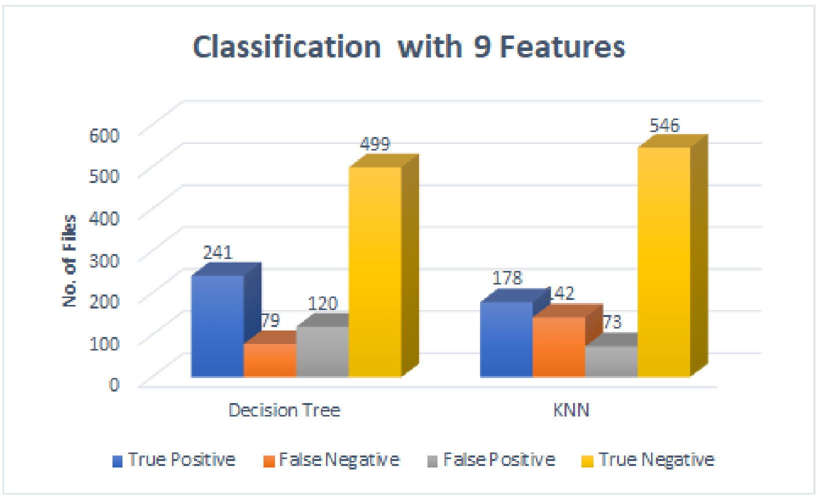


Figure 4.7: Confusion Matrix Statistics for Trojans and Virus

4.3.2.4 File containing instances of Trojans, Virus and Worms.

This file contains all the instances of 3 lables. Now we have to test out classifier with 9 features on our test data. The confusion matrix of 3x3 is shown in the Table 4.12.

Models		**Predicted Labels**				
		N(1574)	Trojans	Virus	Worms	Accuracy
Decision Tree	**Actual Labels**	Trojan	191	98	1	83.45%
		Virus	76	540	0	
		Worms	49	37	582	
K-Nearest Neighbor		Trojan	194	98	2	80.87%
		Virus	99	523	8	
		Worms	58	25	567	

Table 4.12: Confusion Matrix of Trojans,Worms and Virus

4.4 Comparison of Results

In the start, we were training our module on 21 features. Then we did the depth control and recorded the results. After the wrapper method, our features were reduced to 9 and then we performed all the experiments again with two different algorithms and showed their table and graphs.Figure 4.8, shows the difference of results when we were controlling our depth with 21 features comparing with the result of 9 features.

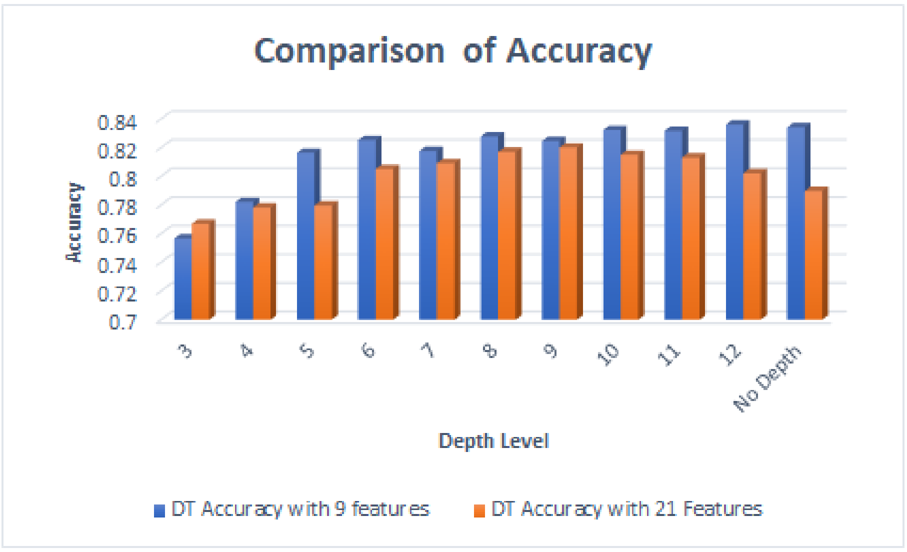


Figure 4.8: Comparison Before and After Feature Reduction

Chapter 5

Conclusions & Future Work

5.1 Conclusion

Cyber-attacks can cause a lot of damage to Information Systems and data in the form of proliferation, theft or total destruction of data. Cyber security has thus been a very vital area in the field of Information Technology. With the increased use of internet and connected devices like those in case of Internet of Things poses a serious challenge to the security of these systems. Moreover, with the increase in sophistication of technology, the dynamics of cyber attacks have become difficult to cater as well. Cyber criminals try to figure out new ways to bypass security measures and evade the detection systems. In the light of this problem and in order to ensure the protection of Information Technology assets, a lot of research and practical work has been done in this area to detect the occurrences of cyber-attacks.

5.2 Future Work

As we have now addressed the problem of 3 major classes of Malwares - Virus, Worms and Trojans. There are many other families of malware which can also be addressed in the future.A good and efficient mechanism can be developed which will predict the family of malware just by looking at the header.We have just worked on the header features of the Portable Executable file.There are many things which can be addressed in the future like looking at the Strings values, System calls imports,exports and many others.That will be a good contribution in this area of research field.There is still room for improvement in the detection system in terms of addition of new attack categories alongside the already existing attack categories.The inclusion of more categories will enable the engine to be even better in prediction of vast majority of attacks on the system where it is deployed.

Furthermore, there can also be another direction. Just like we have addressed three major categories of malwares, one can study about a single family and predict about the sub-families. As there are subfamilies of trojans, each of them may have different structure or mechanism of execution. They may impact differently depending upon their families. So that would be a very good future perspective if someone carries out that work.

Bibliography

[1] R. R. Ravula, *Classification of Malware using Reverse Engineering and Data Mining Techniques*. PhD thesis, University of Akron, 2011. Cited on p. 3.

[2] J. Li and S. Stafford, “Detecting smart, self-propagating internet worms,” in *2014 IEEE Conference on Communications and Network Security*, pp. 193–201, IEEE, 2014. Cited on p. 3.

[3] H. R. Zeidanloo, F. Tabatabaei, P. V. Amoli, and A. Tajpour, “All about malwares (malicious codes).,” in *Security and Management*, pp. 342–348, 2010. Cited on p. 3.

[4] N. Idika and A. P. Mathur, “A survey of malware detection techniques,” *Purdue University*, vol. 48, 2007. Cited on p. 3.

[5] “Idapro..” Cited on p. 3.

[6] “Ollydbg.” Cited on p. 3.

[7] “Lordpe.” Cited on p. 3.

[8] “Ollydump.” Cited on p. 3.

[9] A. Moser, C. Kruegel, and E. Kirda, “Limits of static analysis for malware detection,” in *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)*, pp. 421–430, IEEE, 2007. Cited on p. 3.

[10] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, “A survey on automated dynamic malware-analysis techniques and tools,” *ACM computing surveys (CSUR)*, vol. 44, no. 2, p. 6, 2012. Cited on p. 4.

[11] R. Tahir, “A study on malware and malware detection techniques,” *International Journal of Education and Management Engineering*, vol. 8, no. 2, p. 20, 2018. Cited on p. 5.

[12] J. Landage and M. Wankhade, "Malware and malware detection techniques: A survey," *International Journal of Engineering Research and Technology (IJERT)*, vol. 2, no. 12, pp. 2278–0181, 2013. Cited on p. 5.

[13] E. Gandotra, D. Bansal, and S. Sofat, "Malware analysis and classification: A survey," *Journal of Information Security*, vol. 5, no. 02, p. 56, 2014. Cited on p. 9.

[14] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Transactions on Knowledge & Data Engineering*, no. 4, pp. 491–502, 2005. Cited on p. 10.

[15] G. Kumar, K. Kumar, and M. Sachdeva, "The use of artificial intelligence based techniques for intrusion detection: a review," *Artificial Intelligence Review*, vol. 34, no. 4, pp. 369–387, 2010. Cited on p. 11.

[16] A. Este, F. Gringoli, and L. Salgarelli, "Support vector machines for tcp traffic classification," *Computer Networks*, vol. 53, no. 14, pp. 2476–2490, 2009. Cited on p. 11.

[17] M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo, "Data mining methods for detection of new malicious executables," in *Proceedings 2001 IEEE Symposium on Security and Privacy. S&P 2001*, pp. 38–49, IEEE, 2000. Cited on p. 11.

[18] W. W. Cohen, "Fast effective rule induction," in *Machine Learning Proceedings 1995*, pp. 115–123, Elsevier, 1995. Cited on p. 11.

[19] J. Z. Kolter and M. A. Maloof, "Learning to detect malicious executables in the wild," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 470–478, ACM, 2004. Cited on pp. 11 and 12.

[20] L. Nataraj, S. Karthikeyan, G. Jacob, and B. Manjunath, "Malware images: visualization and automatic classification," in *Proceedings of the 8th international symposium on visualization for cyber security*, p. 4, ACM, 2011. Cited on p. 12.

[21] L. Nataraj, V. Yegneswaran, P. Porras, and J. Zhang, "A comparative assessment of malware classification using binary texture analysis and dynamic analysis," in *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, pp. 21–30, ACM, 2011. Cited on p. 12.

[22] D. Kong and 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, 2013. Cited on p. 12.

[23] R. Tian, L. M. Batten, and S. Versteeg, “Function length as a tool for malware classification,” in *2008 3rd International Conference on Malicious and Unwanted Software (MALWARE)*, pp. 69–76, IEEE, 2008. Cited on p. 12.

[24] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The weka data mining software: an update,” *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009. Cited on p. 12.

[25] I. Santos, J. Nieves, and P. G. Bringas, “Semi-supervised learning for unknown malware detection,” in *International Symposium on Distributed Computing and Artificial Intelligence*, pp. 415–422, Springer, 2011. Cited on p. 12.

[26] R. Moskovitch, D. Stopel, C. Feher, N. Nissim, and Y. Elovici, “Unknown malcode detection via text categorization and the imbalance problem,” in *2008 IEEE International Conference on Intelligence and Security Informatics*, pp. 156–161, IEEE, 2008. Cited on p. 12.

[27] I. Santos, C. Laorden, and P. G. Bringas, “Collective classification for unknown malware detection,” in *Proceedings of the International Conference on Security and Cryptography*, pp. 251–256, IEEE, 2011. Cited on p. 12.

[28] M. Siddiqui, M. C. Wang, and J. Lee, “Detecting internet worms using data mining techniques,” *Journal of Systemics, Cybernetics and Informatics*, vol. 6, no. 6, pp. 48–53, 2009. Cited on p. 12.

[29] M. F. Zolkipli and A. Jantan, “An approach for malware behavior identification and classification,” in *2011 3rd International Conference on Computer Research and Development*, vol. 1, pp. 191–194, IEEE, 2011. Cited on p. 12.

[30] C. Willems, T. Holz, and F. Freiling, “Toward automated dynamic malware analysis using cwsandbox,” *IEEE Security & Privacy*, vol. 5, no. 2, pp. 32–39, 2007. Cited on p. 12.

[31] “Anubis.” Cited on p. 12.

[32] K. Rieck, P. Trinius, C. Willems, and T. Holz, "Automatic analysis of malware behavior using machine learning," *Journal of Computer Security*, vol. 19, no. 4, pp. 639–668, 2011. Cited on p. 12.

[33] B. Anderson, D. Quist, J. Neil, C. Storlie, and T. Lane, "Graph-based malware detection using dynamic analysis," *Journal in computer Virology*, vol. 7, no. 4, pp. 247–258, 2011. Cited on p. 12.

[34] A. Dinaburg, P. Royal, M. Sharif, and W. Lee, "Ether: malware analysis via hardware virtualization extensions," in *Proceedings of the 15th ACM conference on Computer and communications security*, pp. 51–62, ACM, 2008. Cited on p. 12.

[35] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda, "Scalable, behavior-based malware clustering.," in *NDSS*, vol. 9, pp. 8–11, Citeseer, 2009. Cited on pp. 12 and 13.

[36] R. Tian, R. Islam, L. Batten, and S. Versteeg, "Differentiating malware from cleanware using behavioural analysis," in *2010 5th international conference on malicious and unwanted software*, pp. 23–30, IEEE, 2010. Cited on p. 13.

[37] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario, "Automated classification and analysis of internet malware," in *International Workshop on Recent Advances in Intrusion Detection*, pp. 178–197, Springer, 2007. Cited on p. 13.

[38] Y. Park, D. Reeves, V. Mulukutla, and B. Sundaravel, "Fast malware classification by automated behavioral graph matching," in *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*, p. 45, ACM, 2010. Cited on p. 13.

[39] I. Firdausi, A. Erwin, A. S. Nugroho, *et al.*, "Analysis of machine learning techniques used in behavior-based malware detection," in *2010 second international conference on advances in computing, control, and telecommunication technologies*, pp. 201–203, IEEE, 2010. Cited on p. 13.

[40] I. Santos, J. Devesa, F. Brezo, J. Nieves, and 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, 2013. Cited on p. 13.

[41] R. Islam, R. Tian, L. M. Batten, and S. Versteeg, "Classification of malware based on integrated static and dynamic features," *Journal of Network and Computer Applications*, vol. 36, no. 2, pp. 646–656, 2013. Cited on p. 13.

[42] B. Anderson, C. Storlie, and T. Lane, "Improving malware classification: bridging the static/dynamic gap," in *Proceedings of the 5th ACM workshop on Security and artificial intelligence*, pp. 3–14, ACM, 2012. Cited on p. 13.

[43] R. Perdisci, A. Lanzi, and W. Lee, "Mcboost: Boosting scalability in malware collection and analysis using statistical classification of executables," in *2008 Annual Computer Security Applications Conference (ACSAC)*, pp. 301–310, IEEE, 2008. Cited on p. 18.

[44] VirusTotal, "Virustotal." Cited on p. 19.

[45] R. Masri and M. Aldwairi, "Automated malicious advertisement detection using virustotal, urlvoid, and trendmicro," in *2017 8th International Conference on Information and Communication Systems (ICICS)*, pp. 336–341, IEEE, 2017. Cited on p. 20.

[46] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artificial intelligence*, vol. 97, no. 1-2, pp. 245–271, 1997. Cited on p. 23.

[47] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003. Cited on p. 23.

[48] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature extraction: foundations and applications*, vol. 207. Springer, 2008. Cited on p. 23.