



T.C
ALTINBAS UNIVERSITY
Institute of Graduate Studies
Electrical and Computer Engineering

**AUTOMATIC ANALYSIS AND DETECTION OF
MALWARE BEHAVIOR USING MACHINE LEARNING
FOR COMPUTER DEVICE**

Abdullah Qassim Ali

Master Thesis

Supervisor
Prof. Dr. Osman Nuri UÇAN

Istanbul, 2020

**AUTOMATIC ANALYSIS AND DETECTION OF MALWARE BEHAVIOR
USING MACHINE LEARNING FOR COMPUTER DEVICE**

by

Abdullah Qassim Ali

Electrical and Computer Engineering

Submitted to the Institute of Graduate Studies

in partial fulfillment of the requirements for the degree of

Master of Science

ALTINBAŞ UNIVERSITY

2020

The thesis titled “HIGHWAYS TRAFFIC SAFETY ENHANCEMENT USING MOBILE VEHICLES AS RSU” prepared and presented by ahmed naseer saleh OBAIDA was accepted as a Master of Science Thesis in Electrical and Computer Engineering.

Prof. Dr. Osman Nuri UÇAN

Supervisor

Thesis Defense Jury Members:

Prof. Dr. Osman Nuri UÇAN

School of Engineering and
Natural Sciences,
Altinbas University

Asst. Prof. Dr. Abdullahi Abdu
Ibrahim

School of Engineering and
Natural Sciences,
Altinbas University

Asst. Prof. Dr. Izzet Parug DURU

Vocational School ,
Istanbul Gedik University

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Approval Date of Institute of Graduate Studies:

____/____/____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and conduct. I also declare that, as required by these rules and regulations, I have fully cited and referenced all material and results that are not original to this work.

Abdullah Qassim Ali

ACKNOWLEDGEMENTS

I might want to thank earnestly Prof. Dr.Osman Nuri UÇAN for his direction, getting, persistence and in particular, his kinship during my alumni learns at Altinbas University. His mentorship was principal in furnishing a well-round encounter predictable with my long haul profession objectives.



ABSTRACT

AUTOMATIC ANALYSIS AND DETECTION OF MALWARE BEHAVIOR USING MACHINE LEARNING FOR COMPUTER DEVICE

Ali, Abdullah Qassim

M.Sc., Electrical and Computer Engineering, Altınbaş University,

Supervisor: Prof. Dr. Osman Nuri UÇAN

Date: 09/2020

Pages: 120

Malicious software, known as malware, puts the protection of computers at major risk. Classic safety defenses are ineffective because of the number and variety of variants, and millions of hosts on the network are compromised by malware including computer viruses, web worm and trojan-horsed. Although malware obstruction and multiformity largely hinder file detection, dynamic malware-binary analysis during runtime offers a method for detecting and defending malicious software against danger. We suggest a framework in this article for the automated analysis of malware behavior by means of computer learning. This system allows new malware classes that have a common behavior (clustering) to be automatically detected and unknown malware for the classes (classification) that have been found. Incremental behavior analysis, which can analyze the actions of thousands of malwares on a daily basis, is implied by both clustering and classification. The radical research significantly reduces the overtime of current test methods while ensuring that novel malware variants are correctly detected and discriminated against.

Keywords: Computer virus detection, Machine Learning, classification behavior, Clustering behavior, Malware Detection

TABLE OF CONTENTS

	<u>Pages</u>
ABSTRACT.....	vi
LIST OF TABLES	xii
LIST OF FIGURES	xiii
LIST OF ABBREVIATIONS	xv
1. INTRODUCTION.....	1
1.1 OVERVIEW	1
1.2 RATIONALE FOR THE STUDY	3
1.3 THE PROBLEM STATEMENT	3
1.4 THE PROPOSED RESEARCH FRAMEWORK.....	5
1.5 ORGANIZATION OF THE THESIS	6
2. RELATED WORK	7
2.1 MALWARE: A CYBER THREAT	7
2.1.1 What is Malware?.....	9
2.1.2 What is Cleanware?.....	9
2.1.3 Behavior-Based Detection	9
2.1.4 Malware Families	9
2.1.5 Naming Malware.....	10
2.1.6 Types of Malware.....	10
2.1.6.1 Virus.....	11
2.1.6.2 Worm	12
2.1.6.3 Trojan	13
2.1.6.4 Backdoor	13
2.1.6.5 Rootkit.....	13
2.1.6.6 Spyware and adware	14
2.2 FEATURES USED WITH ML ALGORITHMS FOR MALWARE DETECTION	14

2.2.1	Static Features	15
2.2.2	Dynamic Features.....	16
2.2.3	Hybrid Features	17
2.3	TYPE OF DETECTION.....	17
2.4	MALWARE DETECTION TYPE.....	18
2.4.1.1	Signature-based detection	19
2.4.1.2	Anomaly-based detection.....	20
2.4.1.3	Specification-based detection.....	21
2.5	NEED FOR MACHINE LEARNING	22
2.6	MACHINE LEARNING METHODS FOR CLASSIFICATION REVIEW	23
2.6.1	Random Forest	24
2.6.2	Naïve Bayes	25
2.6.3	K-Nearest Neighbour.....	25
2.6.4	Artificial Neural Networks	26
2.6.5	AdaBoost	27
2.6.6	Evaluation Classifiers	28
2.7	TAXONOMY OF MACHINE LEARNING TECHNIQUES FOR MALWARE ANALYSIS.....	29
2.7.1	Malware Analysis Objectives	29
2.7.1.1	Malware detection.....	30
2.7.1.2	Malware similarity analysis.....	30
2.7.1.3	Malware category detection	31
2.7.2	Malware Analysis Features.....	32
2.7.2.1	Feature extraction	32
2.7.3	Malware Analysis Algorithms	33
2.7.3.1	Supervised learning.....	33
2.7.3.2	Unsupervised learning.....	33
2.7.3.3	Semi-supervised learning	33

2.8	MALWARE DETECTION RELATED WORK	33
2.9	CONCLUSION.....	40
3.	ANALYSIS AND DETECTION OF MALWARE BEHAVIOR USING MACHINE LEARNING FOR COMPUTER DEVICE	41
3.1	OVERVIEW	41
3.2	MODELLING.....	42
3.2.1	Selection Machine Learning Techniques.....	43
3.3	AIM OF THE PROJECT.....	43
3.3.1	Methodology	43
3.3.2	Main Issue with Network Security Data.....	44
3.4	DATA ANALYSIS	45
3.4.1	CTU-13 Dataset.....	45
3.5	MACHINE LEARNING METHODS.....	46
3.5.1	Machine Learning Basics.....	46
3.5.1.1	Feature extraction	47
3.5.1.2	Supervised and unsupervised learning.....	48
3.5.2	Classification Used Methods	49
3.5.2.1	K-nearest neighbours	50
3.5.2.2	Support vector machines	51
3.5.2.3	Naive Bayes.....	53
3.5.2.4	J48 decision tree	54
3.5.2.5	Random forest random.....	57
3.5.3	Cross-Validation.....	59
3.6	CONCLUSION.....	59
4.	PRACTICAL PART	61
4.1	DATA.....	61
4.1.1	Dridex	61
4.1.2	Locky.....	63

4.1.3	Teslacrypt.....	64
4.1.4	Vawtrak.....	65
4.1.5	Zeus	66
4.1.6	DarkComet.....	67
4.1.7	CyberGate	68
4.1.8	Xtreme	68
4.1.9	CTB-Locker	68
4.2	CUCKOO SANDBOX.....	69
4.3	SCORING SYSTEM.....	70
4.3.1	Reports and Features	72
4.3.1.1	Files.....	73
4.3.1.2	Registry keys	73
4.3.1.3	Mutexes	74
4.3.1.4	Processes	74
4.3.1.5	IP addresses and DNS queries.....	74
4.3.1.6	API calls	74
4.4	FEATURE REPRESENTATION.....	75
4.4.1	Binary Representation	75
4.4.2	Feature Selection	75
4.5	IMPLEMENTATION	76
4.5.1	Sandbox Configuration.....	76
4.5.2	Feature Extraction	77
4.5.3	Feature Selection	78
4.5.4	Application of Machine Learning Methods.....	79
4.6	RESULTS AND DISCUSSION.....	80
4.6.1	K-Nearest Neighbors.....	80
4.6.2	Support Vector Machines	84

4.6.3	J48 Decision Tree.....	86
4.6.4	Naive Bayes	90
4.6.5	Random Forest	93
4.7	CONCLUSIONS.....	96
5.	CONCLUSION AND FUTURE WORK.....	97
5.1	CONCLUSIONS.....	97
5.2	FUTURE WORK.....	98
	REFERENCES.....	100

LIST OF TABLES

	<u>Pages</u>
Table 1.1: Various types of malware (Source Peng et al 2013)	12
Table 3.1: Decision tree example dataset	55
Table 4.1: Cuckoo scores for malware families	72
Table 4.2: KNN multi-class accuracy	82
Table 4.3: KNN binary accuracy	83
Table 4.4: SVM CrossTable	84
Table 4.5: SVM multiclass accuracy	85
Table 4.6: SVM binary classification accuracy	86
Table 4.7: Decision Tree multi-class accuracy	89
Table 4.8: Decision Tree binary classification accuracy	90
Table 4.9: Naive Bayes multi-class classification accuracy	92
Table 4.10: Naive Bayes binary classification accuracy	93
Table 4.11: Random Forest multiclass classification accuracy	95
Table 4.12: Random Forest binary classification accuracy	96

LIST OF FIGURES

	<u>Pages</u>
Figure 2.1: Total and new malware during the period 2013-17	8
Figure 2.2: Malware type distribution according to Google [9] (left) and Kaspersky [10] (right)11	11
Figure 2.3: Types of features used for building machine learning based malware classifier	16
Figure 2.4: Types of Malware Detection Techniques	19
Figure 2.5: Signature-based detection method	20
Figure 2.6: Anomaly-based detection method.....	21
Figure 2.7: Specification-based detection method.....	22
Figure 2.8 : Example of Decision Tree	24
Figure 2.9: Example of K-Nearest Neighbour Classification.....	26
Figure 2.10: Example of an Artificial Neural Network	27
Figure 2.12: Performance Metrics	29
Figure 2.13: Taxonomy of machine learning techniques for malware analysis	30
Figure 3.1: Modelling activities and output	42
Figure 3.2: Scenarios of the CTU-13 Dataset (CTU University 2011)	45
Figure 3.3: General workflow process	46
Figure 3.4: KNN example	50
Figure 3.5: SVM scheme	52
Figure 3.6: Decision tree example	55
Figure 3.7: Random Forest scheme.....	58
Figure 4.1: Dridex operation scheme (Aquino 2014)	62
Figure 4.2: Locky operation	63
Figure 4.3: TeslaCrypt operation	64
Figure 4.4: Vawtrak operation.....	65
Figure 4.5: Zeus operation.....	66
Figure 4.6: DarkComet communication scheme	67
Figure 4.7: CTB-Locker operation	69
Figure 4.8: Severity levels of cuckoo signatures	71
Figure 4.9: Feature extraction process	77

Figure 4.10: Machine learning classification scheme	80
Figure 4.11: CrossTable for KNN multi-class classification	81
Figure 4.12: CrossTable for KNN binary classification	83
Figure 4.13: SVM binary classification CrossTable	85
Figure 4.14: Binary Decision Tree.....	87
Figure 4.15: Decision Tree multi-class CrossTable.....	88
Figure 4.16: Decision Tree binary classification CrossTable	89
Figure 4.17: Naive Bayes multi-class classification cross-table	91
Figure 4.18: Naive Bayes binary classification cross-table.....	92
Figure 4.19: Random Forest multiclass classification cross-table.....	94
Figure 4.20: Random Forest binary classification cross-table	95
Figure 5.1: Results	Error! Bookmark not defined.

LIST OF ABBREVIATIONS

API : Application Programming Interface

TOCTOU : Time of Check to Time of Use

SVM : Support Vector Machine

KNN : K-Nearest Neighbors

ID3 : Iterative Dichotomiser 3

VNC : Virtual Network Computing

1. INTRODUCTION

1.1 OVERVIEW

Computer viruses, worms, Trojans, malware (in short, malware) and other malicious software are as long as computers. In 1969 the primary correspondence framework showed up and just two years after the fact an infection spread. The infection was designated "Creeper" and a trial try indicated that a framework. "I am the creeper on the off chance that you can trap me." It reported as it were.

At the point when the PCs pushed ahead and took their places in governments around the globe, uncommon infections developed to upset this utilization. The programmer bunches far and wide either attempt to simply over-burden and execute some taxpayer driven organization or take some mystery information. In the two cases there is normally some malware engaged with the activity. The most well-known programmer bunch is likely Anonymous [1]. Obscure is the greatest secretive social affair of software engineers and political activists on the Internet [2]. They are generally centering for gigantic degree DDoS (scattered refusal of organization) ambushes, which are an undertakings to make server or other framework resource difficult to reach [3]. The idea behind this kind of attack is to run a bot, on anyway numerous PCs as could be normal considering the present situation. The bot attempts to over-burdensome site page or another sort of administration by rehashing demands. Such a bot can be either disseminated deliberately between individuals from the affiliation or be a piece of some malware dispersed on client's PCs. The clients don't require understand they are a piece of antigovernment offense. In increasingly genuine cases the programmer bunches are taking touchy data from government databases. Most pulling in to programmers are, obviously, cash related exchanges. As e-shops are taking offers available and supplanting normal stores, the vast majority of the installments move into the Internet also. This open door isn't discarded by malware makers. So malware centers on steeling of your financial balances, Mastercards numbers, etc. Tragically, this product shouldn't be extremely refined considering the low measure of consideration individuals pay while filling in reasonable information. As an outcome the product regularly can be only a phony page requesting that you fill in the data being referred to. 2 In brief time span there were progressively increasingly tantamount contaminations and worms. Obviously the most idea

about them everything was "I love you" worm written in Visual Basic. This worm alone contaminated 10% considering and caused hurt worth about \$8 billion [4]. Another \$15 billion were spent to purge the sully far and wide. These days, each and every PC, tablet and even PDA needs to run an antivirus framework. Dependably there are endless new infections attempting to imperil your PC. Then again, just infrequently somebody composes new malware without any preparation. Frequently, the new malware is only some transformation of recently composed malware, subsequently it is normally like its ancestor. Discovering this fluffy similitude is practically equivalent to numerous man-made brainpower assignments, consequently we attempt to utilize man-made consciousness approach for malware related undertakings.

Cell phone is getting hugely well-known everywhere throughout the world. It has gotten the individual remote of life than simply being a medium to impart inferable from the colossal usefulness it offers. It has changed into everybody's online bank, web based shopping center, and online mentor alongside the customary voice correspondence office. It is a Go-to-Device for different everyday exercises, for example, clicking pictures, watching motion pictures, shopping, calling, visiting and substantially more. Cell phones are furnished with cutting edge UIs, preparing ability, and sufficient memory. Cell phones hold loads of individual data like contact list, internet banking passwords, charge card subtleties and area of the owner and so on. As indicated by a study by Pew Research Center, the overall utilization of Smartphone has expanded from a middle of 45% to 54% during the range of 2013 to 2015 [5]. As indicated by Statista, the general number of cell phone clients arrived at 4.43 billion of every 2015 and expected to arrive at 4.77 billion out of 2017 [6].

This theory entitled "A Framework for Malware Detection with Static Features utilizing Machine Learning Algorithms" intends to improve malware location and conquer the confinements of conventional mark based identification strategies. The inescapability of malware is a set up truth and is a basic danger for a wide range of registering conditions. This examination work centers around Portable Executable (PE) and Android Package Kit (APK), the document designs for Windows and Android Operating System (OS) separately. The determination of these two record positions depends on the most noteworthy number of clients of particular OS. Windows and Android have most elevated number of clients as for their serious OSs for Desktop and

Smartphone gadget separately. This work statically analyzes and uses the structure information of PE record and uses assents removed from show archive of utilization to produce incorporate sets to get ready and test distinctive AI computations for malware acknowledgment.

1.2 RATIONALE FOR THE STUDY

With the introduction of new attack methods and the growing use of the Internet, a number of new vulnerabilities are discovered every day. Over recent years, the field of computer security and network security has evolved significantly. Attacks are now so prevalent that many companies, particularly large financial institutions, directly invest more than 10% of the overall budget on computer and network security.

Enterprises want to boost productivity, increase the efficiency of the workforce and take less time to execute business activities. Business managers look at iPhones, Android devices and even Windows handsets and everywhere see opportunities for growth in sales, cost reduction and improved communication. However, they are concerned that the company's network will be contaminated with devices, data loss and malware through SMPs and tablets.

Changes in the sort of assaults, for example, the utilization of botnets and the distinguishing proof of new vulnerabilities, have brought about an exceptionally unique risk scene that isn't manageable to conventional security draws near. Information mining systems like arrangement (Sami et al 2010) [7] , bunching and affiliation rule mining (Ye et al 2007) [8] calculations which join enlistment calculations that investigate information so as to find concealed examples and create prescient models, have end up being powerful in handling the previously mentioned data security challenges.

1.3 THE PROBLEM STATEMENT

Portable executables and apk have two big Windows and Android OS file formats, so the bulk of wild malware is in both file formats. The practice of "zero day" and "unknown" malware, which includes alternative, signature-based detection techniques are not identified by conventional signature-based Malware techniques. Machine learning methods provide an alternative solution which, using static or dynamic properties, can overcome the restrictions of the traditional system. Malware detection is time consuming due to the exponential growth of the malware number and

the rising complexity. The most common concern for cyber space and the users is in many cases the detection of malware. This work focuses on the research problem as follows: The research problem

To develop a malware detecting system by analyzing the potential of static features using function engineering techniques to improve the identification of features to better identify malware using machine learning algorithms.

The following research questions are used in this study to address the above research problem:

Question 1: How are various machine learning strategies for detecting mobile trojans like Random Wood, K-Nearest Neighbours, Naïve Bays, and Multilayer Perceptrons?

The Random Forest Classification Classificatory, K-Nearest Neighbor and Naive Bayes provided the most promising literature in this study. Neural networks are promising results, but barely investigated for the identification of mobile malware. Neural networks will therefore be investigated along with the above classifications in this study. These works relate to the dynamic detection of mobile malware and the classified results. Chapter 4 outlines the answer to S.Q.1.

Question 2: That software and/or hardware is most important to the identification of Mobile Trojans without root permissions?

The use of battery, CPU and RAM bandwidth is limited to mobile devices. The analysis of the main characteristics in mobile malware detection and the characteristics which can be excluded therefore improves the detection models ' performance. In addition it provides an overview of the characteristics of different subspecies of mobile tropes in the response to this sub-question. Because these observations are focused on real-life information, the results reflect real-life conditions instead of (clean) laboratory conditions. Section 6 explains the answer to S.Q.2.

Question 5: What is the functionality on a specific computer of these different classifiers?

In view of the above tools, this sub-question focuses on the usability of the different classifiers. The efficiency of the various detection models is the use of the device resource (for example battery use, RAM usage). Usability is also evaluated from a market point of view in S.Q.3.

It explores the expense and advantages for an organization that requires or does not use self-made mobile malware detection methods. Chapter 4. Best performance features among different features extracted from PE headers and Android applications explain resource usability and business usability. The working theory of the algorithm is different. So, an algorithm can work well with a set of features, but its output can be reduced with other set of features. This work experiments on the proposed feature set with various ML algorithms to evaluate and choose the best ML algorithm for malware detection.

So as to discover answers for all the above recorded research questions, this examination has conceptualized a layered research system, under which different analyses are done and the means of AI, for example, pre-preparing, include extraction, highlight determination and preparing and testing are finished.

1.4 THE PROPOSED RESEARCH FRAMEWORK

One of the job is to identify malware, which helps malware families to be identified and then effectively treated. We therefore suggest using artificial intelligence approaches, including different types of classifiers (neural networks, vector support machines and random forests). In order to learn anything from the data, each of these methods requires some insightful functionality. The main purpose of this research is to recognize and compare various sets of characteristics which are suited to malware classification. In terms of its utility for the particular purpose we also intend to compare the different feature sets. Lastly, the performance of the systems selected is compared and measured.

We identify and explain our question at the start of this study. In the second chapter we analyze related work in this area and explore some common data classification techniques. Chapter three includes an overview of the data set used for research. The fourth chapter is about extracting functions, and we try to set a benchmark first in a few simple and obvious ways. We then propose a new method for comparing call charts based on our new internal and external call charts. Combining tf-idf with a specific characteristic (instruction histogram) of natural language processing techniques gives one of their greatest features. That is the reason we are attempting to lessen the amount toward the finish of the section. That carried us to three most encouraging arrangements of highlights. We additionally take a gander at current strategies in

bioinformatique and the handling of characteristic systems, explore how to use these approaches for our function and evaluate some of them. Then in the next segment we are reviewing and analyzing more versions for the most promising feature kits. There are some comments on our models, topics and ideas for future study at the very end of the thesis.

1.5 ORGANIZATION OF THE THESIS

- Chapter II presents a study of Malware recognition systems including prior works which have utilized Machine Learning calculations and with static highlights.
- Chapter III presents inferred highlights which are the result of field's estimations of PE headers and related guidelines for the fields. Consolidating determined highlights with 14 crude highlights as coordinated list of capabilities improved the order execution (exactness, TFP and so forth.) of malware classifiers.
- Chapter IV clarifies and presents the capability of area name as twofold component to arrange test into malware and generous class. It requires exceptionally low handling for extraction and list of capabilities arrangement and gives sufficient characterization execution.
- Chapter VI investigate the capability of highlights separated from picture portrayal of Android applications. Each application was changed over to picture dependent on four distinctive shading channels (Grayscale, Red, Green, Blue (RGB), Cyan, Magenta, Yellow, Key (black) (CYMK), and Hue, Saturation, Value (HSV)) and GIST highlights were utilized to prepare and test Machine Learning calculations.
- Chapter VII examines the ends and point out the future degrees and further enhancements.

2. RELATED WORK

This chapter discusses latest works on the identification of malware using static algorithms for machine learning. Research is structured and described in different sections on the identification of malicious Portable Executable (PE) and Android application. In this analysis, the limitations and reach of developments in present methods and techniques are understandable. The whole investigation centers on the current works which have utilized static highlights with AI to recognize malware. In the arrangement of static highlights, works which have utilized Portable Executable (PE) header field as highlights just as Android application's authorizations as highlights or some other static highlights to characterize Android and PE applications have been investigated.

Area 2.1 records and clarifies different static highlights which are utilized with AI calculations. Based on examination type, these static highlights are ordered into static, dynamic and half breed.

2.1 MALWARE: A CYBER THREAT

Today, the World is moving towards advanced period where the utilization of Cyber advances have become a necessary piece of day by day life. The utilization of Computer and Internet isn't just constrained to work force calculation and data access, but its job is likewise reached out to everything by the utilization of innovations, for example, Internet of Things (IoT), digital currency and so on. As of now, the World is discussing computerized economy to digital provinces, such profound inclusion of Computer and different advancements carries new difficulties to the advanced World. Ordinary, individuals and firms are seeing various kinds of digital assault on different digital framework from e-banking to web based business 1 (Anderson et al., 2013)[10] . Phishing, Distributed Denial of Service (DDoS), data fraud, Ransom ware and so on are a portion of the basic assault techniques. A considerable lot of the cutting edge digital assaults are done by the immediate or circuitous utilization of the Malware.

The malware is a computer program that has been developed deliberately for malicious activities such as computer networking attacks, device hijacks, deletion of data, stealing, spamming and malware downloads (Szor, 2005)[11]. The list is very comprehensive and expands at a steady

and frequent pace with new entries. For instance, the malicious list has been targeted by Stuxnet with critical infrastructure (Langner, 2011)[12]. The newly produced malware and malware variants exist in two different classes. Figure 2.1 shows the large growth rate between 2013 and June of 2017 of existing malware and new malware.

Malware complexity in terms of self-mutation, multi-payload 3 (e.g., Nimda worm), multi-platform 4 execution and various obfuscation (You and Yim, 2012) [14] and antimutrient analysis (Bethencourt et al., 2008) [16] is growing in the regular domain of malware production (Bruschi et al., 2006). Their sophistication also is increasing in everyday terms. [14] This sophisticated malware and its exponential growth present major challenges for the anti-malware solutions available. Malware designers five have also changed their motivations, initially for fun, jolt and glory, but now the focus has shifted towards benefit (Caballero et al., 2011)[17], support for terrorism (e.g. Stuxnet) and cyber spying(Zhioua, 2013)[18] (e.g.: flame, duqu, dino).

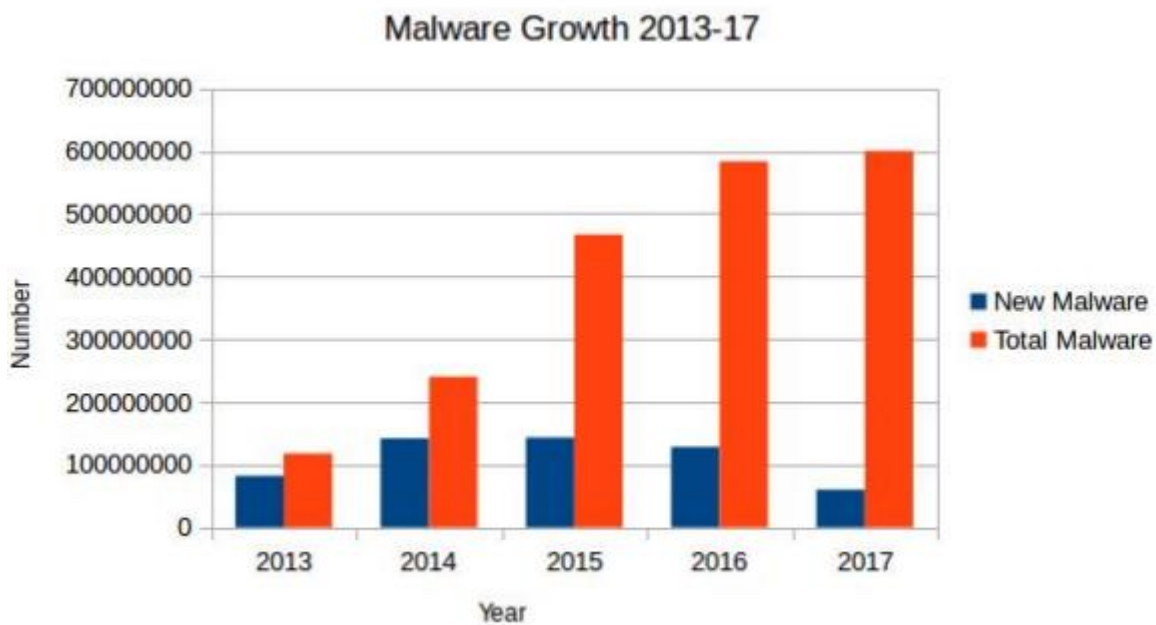


Figure 2.1: Total and new malware during the period 2013-17

2.1.1 What is Malware?

Malware [19] [20] are programs which have malicious intentions and which are intended to give access to a data framework without manager's authorization. Malware is a program that allows the attacker to use simple words to execute his malicious and criminal intent.

2.1.2 What is Cleanware?

Cleanware is the sort of malignant free programming. It is critical to isolate malware from clean products to ensure an obscure document isn't noxious. Models might be:

- E-mail accessories available.
- Connect your device with your external hard drive or USB stick.
- Software context sensing.
- A file to download.

2.1.3 Behavior-Based Detection

Researchers evaluate the malware and its actions during operational period when behavior-based malware is detected. The explanation that behavior-based identification is sometimes important is because you know the malware's true source code. The source code can then perform various techniques for obscuring code, e.g. packaging the data, polymorphism and metamorphism.

2.1.4 Malware Families

A family of malware may be treated as a set of malware that has similar features, whose main source code is identical, while only its behavior is modified. This is the motive why new virus is repeatedly mentioned to as updates or refreshments by industry and researchers of older malware. For the characterization, an emphasis is put on the first and same basic highlights and the behavior of deceptive examples, given their specific practices. More than 236 malware families were detected and registered in its database in 2014. Analysts need this registration and have many families and schools, because more examples mean that they have additional

highlights for improved consequences and can focus more upon presenting their estimates. There is a benefit that an analyst has a database that tracks these instances.

2.1.5 Naming Malware

As Microsoft[21] states, Malware with explicit drive able to have extra than one AV vendor name since they use diverse malware calling techniques and approaches and rely on the quantity of trials they obtain and the particular behaviour. The CARO Name Standard is one of the most well-known methods for naming malware checks. CARO is a casual identification program established by individuals and scientists in the AV industry. Notice that the CARO identification display does not tackle the problem of misleading class names, but seeks to overcome the contradictory marking. In order to prevent disorder among the states, it is intended to make a common norm or a linguistic framework for naming malware. AV distributors and clients.

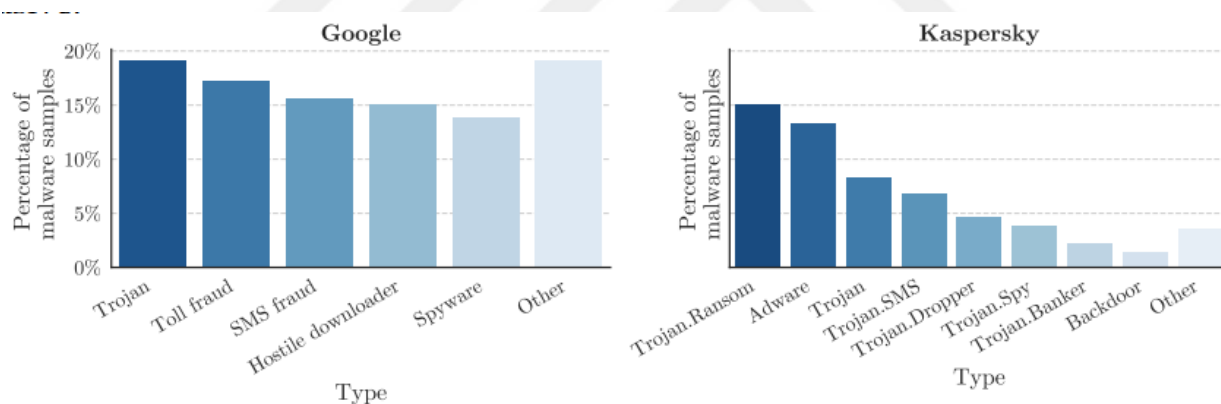
2.1.6 Types of Malware

Malware comes in a wide range of shapes and structures. Peng et al (2013) [22] ordered some principle kinds of malware concerning their conduct as appeared in Table 1.1. Their characterization depends on the accompanying components

- Existing form: Malware can be split into categories based on the software and individual categories. The former, regarded as a parasite, are simply fragments of programs which cannot exist irrespective of any particular function, utility or machine programmer. Independent malware is a program which can be programmed and run by the operating system. Some malware disguises themselves as useful and background malicious programs.
- Attack target: This categorization is based on the entity that the malware targets. It can be a simple file targeted by viruses to the network as such, by self-propagating programs. Certain malware bring extensive damage to the system as such.
- Human intervention: this is based on the need for a trigger to begin its operation by the malware. Some malware requires human involvement in some way.
- Major hazards: different malware causes varying levels of risk. Sensitive information can be used to effectively control a nation's critical infrastructure

- Speed spread: once malware is discharged, it is another measure for how rapidly it spreads. Some worms spread very quickly in contrast to certain forms of malware which don't spread.
- Method of detection: The nature of malware detection is varied. Many forms of malware reveal their existence through certain patterns and are therefore easier to detect. Many malware types are stubborn, as their detection activity is extremely difficult to identify.

The genuine use of different malware forms is difficult to estimate because antivirus (AV) sellers ' detection quantities ideally reflect their recognition strategies ' viability over real circulation. In any case, utilization of various sources assists with making an understanding of the natural Android malware program. Figure 2.2 shows the transmission of various malware shapes in Google's most recent security report[23] (left) and in Kaspersky's most recent security report [24] (right). Although Kaspersky uses an alternative phrase, the two figures indicate that the Trojan sort is the best-known malware. Notice that forms of malware are not necessarily



unrelated.

Figure 2.2: Malware type distribution according to Google [9] (left) and Kaspersky [10] (right)

A concise depiction of the different sorts of malware is introduced beneath.

2.1.6.1 Virus

A virus is a self-replicating piece of code which typically needs human intervention to spread through other programs. Self-replication is not always successful, but the virus can lead to changes. A virus is therefore not distinct from many other malware types since it can not be executed independently. The virus infects certain files that the virus prompts to run. The

consumer must therefore run the infected executable often to execute the virus. Often viruses spread via removable storage, emails and downloads or shared directories.

Table 2.1: Various types of malware (Source Peng et al 2013)[22]

Type	Virus	Worm	Trojan	Backdoor	Spyware	Rootkit	Botnet
Existing form	Parasitic	Independent entity	Disguised as other files	Disguised as other files	Disguised as other files	Disguised as other files	Disguised as other files
Propagation mode	Depend on the host file or media	Self replicate	Deceptive means	Deceptive means	Deceptive means	Deceptive means	Deceptive means
Attack target	Local file	Network host or network itself	System	System	System	System	System
Human intervention	Yes	No	Yes	Yes	Yes	Yes	Yes
Major risks	System damage, delete files, data loss	Network paralysis, data loss	Information leakage	Information leakage	Information leakage	Information leakage	Information leakage
Spreading speed	Fast	Very fast	Slow	No	Slow	Fast	Very fast
Detection method	Simple	Very complex	Complex	Complex	Complex	Complex	Complex

2.1.6.2 Worm

Worms are self-repeating bits of code that spread through systems and normally don't require human collaboration to engender. Worms are maybe answerable for the most serious harm brought about by various malware types, spreading wildly with the endeavors installed. Worms will typically abuse certain vulnerabilities in the frameworks they spread to, making them difficult to forestall and difficult to distinguish. Aside from 'great' Internet worms which abuse vulnerabilities in organize administrations, the best and across the board worms have been email worms. This class of worms regularly depends on clients opening connections to messages to clear something up. Duplicating with this fairly crude strategy, different adaptations of NetSky,

MyDoom and Bagle have been overwhelming the worm hit records for quite a while. New worm subsidiaries show up on the Internet consistently.

2.1.6.3 Trojan

Virgil's poem "The Aeneid" gives the word Trojan horse, and it serves almost the same role as in the poem in the digital world. It is a malicious thing disguised as a gift or a useful thing. Code of the Trojan horse is covered up in the hiding place as safe, harmless code. Microsoft and PandaLabs have found Trojans to be the most common type of malware.

2.1.6.4 Backdoor

A backdoor is a program that permits assailants to sidestep typical security controls on a framework, accessing the framework without substantial approval and perhaps without logging. The secondary passage is frequently introduced by an assailant subsequent to accessing a framework for future unapproved and simple access. Two of the most punctual models for windows based secondary passages are 'Back Orifice' and 'SubSeven', both created in the late 1990s, however fundamentally the same as projects have been grown from that point forward.. Albeit Back Orifice has authentic purposes, for example, remote organization, there are different components that make it appropriate for less benevolent business. The server can conceal itself from superficial looks by clients of the framework. As the server can be introduced without client connection, it tends to be circulated as payload of a Trojan pony.

2.1.6.5 Rootkit

Rootkits are tools that alter existing working framework programming with the goal that an aggressor can keep up access to as well as cover up on a machine. A case of a rootkit is change the 'dir'- order in Windows (list index substance) to shroud documents determined by the rootkit adequately from the client. The client would not see anything as the framework carries on as typical, yet in reality the aggressor has effectively concealed his vindictive program. Rootkits can be introduced in a few different ways and on a few levels. The establishment relies upon what bit of code of the working framework the rootkit alters, either portion code or client level code, and what sort of working framework and what procedures are feasible for that particular adaptation of the working framework. Two remarkable instances of rootkit based malware in Android are DroidDream and DroidKungFU.

2.1.6.6 Spyware and adware

These sorts of programming frequently do a stealthy and undesirable establishment on a PC. The projects can for instance show promotions or commandeer web perusing meetings to give advertisement locales instead of the needed destinations. As the name spyware proposes, the product can likewise follow client activities, catch individual data and so forth and disperse this to its proprietors. In any case, these two kinds contrast from different sorts of malware in that they are not as forceful. They for the most part require client communication to be introduced and don't spread without anyone else. They may fill in as preface to malware establishment on a framework, baiting the client to malevolent sites or to download pernicious records. The Gator eWallet program (Gator 2005) [25] is a normal case of spyware. The client would download and introduce a program to help them in 'recalling' passwords and structure passages on site pages. What the client probably won't have acknowledged is that the program is additionally following their web utilization propensities and sending information back to the Gator Company.

2.2 FEATURES USED WITH ML ALGORITHMS FOR MALWARE DETECTION

Highlight extraction is the way toward separating or ascertaining an incentive from each example of dataset. Feature extraction is domain dependent; for instance, Information Retrieval (IR) domain feature extraction is straight and simple whereas for recommendation system features set are structured and organized.

For malware research, 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. Features extracted by static analysis techniques are called as static features and features extracted by dynamic analysis are termed as dynamic features. Figure 2.1 shows different types of static, dynamic and integrated feature set used for malware detection.

Feature sets are classified into three groups: simple heuristic feature, static features and dynamic features (features extracted at runtime)(Dai et al., 2009) [26]. One of the studies considered detection techniques as structural fingerprints which are statistical in nature and positioned it as 'fuzzier' between static and dynamic heuristics (Komashinskiy et al)[27]. have presented classification of features in detail; they have divided whole features as external and internal.

Later the internal features are grouped as static and dynamic, under these two groups generic feature is also listed such as string, n-gram etc.(Komashinskiy and Kotenko, 2012)[27].

Yen et al. have presented the exploration of discriminatory features for ML-based malware classifier. They have experimented with various feature sets (byte-n-gram, opcode-n-gram, fields of PE header and dynamic trace) for malware families classification. They presented their result to show which algorithm works best with which feature set and how many features are optimal. With their study, they have suggested that Decision Tree (DT) performance is highest among all or equal to the performance of Support Vector Machine (SVM) for all features and provides highest accuracy with minimum features (Yan et al., 2013a)[28].

2.2.1 Static Features

As discussed earlier, on the basis of analysis type, features are mainly grouped into three classes i.e. static, dynamic and hybrid. This section lists and describes the comparison to related works of the different class of static characteristics. After running the executable file, static functions are extracted. Reverse technology is a common approach in which static data are derived from a binary. The two key methods for pre-processing and receiving static features are disassembly and hexadecimal dumping of binary format.

The Static highlights can be of two kinds: n-gram-based and heuristic-based. N-gram is an idea taken from Information Retrieval (IR), which alludes to the N back to back character with a move of size n from a book record. N-gram for malware can be byten-gram (n-successive byte arrangements taken from a hexdump1 yield) or opcode-based (n opcode grouping taken together from a dismantled paired). Ida-professional and Objdump are two regular apparatuses utilized for the dismantling. A scientific classification for arranging identification strategies for malware by AI techniques dependent on static highlights separated from executable is introduced by (Shabtai et al., 2009b)[29]. They have recorded and clarified five static highlights: Byte-n-grams, OpCode-n-grams, Function-based, PE highlight and Strings. In the accompanying segments every one of these static highlights alongside other static highlights are clarified.

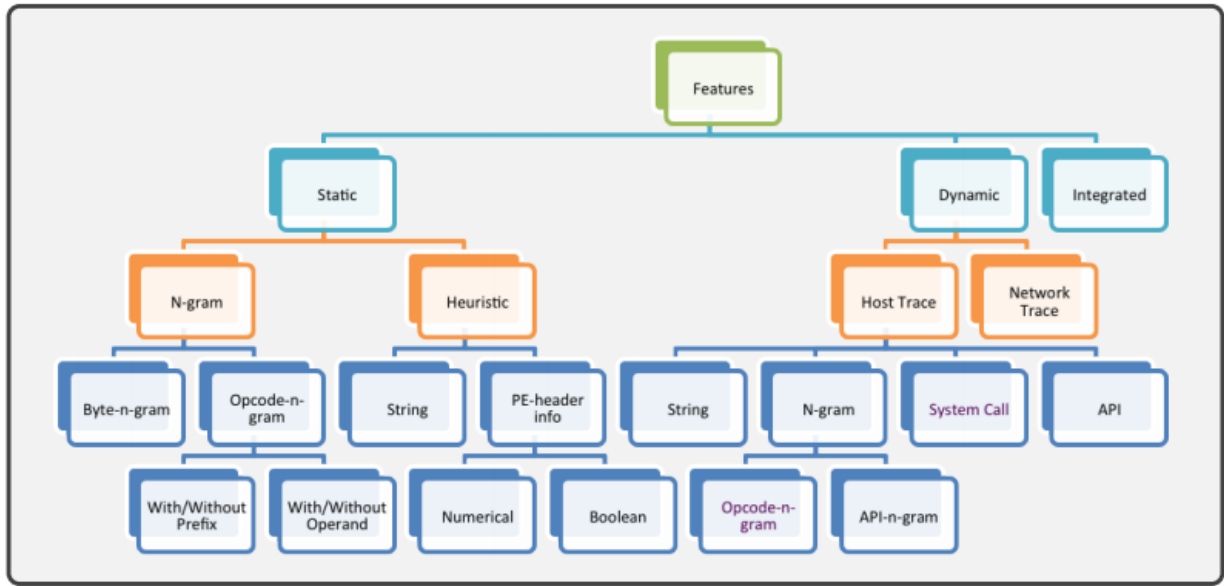


Figure 2.3: Types of features used for building machine learning based malware classifier

2.2.2 Dynamic Features

Dynamic characteristics are characteristics that are only derived by dynamic analysis. After executing a sample in controlled environment, dynamic features like function call, network use, device calling and hardware output counter are registered. Dynamic methodology reduces all constraints of static analysis. With static analysis packed, polymorphic (Aljawarneh et al., 2016) [30], metamorphic (Raphel and Vinod, 2016) [31] and encrypted or obfuscated sample can escape the analysis or make it difficult while dynamic analysis handle this because it works by executing malware's payload and recording sample behaviors during run time.

There are also dynamic process limitations, including speed, native run, experimental cleaning system and overhead system. A survey work has presented dynamic malware analysis techniques and provided a detailed discussion on such analysis tools (Egele et al., 2012) [32]. Dynamic features can be grouped into two major groups: host trace and network track-based features.

2.2.3 Hybrid Features

A hybrid design is a blend of the static and the dynamic elements. Elements for building hybrid feature sets, both static and dynamic, are omitted. By combining both static and dynamic features, an integrated feature was created. The combined characteristic set is found to be more accurate than individual characteristic sets (Santos et al., 2013)[33]. Experimental results were made. The implementation of static and dynamic malware classification functions (Awan and Saqib, 2016) has been suggested in a recent study[34]. Static features such as PE header information and printable strings, as well as API call logs (Awan and Saqib, 2016)[34] have been used in the analysis.

Comparing different features is a complex task. Based on the requirement, the selection of features can be done on the basis of various metrics. Feature extraction time, accuracy and FPR are the crucial metrics to make a choice to select a feature set. PE header features extraction is fast and require very less space in comparison to byte-ngram and opcode-n-gram This section has presented and explained about various features which are used with machine learning algorithms for building malware detector. This thesis work involves static analysis and use Portable Executable (PE) file and apk file for building static features & feature set for malware detection. So, in the following sections, features specific to Portable Executable (PE) file and Android's apk file are listed and discussed with references to related earlier works.

2.3 TYPE OF DETECTION

In terms of the approach to collecting functions, the biggest improvement in mobile malware detection methods is made [4]. Three methods are available to obtain characteristics: i) static and ii) dynamic. Fixed approaches aim to detect malware without running apps. In contrast, the program is run by dynamic methods and review takes place during runtime. A hybrid approach is a blend of static and dynamic analysis. The key downside to static analyzes is that these analyzes are vulnerable to obscure techniques that delete or restrict access to malware code. However, other methods, such as non-java code injections, network behavior and object changes, are only available during runtime. This limits their effectiveness in the face of null-day vulnerabilities [2]. Dynamic analytical methods can be used to overcome the limitations of static analysis, since these research applications during runtime. The dynamic research drawbacks from the way that

these methodologies are generally trailed by high bogus positive levels and are substantial on gadget assets [3]. There are additionally a few entanglements in the mind boggling investigation of virtual conditions, which is discussed more in the paragraph below. Due to the reduction in static analysis on zero-day attacks and the recent use of more Android malware samples to avoid static successful analysis [2], the research focuses on dynamic mobile malware analysis.

2.4 MALWARE DETECTION TYPE

Malware's rapid growth and development is a major challenge for the digital world. Many security solutions, i.e. to track and minimize losses from malware. Techniques 6 for anti-virus (AV) have been developed and new approaches explored. Such AV techniques are widely known as signing and non-signing techniques. AV software uses the professional scanning of signature based applications. It scans for signature the suspicious files (specific byte sequence). This is an extremely rapid method, which provides nearly 100 percent accuracy for well-known malware, but which completely fails to detect 'zero-day' malware (Bilge and Dumitras, 2012) (35). In addition, generating a signature is a time-controlled and complicated process that can provide the attacker with a wider attack time window⁹. Signature based technology is constrained by signature bases.

Using known tools and techniques will dramatically decrease the operation's likelihood of success for the attacker. So attacker must create a new tool for committing a successful cyber-crime and hence different anti-malware evading techniques (Christodorescu and Jha, 2004) [37] are being developed. Anti-malware process involves many tasks amongst which detection is the most important and complex one. Many malware detection techniques have been discussed in literature and many of them are being used by the anti-malware industry.

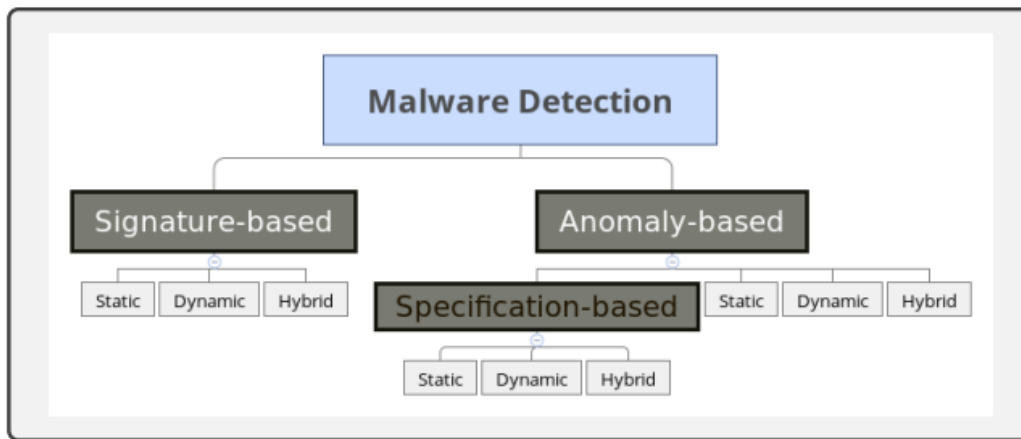


Figure 2.4: Types of Malware Detection Techniques

In past, researchers have grouped malware detection techniques in different ways, Signature-based and Anomaly-based are two main classes of malware detection techniques (Idika and Mathur, 2007) [38]. Anomaly-based detection is further branched into specification-based detection. On the basis of underlying analysis techniques, all the aforementioned detection techniques are grouped into static, dynamic and hybrid techniques. Figure 2.4 delineates the order of Malware discovery procedures (Idika and Mathur, 2007) [38]. Static identification methods depend on static examination which doesn't execute the example. Dynamic recognition strategies utilize dynamic investigation which executes the example in a controlled situation to extricate out various practices of test. Cross breed location methods utilize both static and dynamic investigation strategies.

The detection methods can also be defined by the recognition guiding principle.

2.4.1.1 Signature-based detection

This method of identification, which is otherwise referred to as abusive recognition, uses malware tags. For example, these marks can be double examples or pieces of programming code during static discovery. Such markings may be an indicator of actions with a particular identification. Registered malware is used to erase designs and form position marks. Such marks are used to identify malware at that level. This kind of recognition helps in particular known malware, which is less effective for multi-day attacks [40]. Figure 2.5 illustrates the cycle of the brand-based strategy of recognition. This figure describes a case of a signature pattern, which uses programming code pieces as markings.

The mark-based discovery model has a fundamental label database in Figure 2.5. This database provides malware marks. Right now, there are three pieces of pernicious code, three squares that have been altered by the name of the mark. Info is the complete code of a commodity in this identification model. This total code now contains some 10 bits of programming in various parts. These 10 bits contrast with the different characteristics in the database. When the model position model identifies 3 of 10 scraps as malignant for every mark from its model database. Signature 2 fits the code programming scraps in the model figure and the device is therefore recognized as malignant. The signature-based recognition technique poses two major issues. First, if the mark is already identified and in the mark database along these lines, any pernicious application must be recognized. In this way the detection of zero-days attacks is less convincing. Moreover, if the malware developers change their application slightly, they can modify the product code, modifying the application's mark [39], without any effort, the position technique can also be circumvented.

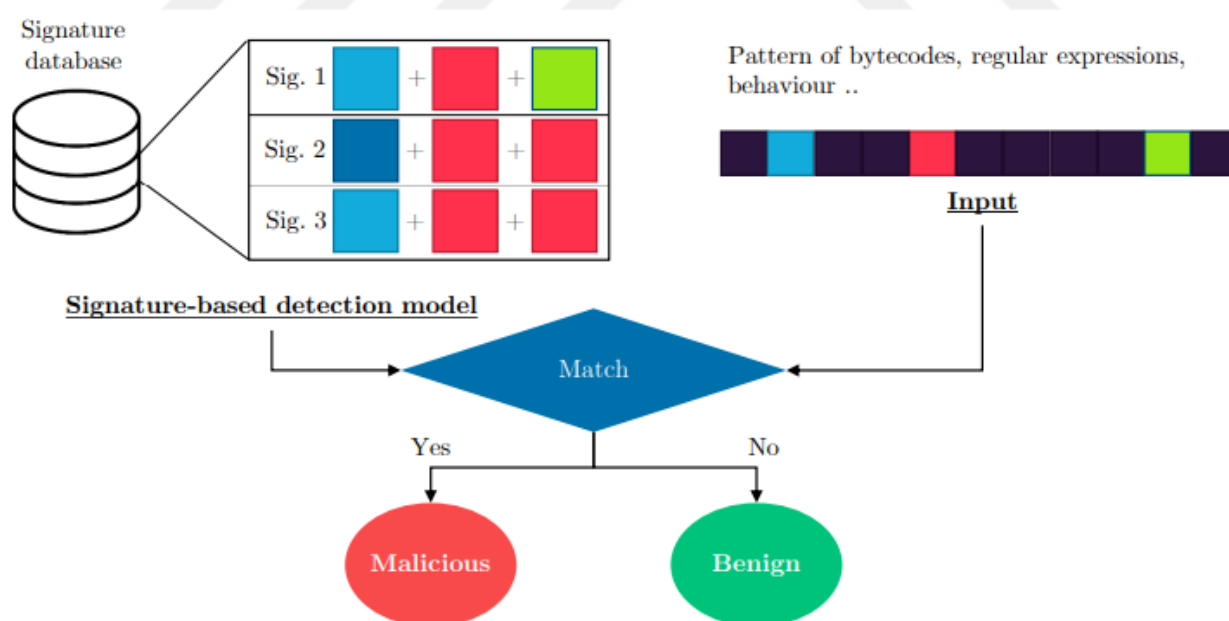


Figure 2.5: Signature-based detection method

2.4.1.2 Anomaly-based detection

This form is determined by anomalous and regular behaviour. The first is an action within the ordinary culture, which differs with normal behaviour. This detection form is sufficient for the

detection of null daily attacks but is also prone to false positives. This kind of identification may find unusual valid behaviour. Figure 2.6 demonstrates the method of anomaly detection.

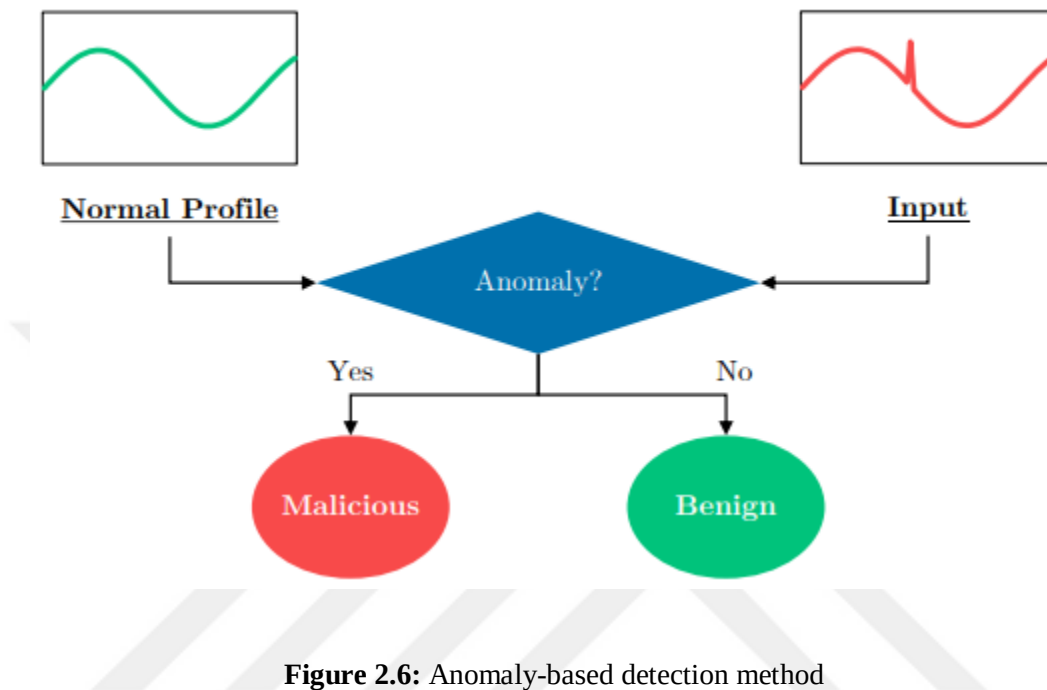


Figure 2.6: Anomaly-based detection method

The identification technique needs a normal behavior profile, Figure 2.6 shows. The detection method uses this profile to check if any information is close to this normal behaviour. The figure shows normal conduct as a function over time in a graph. For example, this graph will display the use of the CPU over time. The normal behavior in this case shows that use of CPUs decreases and increases over time. The feedback showing an increase in the use of the CPU is seen in the figure to the right. The input is marked as an abnormality if this spike is greater than some threshold.

2.4.1.3 Specification-based detection

This is a particular kind of detection method based on anomalies. It predefines permitted behaviors, which are a certain number of permissible rules. Any action that does not comply with these laws is considered malicious. One of the drawbacks is that the complete and accurate development of all the permissible rules is almost impossible [40]. Figure 2.7 shows the phase of the stated method of detection.

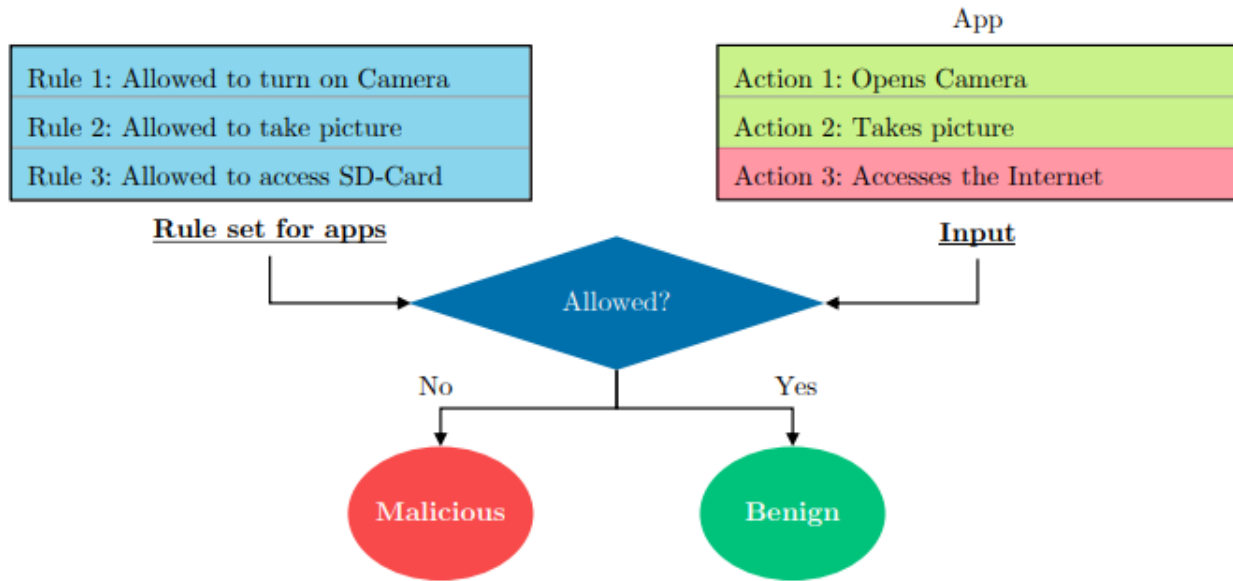


Figure 2.7: Specification-based detection method

A rule collection of 3 laws is employed as an example in Figure 2.10. All three laws are software approved interventions. Applications will activate the camera, take a photo, and access the SD card in this example. This can be a case in point for a simple camera use. The feedback is in action form. If the only rules described in Figure 2.10 are the three rules of the rules set, the input is flagged as malicious. This is because in this example, the first two acts are allowable but not the third.

2.5 NEED FOR MACHINE LEARNING

As previously mentioned, signature-based malware detectors can work well with previously recognized malware that some antivirus providers had already discovered. However, polymorphic malware cannot be identified, which can alter its signatures and new malware for which marks have not yet been created. On the other hand it is not always possible for the exactness of heuristically detectors to be measured sufficiently, resulting in many false positive and false refusals. (Ralescu and Basque Countries 2016)[41].

The high spread rate of polymorphic viruses dictates the need for the latest detection methods. One solution to the problem is the heuristic analysis combined with machine learning methods that boost detection efficiency.

A certain threshold must be used to assess the heuristics required for malware in order to depend on heuristics. We may, for example, recognise a number of suspicious features such as modifying a registry key or establishing a connection.' We would infer from this that any program that causes a minimum of five functions could be considered malicious. Despite the fact that this methodology gives some unwavering quality, it isn't constantly exact, as certain highlights may have a higher 'weight' than others. For instance, an 'adjusted pass ability' for the most part influences the framework more than a 'changed vault key. 'Moreover, a few mixes of highlights can be more suspicious than highlights essentially. 42] (Rieck, et al. 2011). Machine learning approaches may be used to take these associations into account and to identify them more accurately.

2.6 MACHINE LEARNING METHODS FOR CLASSIFICATION REVIEW

"Machine learning" is the dynamic computational strategy for programmed design acknowledgment and wise dynamic dependent on test information training [11]. "An increasingly broad idea of AI is the instrument by which the registering technique is utilized to apply learning calculations." [13]. This section, based on different books about machine learning [12][13], offers an outline of the fundamental theory of the various techniques used in this study. There are 3 types: supervised, unregulated and semi-controlled schooling. The objective is to make a regulated learning model which predicts Y on a specific X dependent on a lot of (xi, yi) models. In administered tutoring, Yi is known to be the xi name here. At the point when y is consistent, the issue is known as a relapse issue, and in the event that you are cautious, it is known as an arrangement issue. The research focuses on supervised learning to see if a malware-defined program that performs malicious actions is present. If the malicious program conducts malicious activity on the device and the prediction value y takes 1 and 0 if the malicious action on the device is not performed. In the next section, the classifiers used in this research are described.

2.6.1 Random Forest

Random Forest (RF) is a group classification that employ several decision-tree classifications in order to identify test instances. Figure 2.8 gives an example of a decision tree.

Its uncertainty is a significant drawback for decision-making bodies. Choice trees have an enormous difference and a little change in the information can regularly prompt a significant difference in the last tree. In Random forests, the variance of decision tree structures is reduced in test cases by several decision tree classification groups. Classification is then carried out by majority voting between all the decision-making bodies. Some of the benefits of Random Forest are i) overcoming overfitting ii) high-dimensional results. Benefits include i) precision based on number of trees ii) sensitivity to imbalanced information [3][2].

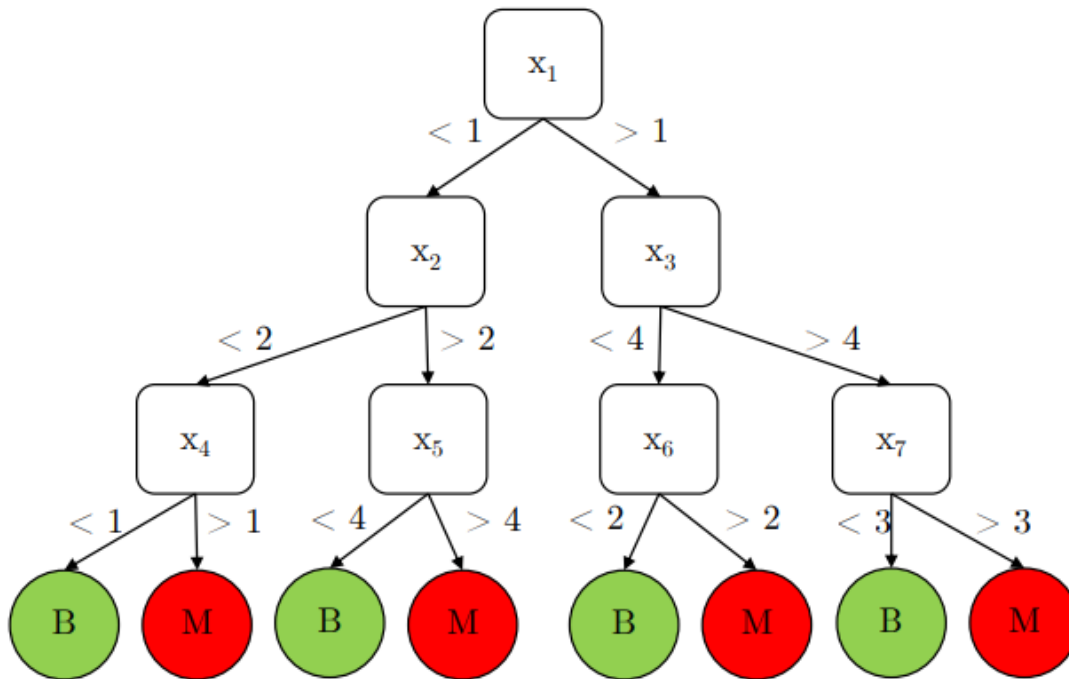


Figure 2.8 : Example of Decision Tree

2.6.2 Naïve Bayes

The statistic classifier Naïve Bayes (NB) is that predicts the likelihood of a given query instance of a particular class using the theorem of Bayes. In accordance avec the following formula, Bayes ' theorem, also called the law of Bayes, calculates the likelihood that a hypothesis H is valid.

$$P(H|e) = \frac{P(e|H)*P(H)}{P(e)} \quad (2.1)$$

Where

$P(H|e)$ denotes the posterior probability of H, conditioned on e

$P(e|H)$ denotes the posterior probability of e conditioned on H

$P(H)$ denotes the prior probability of H

$P(e)$ denotes the prior probability of e

The categorizer is called naive because the estimation of the above formula becomes less computationally expensive and assumes conditional independence. While Naïve Bayes is conditional independent, it does well in areas of violation of independence [14]. Naive Bayes ' advantages: I high speed ii) insignificant trait d

2.6.3 K-Nearest Neighbour

The K-nearest neighbor (KNN) is a distance classifier. Ranging from training data to invisible data, classification systems explore similarities between exercises. If the query instance Q is used, k training instances that are nearest q should be found in the classification. Then it categorizes the query case with majority voting among k neighbors. The cross examination occurrence will figure various measurements, for example, good ways from Euklid, good ways from Minkowski and good ways from Manhatten. Portrayal of neighboring gatherings is appeared in Figure 2.9.

KNN's focal points include [3]: I high exactness and accuracy (ii) non-straight evaluations (iii) no highlights are assumed. The disadvantages are I that it is inclined to unequal inspecting,.

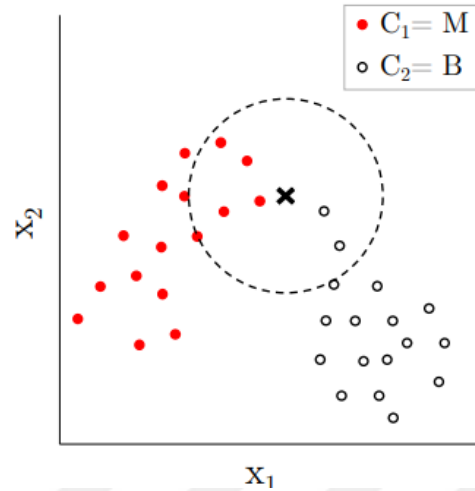


Figure 2.9: Example of K-Nearest Neighbour Classification

2.6.4 Artificial Neural Networks

ANN is a learning machine that uses a node system to identify test instances, i.e., artificial neurons. Such nodes are connected by guided ties with each other. An ANN contains a layer of data, several layers, and a layer of output. In the sample ANN displayed in Figure 2.10, each guided relation between neurons has some numerical weight. These numerical weights are used for each node's activation function. This capacity is utilized to assess the hub yield. Distinctive learning calculations can quantify the quantity of shrouded layers, the quantity of neurons and the loads between the neuron. This research uses a kind of ANN that understands with back-propagation, the Multilayer perceptron (MLP).

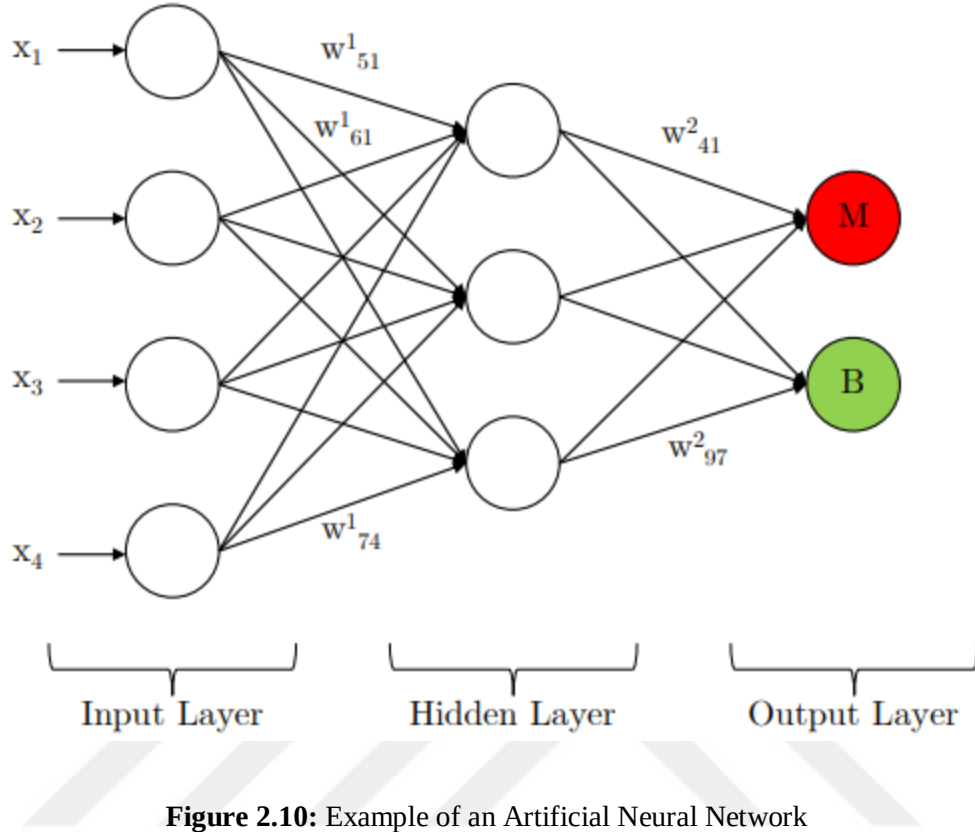


Figure 2.10: Example of an Artificial Neural Network

2.6.5 AdaBoost

Versatile fortifying (AdaBoost or Ada), like the Random Forest arrangement, is an outfit structure. AdaBoost utilizes multi-preparing emphases in dataset subsets to improve the exactness of the low learning classifier. A data subset is used to characterize the machine-learning. The weighting of all tasks and the incorrect recognition of a sample within the training sequence is improved, and the chances of the next iteration are higher. Furthermore, every sample labelled correctly is weighted less heavily. The cycle is continued until the estimator is complete. AdaBoost is known for its proper machine learning classification [43]. But the drawback of AdaBoost is selfish learning. AdaBoost with (standard) decision-making trees is used in this study.

2.6.6 Evaluation Classifiers

To test a classifier there are different performance metrics. The most important success measures are summed up in an uncertainty matrix. Table 2.2 shows the idea of a misunderstanding matrix.

Table 2.2: Confusion Matrix

		Predicted class	
		Malicious	Benign
Actual Class	Malicious	True Postive (<i>TP</i>)	False Negative (<i>FN</i>)
	Benign	False Positive (<i>FP</i>)	True Negative (<i>TN</i>)

The uncertainty matrix shows the correctly-classified malware instances (*TP*), the number of malware instances missing (*FP*), how many benign instances are correctly identified (*TN*) and the incorrect classification (*FN*) of benign classes.

Different measurements and their equation are appeared in Table 2.3. These measurements utilize the measurements appeared in Table 2.2. An every time calculation of the correct degree of estimation ($TP+TN$) of the all-out standards is the precision of a malware ($TP+TN + FP+FN$). The measurement has been used every now and then. This calculation, however, will probably not well reflect the classification presentation. High precision can be achieved in a slant data set, a dataset with a greater amount of a class than the other, by continuously predicting the lion's share class. For instance in a dataset comprising of 90% vindictive activities and 10% benevolent activities, continually anticipating noxious activities brings about a precision of 90%. The Precision (PPV) and Recall (TPR) measurements indicate an increasingly rational classification because of a slanted dataset. Accuracy and recall consonant mean is expressed in f1 (F-partage with $\alpha = 1$).

Metric	Formula
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
True Positive Rate (TPR)	$\frac{TP}{TP+FN}$
False Positive Rate (FPR)	$\frac{FP}{FP+TN}$
True Negative Rate (TNR)	$\frac{TN}{TN+FP}$
Precision (PPV)	$\frac{TP}{TP+FP}$
F-score (F-measure)	$(1 + \alpha^2) \left(\frac{PPV * TPR}{\alpha^2 (PPV + TPR)} \right)$

Figure 2.11: Performance Metrics

2.7 TAXONOMY OF MACHINE LEARNING TECHNIQUES FOR MALWARE ANALYSIS

This section explains the taxonomy used by the computer in examined papers for malware analysis. The surveyed works can clearly be represented in three key aspects. The first defines the final goal, e.g. the detection of malware. The second dimension defines the framework on which research is centered by dynamic analysis, for example. Registrations of Gpu. Finally, the third dimension points out what sort of computer education algorithm is used, such as supervised training, for analysis. Section 1 reveals a taxonomy scheme. Figure 1. The majority of this section is taxonomically arranged.

2.7.1 Malware Analysis Objectives

In general, malware analyzes require high detection capabilities to suit the information that is established from analyzing previous samples. However, the ultimate goal is special in finding the matches. For instance, a malware researcher may only want to see if new suspicious specimens are malicious or not, while another may be searching for the family they are likely affiliated with. This article outlines the research objectives of the papers analyzed, which are grouped into three main objectives:

Malware identification, malware likeness investigation and malware classification location.

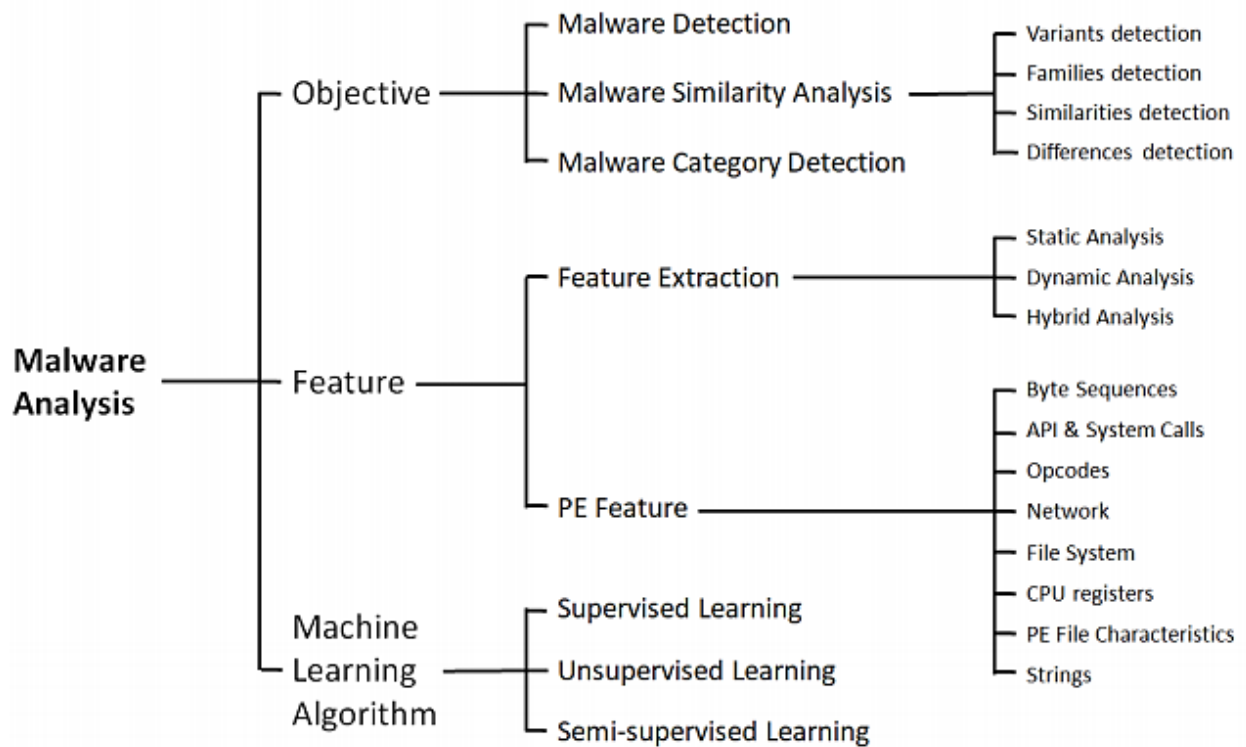


Figure 2.12: Taxonomy of machine learning techniques for malware analysis

2.7.1.1 Malware Detection

In malware detection, the most important purpose is to determine if a specific sample is malicious. This is also the main aim because it is possible to block a sample before it becomes hazardous, since a sample is dangerous in advance. In fact, this is the main aim of the majority of works reviewed [44, 45].

2.7.1.2 Malware similarity analysis

The important target is the discovery of malware similarities, such as the comparison between novel examples, established examples. We consider four different versions of this goal: discovery of variants, family recognition, and identification of sequences and position of contrasts.

Variants Detection. Making varieties is truly outstanding and most economical frameworks for an attacker to avoid acknowledgment parts, while reusing at any rate much as could reasonably be typical satisfactorily accessible codes and assets. Seeing that a model is really an assortment of a known malware thwarts such methodology to succeed, and plans to see how malware advance after some time through the improvement of new assortments. In like manner this objective has been significantly moved recorded as a hard copy, and a couple reviewed papers center around the revelation of varieties. Given a poisonous model m , varieties disclosure includes in looking over the open data base the models that are varieties of m [46]. Considering the huge number of malicious models got step by step from huge security firms, seeing varieties of certainly known malware is fundamental to reduce the remarkable job needing to be done for human specialists.

Families Detection. Agreed a poisonous model m , families disclosure involves in looking over the available data base the families that m likely has a spot with [47]. At the present time, is possible to relate cloud guides to known families and, by result, give an extra worth information to also examinations.

Similarities Detection. In order to evaluate with regard to previous observers, it may be of interest to recognize the stated similitudes and contrasts of the doubles. The identification of similarities involves finding out which sections and features of a sample are identical to anything already studied in the past. It enables us to focus on what is truly new and thus to eliminate the rest because further investigation is not necessary [48].

Differences Detection. Furthermore, define what differs from anything seen in previous findings. The discovery of new things that need to be further explored will actually lead to differences [49].

2.7.1.3 Malware category detection

Influential actions and intent can categorize malware. You may want to force and remove users of their sensitive information (for example, spyware), encrypt documents and request restitution (for example, ransom ware) and to remotely control the computer that is infected (e.g., toolkits for remote access). This is a gross but important way to describe malicious samples using these

categories [50]. While cyber security firms have still not accepted a standardized malware category taxonomy, it can actually add valuable data to the study by defining a sample category.

2.7.2 Malware Analysis Features

The characteristics of samples which are considered for analysis are discussed in this chapter. The executable extracts features in paragraph 2.7.2.1, while sub-paragraph 2.7.2.2 provides information about which specific features are taken into consideration.

2.7.2.1 Feature extraction

A static and dynamic analysis or a combination of the two methods of data extraction is conducted while the inspection and evaluation are achieved using computer teaching techniques. Static methods of study evaluate the sample content without needing its application, while dynamic experiments analyze its activities with samples. An assortment of methods might be utilized for complex malware location. Debuggers are utilized for level examination. Test systems model and view conduct like the malware condition, while emulators repeat a framework's activities with more prominent precision, yet need more assets. Sandboxes are virtualized environments for the detonation of malware, offering an isolated and secure environment. See the summary of these techniques by Ye et al. [52] in more detail. Traces of execution are typically used to derive functionalities when dynamic modeling is used. Execution traces can be generated using either sandboxes [42, 56] or emulators [53] in revised papers. Tools and techniques of software analysis can also be of use in the function extraction process, for example by providing disassembly code and control and data-flow charts. For correct byte sequences and Opcodes functionality, an accurate disassembly code is critical while control and data-flow diagrams can be used to extract the API and System Calls. Link to [54] for a detailed article on complex analyses. The majority of the works analyzed rely on dynamic analysis [48], while the remainder uses either static analyzes alone [11, 12] or a mixture of static and dynamic approaches in equal proportions [18]. The extraction method can be performed using static, dynamic or hybrid analyzes, depending on the particular characteristics.

2.7.3 Malware Analysis Algorithms

This section provides information about the use of machine-learning algorithms in surveyed works, whether they are supervised (section 2.7.3.1), unsupervised (section 2.7.3.2) and semi-supervised (section 2.7.3.3).

2.7.3.1 Supervised learning

Monitored learning is the activity of learning through the provision, during a preparatory process called preparation, of statistical modeling with correct instance examples. The supervised algorithms used are the rule-based classifier[55], the Bayes classifier[56], the Bayesian Network[55], Supporting Vector Machine (SVM)[12], the Bayesian Protector Classification [55], the Random Forest [59], the Gradient Boost Treatise [67], the k-Nearest Neighbor(k-NNN)[49], and the Artificial Neural Network [46], the Multilay Network. The following are the algorithms that have been controlled throughout the documents examined.

2.7.3.2 Unsupervised learning

Uncontrolled strategies don't depend on preparing and gain straightforwardly from obscure information. The reports utilized are: bunching of territory right of passage sensory [39]; the clustering separation and closeness measurements (either utilizing Euclidean[53] or hamstring distance[53]; cosine[53] or jacketeer similarity[62]; expansion of expectations[54]).

2.7.3.3 Semi-supervised learning

Half-monitored research incorporates both labeled and unlabelled information to improve expertise in the creation of mathematical models. Local and globally consistent thinking is used in [17] while the propagation of belief is used in [27].

2.8 MALWARE DETECTION RELATED WORK

Diverse AI counts (Perceptron, Mean Shift, DBSCAN, etc.) have been driven and made as for the area, recognizing verification and course of action of unidentified and unrecognized malware into known malware families. A touch of these tallies and techniques for thinking being used, are depicted right now this part. Regardless, Schultz, Eskin, Zadok, and Stolfo, 2001 [14], applied and expelled three engraving based attributes for malware game-plan: Byte groupings Portable

Executables (PE) and PC factors no uncertainty strings. The rundown of DLLs, work calls and two or three structure calls utilized inside each DLL utilized by the report and executable, are rotated arranged and fascinated from DLL records and information that are encased to Portable Executable documents. PC factors are managed and isolated from the executable records created by the substance PC factors that are encoded in program chronicles. All the blueprints of n bytes that are being changed and gotten from an executable record are named everything considered as byte movement.

Their presentation results were improved by Kolter and Maloof, 2006 in [15], utilizing information mining systems and n-gram as a segment to perceive malware. The estimations picked were Naïve Bayes, Support Vector Machines, Decision Trees with the last giving the best assembling outcomes. Essential for mechanization on malware game-plan was worried by Kong and Yan, 2013 in [16]. Creators proposed a system that relies on work call diagram of malware. Resulting to finding a superior than normal technique to expel the highlights subject to work call layout for each malware test, they utilized division estimations to locate the practically identical characteristics between two malware programs. These estimations packaged and mentioned the malware tests to same malware and class family while utilizing a constrained range kept the diverse get-togethers disengaged. Having endeavored that approach an arrangement of assortment estimations was used and prescribed that gains from pairwise malware parcels to plan malware into certain malware families.

Tian, Islam, Batten and Versteeg, 2010[17] focused on ordering the recurrence of Trojans for use over the course of their work. The byte measure which determines the capacity length is shown. The calculations presentation shows that the potential is important along with its recurrence in malware families ' detection and is linked to various attributes for the grouping of malware in terms of its execution. The WEKA library supplies malware structure with these calculations.

An alternative approach suggested by Santos[18] refers to the fact that the semi-regulated ID and multi-day misuse discovery theory employs a sensitive number of direct exercise records for malignant and kind-hearted instances. This technique uses a lot of directed and unassisted cases and analyzes and attempts to execute AI. The semi-administered Local and Global Cohesiveness Training (LLGC) is used to generate direct and solo knowledge and to provide a response to the fundamental engineering presented both by controlled and unassisted circumstances. A n-gram

strategy describes and characterizes executable. Also, specialists direct and survey the perfect measure of regulated circumstances and the impact of these circumstances with respect to exactness. Objective accomplished of this request and examination is to diminish the measure of vital managed cases while accomplishing noteworthy precision. The main downside is that managed preparing approaches were appeared and introduced better execution above or approach 90%.

Siddiqui, Wang, and Lee completed another fascinating review in 2009 in [19]. Their objective was to classify worms while looking at traffic bundles and to discover tests that are still not dissected and submitted to traders (or, in the wild, another energetic discovery area). Compilers and packers are differentiated and found before the examples are found. For characterization and group decrease, choices tree and random forest are used.

In [20] 2011 Zolkipli and Jantan needed to see malware courses of action from the perspective of malware execution investigation since they concurred that mind boggling work could upgrade exactness and productivity. That part will be performed on Anubis and CWSandbox, where malware movement is noticeable. Analyzers produce results which rely on dynamic analysis to use a fake astute and neural system. The malware is then stored in malware families. Fundamental inconvenience of the exploration is the present web traffic makes it difficult to utilize social investigation to accomplish the ideal outcomes.

The grouping was centered on appending and associating zero-day vulnerabilities to distinguished bunches. This, was actualized to show and present that bunching and arrangement, concentrated and dependent on conduct based examination can process the exercises of malware executables consistently

Anderson et al., in 2011 published a formula for malware discovery based on the study of diagrams that obey the dynamic guidance assortment in [22]. A shift in Ether-dependent malware research structure was used to accumulate tests. In order to express the likelihood of a chain diagram, the framework proposes controls for 2-gramm. Diagram portions are arranged to register and to represent a close-up vector in the learning process between instances. A Gaussian component which defines and specifies the nearby similarity between diagram edges and an unorthly section which records the foreign comparability between contours calculates a little

vector. a similarity can be identified and researched by two measurements Special simple malware activities agree on showing various component learning strategies. A disadvantage is the high computational complexity, which limits the use under genuine conditions

In [23] Bayer et al. proposed a technique that divides malignant datasets into groups. For the implementation of further data sources, a fraudulent capacity expansion for Anubis was carried out. However, a reflection of the documented trail has been performed using a familiar structure that helps to quantify the locality sensitive hazing (LSH). Experts demonstrate their strategy's adaptability by combining in several hours an enormous dataset of malware information.

Tian et al., 2010 in [24] used a mechanized device for separating API call arrangements from parallels while these are running in a virtual domain. Tian used and applied grouping techniques from WEKA programming to isolate malevolent information from great information yet in addition, for characterizing malware into their families.

In 2010 in [26] a malware structure technique based on maximum subgraph discovery was suggested by, Park et al.. Initially, a sandbox situation is used to run and interrupt malware tests while frame calls are made and from this system call platform a structured diagram is developed. The full normal category is calculated and evaluated for the similarity of the two projects. Nevertheless, other malware are definitely known, whose key skill is to obtain the root approval that bypasses the research strategy.

The malware detection by using Anubis is also discussed in another technique for the discovery suggested by Firdausi et al. in 2010[27]. AI is used to prepare data and records for insufficient arrangement vector models. Nari and Ghorbani in 2013[28] have built a programmed malware order model into their individual and organizational groups. Their technology depends on organize the following, which have been breached, generated and removed as pcap input records for the model. Sometime later, an outline of system malware exercises was given. Some highlights are obtained and used for grouping malware using order calculations.

Lee et al., 2007 developed another AI technique for malevolent grouping in [29]. Following the action of the dataset in a virtual situation where reports are exchanged, a social profile portrays the contact from the example with system properties. After identification of proximity between two profiles, community calculations are implemented as k-implies and the closest neighbor

bindings them accordingly. At the moment, the obstacle to uncertainty and slowing down implementation approaches have made half-and-half techniques key to better results being examined. The OPEM, the half-and-a-half zeroday finder that has accumulated and acquired attributes of adventure and manipulation of malicious code, has once again been built up in Santos, Devesa, Brecio, Nieves and Bringas in 2013. Called malware testing has static qualities, and dynamic malware tests are dynamic. By modified characterization calculations, two different data sets are analyzed. This technique improves all techniques ' precision and pace if they are run independently. Islam et al., reached a comparable achievement in 2013 [31]. In [32] 2011, Anderson et al. proposed a technique that uses various data and highlights. For the paired record, the dismantled document and two powerful submissions are proposed pieces based on Markov chain diagrams. For the control stream map, a diagram allows portion to be added and executed. For the text data data network, a Gaussian piece is executed. Diverse portion learning is used to find weighted correlations between the knowledge. However, bolster vector machine classification is used to organize and divide the dataset into negative and kind-hearted information. The findings have proven astounding. As it is written, information science suggests multiple answers to the malware order. AI is being implemented slowly in a variety of companies. Probably, information security would be one such thing, because the complexity and sophistication of the systems are growing ever further. Web and "cloud" apps generate enormous execution tracking information indexes and occassion logs that require adaptable and versatile procedures to distill useful and remarkable data.

Thinking about the expanding number of PCs, tablets and telephones everywhere throughout the word, clearly there is additionally constantly a more prominent number of malware introduced. It's therefore more than obvious for most PC organizations the malware related issue is fascinating.. Since we research man-made reasoning techniques for malware order we center on past work dependent on comparable thoughts right now. Know that every one of following methodologies is utilizing totally unique dataset to test calculation's exactness and in this manner looking at exact outcomes legitimately is unimaginable. It is additionally conceivable that enormous organizations creating antivirus programming have some progressively advance procedures in their exploration gatherings, anyway the vast majority of them don't distribute any accessible papers about it for clear reasons.

Microsoft Research group in their paper [17] shows execution of a reaction for relative issue utilizing neural structures. What has any kind of effect is that they are basically making malware identifier rather than classifier (trying to see whether a given record is malware or a made sure about equivalent chronicle), in a little while they disregarding everything have the malware named into for all intents and purposes indistinguishable classes as we do and give this data to the learning tally. As in every equal strategy they are going toward the issue of a huge amount of highlights. To administer it they at first utilize some conventional part affirmation in any case in the second step they utilize optional projections to chop down the dimensionality broadly more. This technique on a fundamental level takes the cross section of highlights and duplicates it by abstractly picked projection structure which endeavors consolidate vectors to other space with lower estimation. This procedure is for all intents and purposes indistinguishable from one secured layer of a neural structure which isn't found by any figuring yet allotted self-confidently. The paper shows that this framework prompts tremendous theory resources in learning time comparatively as assessment time. No ifs, ands or buts, even while lessening the information space by a factor of 45 they had the decision to set up a social occasion check with exactness over 90%. Another charming outcome is that adding more layers to their neural structure doesn't improve execution in any capacity whatsoever, paying little heed to the way that they have more than more than 2,000,000 train tests. The basic highlights are work calls. To pick them they are utilizing a sharp lightweight Windows emulator which records the movement of the program as it is running on the PC. This way of thinking looks like our call chart approach portrayed in divide Call layout.

Cesare and Xiang[18] face the issue of the matched weight and encryption of reports. As the malware engineers look to separate malware locators, they utilize some weight or encryption stunts to do as such. The question is handled over two cycles. Second, you will find out if the code was compacted. To do this, they are suggesting a shrewd entropy technique. Compacted documents convey a similar measure of data, however are littler, thus the packed records have a lot higher entropy than the first records. Another significant perception is that previously packed documents are a lot harder to pack more by some other calculation in light of the fact that the entropy can't be raised any more. So the compacted documents can be likewise found by attempting to pack and checking the achievement of pressure. Accomplishment of pressure can be estimated as pressure proportion, the pressure proportion can be accordingly considered as

phenomenal element for classifiers. Third, the data from the paper must be decompressed. This is why a virtual computer, which recreates the program running from a standard PC, is executed again. As this computer reconstruction functions, it also offers highlights for the detection of malware. They take feedback and power calls into consideration.

Tian, Batten and Versteeg suggested another interesting way of understanding the business [19]. The most important issues (part of highlights) of most methodologies are addressed by a completely different set of capabilities. The most significant highlights used in the paper are the capability lengths in code. We don't use specific lengths of space, but only cans with intermediate lengths. They use m pails and exponentially grow their sizes. For some clear k, the interim I cover capacities of $k I - k i + 1$. For $m=30$, they may correctly prepare a measured recurrence [20] classifier of 90%. The greatest leeway is the speed at which such a classifier can be taught.

An article in the Call Graph Clustering Malware Classification by Joris Kinable and Orestis Kostakis[21] talks about the likelihood of call diagrams gathering malware. The thought is that, regardless of whether the code is adjusted or adopted by various settings, the core structure becomes extremely comparative and perseveres in this activity. The problem is that it is difficult and repetitive to identify the distinction between diagrams. In Part 4.2, our approach for analyzing diagrams is used to check this technique.

Hu, Bhatkar, Griffin and Shin[22] appeared totally in a different perspective for grouping. In the building agent source code for the managed software all highlights are looked at. They offer every imaginable guide specific numbers and group numbers in the code. We can't care less about guidance parameters as they can be modified by malware development muddle or just another compilation setup.. They just recognize operand types (over-burdening of the guidance), so move guidance has another code in the event that it is move from library to memory and on the off chance that it is move from register to enroll [23] [24]. Creators had the option to arrive at average outcomes with 4-grams. Attempting to expand the n did just broaden the hour of the computation, anyway it didn't improve the precision essentially.

The disadvantage is that it is beyond the realm of imagination to expect to utilize an antivirus include sampler to get definite information about procedure conduct. So the conceivable measure

of highlights is constrained. The other interesting thing about the article is that they use self-assertive woodlands for plan. They attempted various things with various tree check, number of features per tree and tree significance. The best results were practiced with 160 trees and 16 features and significance of 16. Considering preliminaries they referenced the going with target realities: More trees are better, significance of tree ought to be in any occasion 16 and lower proportion of features per tree is better. They used sporadic woods execution from Weka [35][34].

2.9 CONCLUSION

This chapter discusses latest works on the identification of malware using static algorithms for machine learning. Research is structured and described in different sections on the identification of malicious Portable Executable (PE) and Android application. In this analysis, the limitations and reach of developments in present methods and techniques are understandable. The whole investigation centers on the current works which have utilized static highlights with AI to recognize malware.

3. ANALYSIS AND DETECTION OF MALWARE BEHAVIOR USING MACHINE LEARNING FOR COMPUTER DEVICE

Malware can be regarded as any device, server or network damaging software that is built deliberately [1]. Malwares in early days have not been encrypted with sophisticated chip algorithms, so that a certain piece of code can be easily detected and marked. One of the main reasons why malware sample is created in a high volume is the extensive use of malware developers ' confusion techniques which constantly modifies and/or obscures malicious files in the same family of malware (i.e. similar code and common origin). A generalized malware analysis based on the Machine Learning is therefore considered an efficient approach, so that it can be accurate on invisible samples. Nonetheless, static [4] and dynamic [5] methods are used for the identification and classification of malware during preparation. The code (assembly or machine) of the malware is normally tested without static methods [6]. While malware activity is monitored via dynamic methods during its execution [7]. All analyzes have their own drawbacks. For example, the flaw in the code cannot be identified at the precise location in a dynamic analysis, although static techniques do this very well. The advantage of static analysis is, on the other hand, that malware can be identified before it is carried out. In dynamic methods the power of contaminated systems is restored, which in static analysis is not possible. The classification of malware is important in malware research, because it is important to identify different types of malware in order to be sure that they can contaminate personal computers that they pose danger and how they are covered. In the event of identification of a malware, a malware classification system is assigned to the most suitable family. Several methods are available to detect malware in the wild, but it is still a difficult task to detect a null day malware.

3.1 OVERVIEW

The thesis suggests that three separate machine learning algorithms be functionally and reliably used in detecting if an executable is infested or clean. The project's algorithms and their advantages will be discussed. The use of antivirus and anti-malware programs and malicious programmers, such as Polymorphic Malware uses auto-learning algorithms to encrypt each times as they reinvest the table and become more and more difficult to detection, is common in the field of master learning. Using the programming language Python, the implementation of these

algorithms was done with scientific knowledge. Scikit-learn is a free software package for Python's machine analysis libraries. It is designed to work with NumPY and SciPi digital and scientific libraries and is composed of various retrieval and grouping algorithms.

The research was intended to determine the best methods to extract, view and classify features best implemented in the Sandbox Cuckoo. K-Nearest-Nearest-Neighbors, Decision Trees, Vector Support Machines, Naive Baye and random forest classifications were evaluated in particular. This study composed 1156 malware file sets and 984 benign files in many formats in 9 different types of families in this study. This research offers suggested methods to identify and assess malware based on machine learning and its implementation recommendations. The study conducted may also be useful as a framework for further work using master learning methods in the field of malware analysis.

3.2 MODELLING

The modeling stage of this research is described in this section. The dataset will be used to train and check malware detection models during this phase. The activities and performance in the modeling process are shown in Figure 3.1. The following sections describe the operations. Section 5.1 explains the various methods for machine learning..

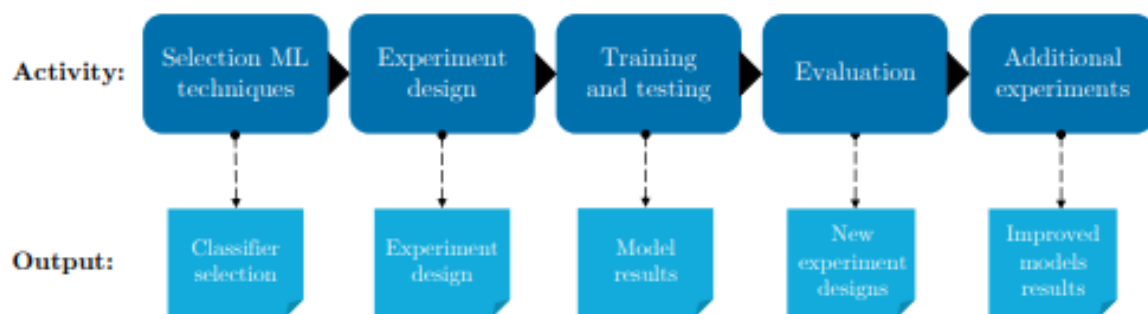


Figure 3.1: Modelling activities and output

3.2.1 Selection Machine Learning Techniques

The most promising methods are selected based on the previous complex malware detection studies listed in Chapter 2. The maximum results with current malware found (TPR) and the low number of flags (FPR and FNR) showed the Random Tree, Naïve-Bayes and K-Nearest Neighbor categorizations. There have also been promising findings in mobile malware development on neural networks recently researched. The study uses classifiers for the detection models of the Random Forest, Naive Bayes, K-Nearest Neighbor and Neuronal Network.

3.3 AIM OF THE PROJECT

This project is mainly aimed at detecting malware or botnet traffic through different Machine Learning approaches from the NetFlow dataset. In particular, this method is aimed at:

- Detecting Netflow data malware or botnet traffic. Any Netflow dataset of any size should be taken by the system, clean or malware, and marked as usual or attacking data.

- Compare a number of Machine Learning approaches for specific applications and suggest the best one.

3.3.1 Methodology

We follow the approach listed below to achieve the above objectives.

- Selecting a Dataset

Traffic flow data is the first part of the methodology. We can do this by choosing a known organization's actual data traffic and removing NetFlows. In the absence of real data traffic, another way of using a collection of public domain data. For this function, CTU-13, KDDCUP993 and CIC-IDS-20174 (University CTU 2011), are well known data sets. In comparison to other public datasets, we have chosen CTU-13 because it's widely available and used in several similar studies. See Section 4.1 for information on the dataset. We then evaluate the statistical data set to distinguish common characteristics and frequencies. The raw data attributes of the netflows StartTime and TotPkts, TotBytes, SrcBytes and Name, srcAddr, Sports and Dir. The flow describes this category of properties.

- Feature Extraction

We then define and extract the features after we have selected a data set. This move is a crucial share of the process. NetFlow data includes categorical features which must be represented in numeric or Boolean values, resulting in too large a matrix and causing memory problems. We use a schema to analyze NetFlow circulation using a time span to reduce the amount of data to be processed. We chose a 2-minute time window with a 1-minute move. The bi-directional NetFlow dataset is preprocessed, and the category and numerical features defining the data set are extracted within the given time frames.

- Feature Selection

You have to pick this process from the apps removed. Usage of feature choice technologies to decrease the size of the input training matrix. The different technics used for this reason are filters by Pearson's Correlation, Backward Feature Elimination Wrapping processes, Random Forestry Classification embedded methods, Principal Component Analysis (PCA) and Stochastic Neighbor Embedding t-distributor (t-SNE).

- Comparison of Algorithms

We needed to test five calculations. The accompanying AI models are: calculated relapse, vector machine support (SVM), ranger service grouping, venturing up inclination, and profound neural system.

- Botnet Detection

The last advance in our approach is to check our model to check whether botnet traffic from the CTU-13 informational index can be effectively recognized. In the f1 scores of the recorded models, the general execution of the botnet recognition is determined.

3.3.2 Main Issue with Network Security Data

Laboring with safety data in the Network poses several challenges. Firstly, the data are very unfair because the bulk and only a small part of the traffic is harmless. This makes it impossible

for the designer to understand negative things. Therefore, the probability of overpowerment during training is high because the network's structure influences how you learn while you want a network algorithm. A dynamic network is also covered by traffic analysis: time-dependent communication and server-based links that appear and disappear in the network with new requests and users. This is why it is a real problem in network security to detect new unknown botnets. An accurate data analyzes are needed to meet all these challenges and mechanisms are necessary to avoid overfitting, such as cross-validation.

3.4 DATA ANALYSIS

3.4.1 CTU-13 Dataset

CTU-13 is an online dataset that is utilized for creating calculations identified with botnet detecting, for instance in Garcia et al. 2014 and Rafal, Choras and Keller 2019. It was created by CTU University 2011 to build up certified botnet traffic blended in with ordinary traffic and foundation traffic. The informational index for extraction and preparing of our models will be depicted right now.

The CTU-13 Dataset is made of 13 catches of various botnet tests outlined in 13 bidirectional NetFlow5 documents (see Figure 3.2).

Table 2 – Characteristics of the botnet scenarios. (CF: ClickFraud, PS: Port Scan, FF: FastFlux, US: Compiled and controlled by us.)										
Id	IRC	SPAM	CF	PS	DDoS	FF	P2P	US	HTTP	Note
1	✓	✓	✓							
2	✓	✓	✓							
3	✓			✓				✓		
4	✓				✓			✓		
5		✓		✓					✓	UDP and ICMP DDoS.
6				✓						Scan web proxies.
7									✓	Proprietary C&C. RDP.
8				✓						Chinese hosts.
9	✓	✓	✓	✓						Proprietary C&C. Net-BIOS, STUN.
10	✓				✓			✓		UDP DDoS.
11	✓				✓			✓		ICMP DDoS.
12							✓			Synchronization.
13		✓		✓					✓	Captcha. Web mail.

Figure 3.2: Scenarios of the CTU-13 Dataset (CTU University 2011)

3.5 MACHINE LEARNING METHODS

This part provides a theoretic context on machine learning approaches necessary to understand the practical application. First, it discusses the machine education field and describes the methods that are essential for this analysis. These approaches include adjacent areas, decision-making trees, natural forests, support for vectors and naive bays.

3.5.1 Machine Learning Basics

The fast advancement of data mining technology and techniques have helped to make machine learning a distinct computer science area. The key idea is that a system should rely on its own activities (computer programme, algorithm, etc.). It is known to be a branch of the field of artificial intelligence. Second, in 1959, it is called the "science area, which enables computers to learn without being expressly scheduled." "Information E is said to have acquired from a computer program in some task classes and P performance evaluation where its progress in T, as calculated in P, increases with E."

-machine's main idea is to train the model to perform a certain task based on some algorithms: classification, clustering, regression and so on. The training is done by input and the built-in model is used for prediction subsequently. The outcomes of these models are determined by the initial task and the implementation. Data on house attributes, such as room number, size and quality, are available for possible application; estimate the price of the previously unknown house; identify a pool of new pictures based on two safe medical pictures and the pictures with tumour; group animals pictures to clusters from an unsorted pool.

The general workflow of the machine learning method as outlined in Figure 3.3 is worth taking into account to gain a deeper understanding.

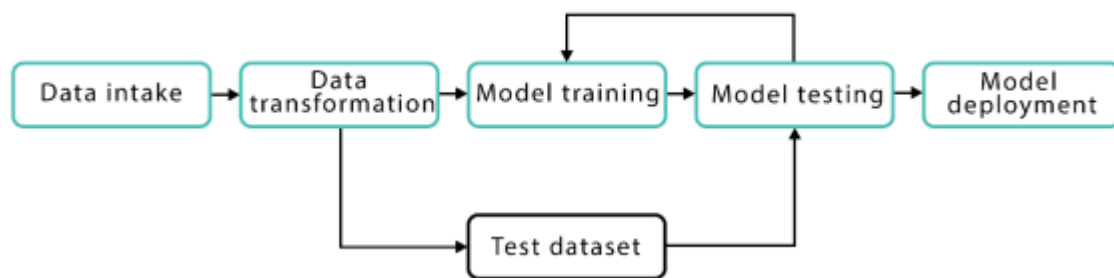


Figure 3.3: General workflow process

The cycle consists, as can be seen, of five phases:

1. Receipt of data. The first move is to load and save the data set from the register.
2. Software transformation. Transformation. The data loaded in step 1 is transformed to match the algorithm, cleared and normalized at this stage. Data is converted to the same set, format, etc. Data is converted. Extraction and selection features are also being done at this stage, which are further discussed.
4. Test template. Test template. The model assembled or prepared in Phase 3 will be tried by methods for the test informational index and the outcomes created will be utilized to manufacture another model which would consider past models, for example to know.
5. Project Implementation. Model implementation. The best model is now chosen (be it after the defined iteration number or once the required result is obtained).

3.5.1.1 Feature extraction

The input data attributes should be extracted in one of these cases to fill the algorithm. For example, data may be represented as a multidimensional matrix for the domestic price where each column has an attribute and the numerical values are shown in the lines. Data can be seen as a RGB value for every single pixel in the picture frame.

Such attributes are known as properties, and the matrix is known as a vector. Functional extraction is the method of extraction of data from the files. A collection of detailed and non-redundant data is used to extract the function. The fact that the apps are important and relevant details about our data set is crucial because without it we can't predict accurately. This is why extraction of functions often does not require much testing and analysis. It is a non-obvious activity. This is rather unique to the domain, and general approaches are poorly implemented in this regard.

Non-redundancy is another essential prerequisite for a good collection of features. The algorithm can be biased and, thus, provide an incorrect result, with redundant features such as information features and redundant attributes which are closely dependent on one another.

If the data is too large, then a decreased vector with a reduced number of properties can be converted. Which can be applied to the algorithm (having too many features). The method by which the vectors are reduced is called the selection of the function. We assume that the chosen characteristics would characterize the details from the first collection at the end of this process in order to be used instead of the initial data without a loss of precision.

Additional transformations are possible:

1. Standardization An example of standardization can be divided by the total number of counts to represent the distribution and remove the dependency from the image size, with the x_i is number of pixels color i . This is the formulation: $x' = x / ||x||$.

2. Institutionalization In certain circumstances, highlights may have various scales in any event, when they identify with practically identical articles. Keep in mind, for instance, the expenses of lodging. Here, the room size element is a number, maybe not more than 5 and the house size element is estimated by square meters. In spite of the fact that it is conceivable to analyze, include, subtract and so on all qualities, before standardization the outcome will be nonsensical. The following move is as often as possible made: $x'_i = (x_i - \mu_i) / \sigma_i$, where μ_i and σ_i are the mean and the standard deviation of highlight x_i over preparing models.

3. Non-linear expansions: While we need to diminish the size of information much of the time, it might be imperative to expand it now and again. This can be valuable for complex issues, where collaborations from the outset request are insufficient to get exact outcomes.

3.5.1.2 Supervised and unsupervised learning

So far we've spoken about the principles of machine learning from the viewpoint where we have initial data that can train the algorithm. That is not always the case, however. The two machine methods – guided and unregulated learning – will be implemented here.

Learning is based on labeled data in supervised learning. Here, we have an initial dataset, which maps data samples to the right result. The case of house prices is an example of controlled education: there is an initial dataset here with homes, their characteristics and their costs. This dataset provides the model with the right results. The model is educated. Regression and classification problems are examples of supervised learning:

1. Regression The value, i.e. sample values from the training system, is predicted by previous observations. Generally we can claim that it is a regression problem if the output is a real / continuous number.

2. Classifications We would like to determine the class of the previously unknown specimen on the basis of a collection of labels of data in which the sample is graded. There is a small and limited range of potential outputs. Generally, if the output is a discrete/categorical variable, we can conclude that it's a classification problem.

There is no initial marking of data in unsupervised learning, as opposed to supervised learning. The goal here is to find a trend in the unsorted data set rather than forecast a value. Clustering is a growing subclass of unsupervised learning:

3. Grouping Find the shrouded designs and separate in bunches of similarity in plain outcomes. A model can be the revelation of the shop's client base.

3.5.2 Classification Used Methods

Malware detection can be viewed from the educational point of view as a classification or cluster problem: unknown malware types should be clustered based on certain features contained in the algorithm. In examination, via preparing a model on the huge dataset for malignant or considerate information, we can lessen this issue to grouping. This issue must be utilized for characterization of known malware classifications—the right class could be discovered all the more rapidly and the outcome will be more exact than with bunching calculations with a little gathering of classes with a malware test. The hypothetical foundation is given right now all the methodologies utilized right now.

3.5.2.1 K-nearest neighbours

While one of the simple machine learning algorithms, K-Nearest Neighbors (KNN). KNN is a parameter-free method that does not presume the information structure. Results in real world problems never follow general theoretical concepts and thus make non-parametric algorithms a successful option. The KNN model is as simple as the dataset to implement—no planning, the whole training system is saved.

All order and relapse questions can be utilized for KNN. The gauge depends on k preparing examples in the two circumstances nearest to the information occurrence. In KNN, yield is the class that the information occurrence is a piece of and that will be casted a ballot by the dominant part for the k nearest neighbor. The yield is the property estimation in the issue of relapse which is commonly the mean estimation of the closest k -neighbors. The schematic proof is recorded in Figure 2.

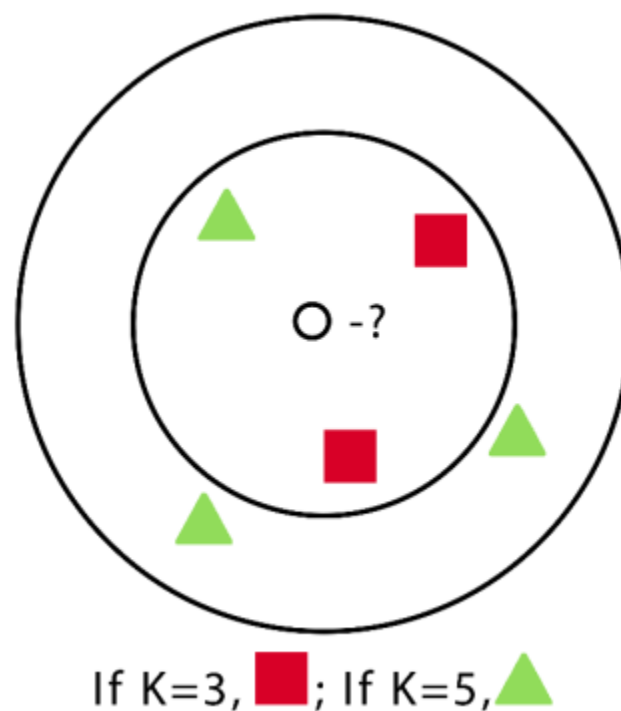


Figure 3.4: KNN example

For the location of nearest neighbors, various distance measuring methods are used. Hamming distance, Manhattan distance, Minkowski distance are the most common ones.

Hamming Distance: $d_{ij} = \sum_{k=1}^p |x_{ik} - x_{jk}|$

Manhattan Distance: $d_1(p, q) = \|p - q\|_1 = \sum_{i=1}^n |p_i - q_i|$

Minkowski Distance $= (\sum_{i=1}^n |x_i - y_i|^p)^{1/p}$

Typically the Euclidean distance is the most used method of continuous variables, described by following formulas:

EuclidianDistance $= \sqrt{\sum_{i=1}^n (q_i - p_i)^2}$; p and q are the points in n - space

Euclidean distance is ideal for problems where the features are the same. For example, Manhattan Distance should be used for characteristics of various types.

In the predictability of the algorithm, k plays a decisive role. However, it is not easy to choose the k value. Less k values will very likely lead to less accuracy, in particular in data sets with lots of noise, as each case with training set has a higher weight while taking decisions. Larger k values reduce the efficiency of the algorithm. Therefore, if the value is too small, the model can be exploited to make the class limits less distinct and again to minimize precision. In general, k should be selected using the following formula:

$$k = \sqrt{n}$$

In case of problems in classification of even races, it is preferable to choose a disability k because it avoids a tie in the majority method of voting.

The limitation is that the data sets are not distributed equally. Therefore, if one class significantly exceeds the other, more neighbors will be involved because of its large number. (Laaksonen and oja 1996). 1996. 1995. 1996.

3.5.2.2 Support vector machines

Another algorithm used for classification problems typically is vector machinery support (SVM). The key idea is to find a hyper-aircraft to help differentiate between stages. "Advantages" applies

to the closest hyperplane lines, which, if removed, will change the direction of the hyperplane. The distance from the vector of support to the hyper-plane is called a margin.

.We understand intuitively that the more our classes lie in the hyper plane, the more accurately we can predict. Therefore the SVM algorithm aims at finding the highest margins in such a hyperplane, though multiple hyperplanes per question can.

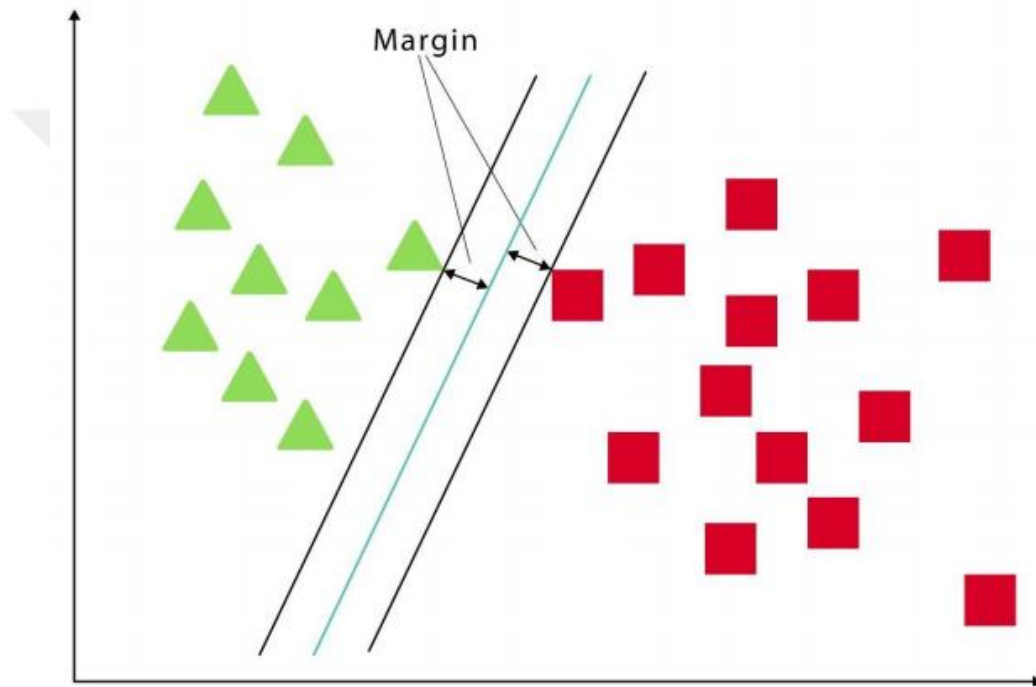


Figure 3.5: SVM scheme

A dataset of two groups is available in Figure 3. The problem lies thus in a space of two dimensions, and an hyperplane is a line. Generally speaking, hyperplanes can be as big as we need.

The calculation can be portrayed as follows:

1. We define X and Y as the input and output sets respectively. $(x_1, y_1), \dots, (x_m, y_m)$ is the training set.

2. Given x , we want to be able to predict y . We can refer to this problem as to learning the classifier $y=f(x, a)$, where a is the parameter of the classification function.

3. $F(x, a)$ can be learned by minimizing the training error of the function that learns on training data. Here, L is the loss function, and R_{emp} is referred to as empirical risk.

$$R_{emp}(a) = \frac{1}{m} \sum_{i=1}^m l(f(x_i, a), y_i) = \text{Training Error}$$

4. We are aiming at minimizing the overall risk, too. Here, $P(x, y)$ is the joint distribution function of x and y .

$$R(a) = \int l(f(x, a), y) dP(x, y) = \text{Test Error}$$

5. We want to minimize the Training Error + Complexity term. So, we choose the set of hyperplanes, so $f(x) = (w \cdot x) + b$:

$$\frac{1}{m} \sum_{i=1}^m l(w \cdot x_i + b, y_i) + \|w\|^2 \quad \text{subject to} \quad \min_i |w \cdot x_i| = 1$$

In general, SVMs can achieve good precise performance, in particular in "clean" data sets. In addition, it is good to work with high-dimensional datasets even when the number of samples is small. However, it can be more efficient for large data sets with a great deal of noise or overlap. Training time may also be high for larger datasets. (Jing Zhang and 2010). Zhang. [6]

3.5.2.3 Naive bayes

Naive Bayes is the Bayes Theorem classification machine analysis algorithm. It can be used for two classification problems, binary and multiple. The main point is that each function must be treated separately. The Naive Bayes approach checks independently of all associations the likelihood of each function and predicts the Bayes theorem. Therefore, this approach is called "naive" –features also have some degree of association between themselves in real-world problems.

So as to comprehend the calculation of Naive Bayes, class likelihood and contingent likelihood ideas ought to be included already.

a. The probability of a class in the dataset is a statistic. It is the likelihood of a particular class if we pick a random object from the data set.

b. The likelihood of the capacity esteem in the class is a restrictive likelihood.

1. The likelihood of class is simply determined as the number of samples split between the total number of samples: $P(C) = \frac{\text{count}(\text{instances in } C)}{\text{count}(\text{instances in } N_{\text{total}})}$

2. Conditional probabilities are determined in accordance with the frequency of each variable separated by the instance frequency of that feature.

$$P(V|C) = \frac{\text{count}(\text{instances with } V \text{ and } C)}{\text{count}(\text{instances with } V)}$$

3. Due to the likelihood of a class case, we can calculate the probability and therefore determine with the Bayes theorem:

$$P(A|B) = P(B|A)P(A) P(B)$$

4. The odds are contrasted with that of the object belonging to all classes and the most likely class if selected.

The benefits of this approach include consistency and ease of comprehension. It also works well on data sets with insignificant characteristics, as they are not likely to contribute to their performance. Consequently, forecasts are not taken into account. In addition, this algorithm also results in good output in relation to used resources, because there is no need to find any coefficients, as in other algorithms, to only measure the probabilities of the features and classes. As mentioned above, its main disadvantage is that every function is treated separately, even though this can not be true in most cases. (Monsignor 2006)

3.5.2.4 J48 decision tree

As the name proposes, choice trees are tree-framed information structures. For the tree development, which is then utilized for anticipating test information, the preparation informational index is utilized. With the least number of decisions, this algorithm aims to achieve the most accurate result. For classification problems as well as regression concerns, decision trees may be used. Table 3.1 provides an example:

Table 3.1: Decision tree example dataset

Predictors				Target
Outlook	Temperature	Humidity	Windy	Play tennis
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Overcast	Cool	High	True	No
Rainy	Cool	High	True	Yes
Rainy	Mild	High	False	No
Sunny	Cool	Normal	False	Yes

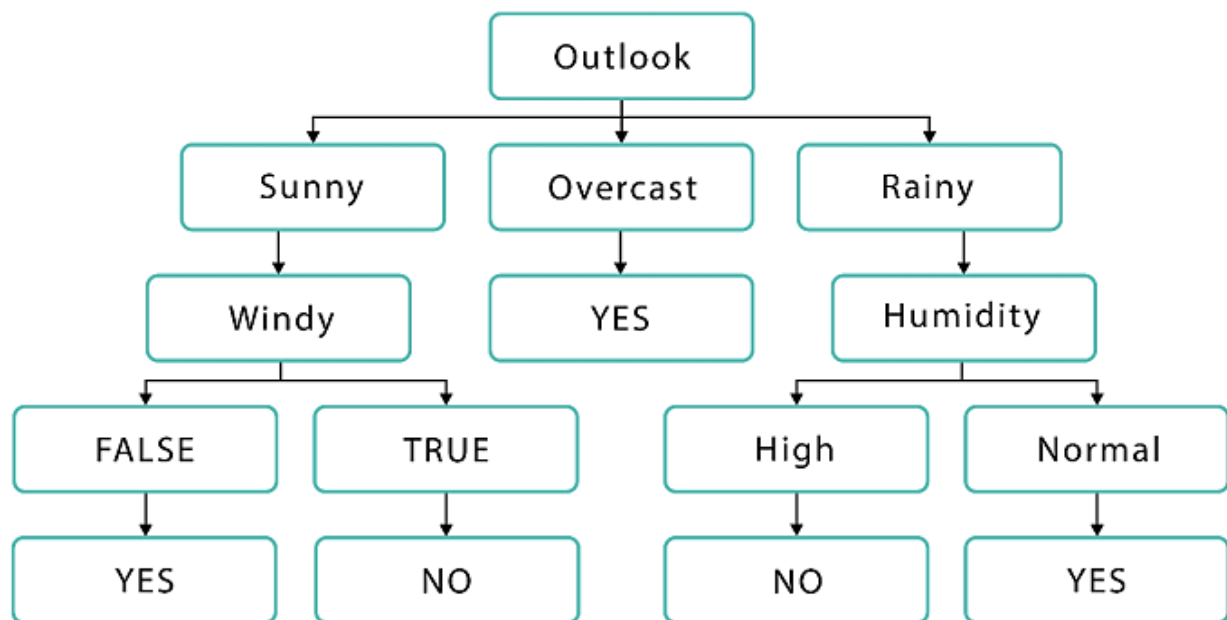


Figure 3.6: Decision tree example

The model is trained on dataset, and the tennis option can now be marked as "yes" or "no," according to Figure 3.6. Here, there are decision nodes and leaf nodes in the tree. There are several branches in decision nodes that lead to leaf nodes. Leaf nodes reflect choices or rankings. The first node is called the root node.

It is the common decision-making algorithm ID3 (Iterative Dichotomiser 3). It is based on the principles of enterprise and knowledge benefit. Entropy applies here to the degree to which the data material is unknown. For instance, the coin toss entropy would be undetermined, as the outcome is not certain. Converse, a coin toss on both sides of the coin will lead to null entropy as we could predict the outcome before each toss with a 100% probability. (The Review Museum 1997).

The ID3 algorithm can be represented in simple words as follows: from the root node, the data is to be divided in a homogeneous (similar in structure) dataset at every point. We wish to find the most common feature, that means the most standardized divisions (Swain and Hauska 1977), that would result in the highest benefit of knowledge.

1. Measure the target entropy.

$$E(T, X) = \sum P(c)E(c) \quad c \in X$$

$$(S) = \sum -p_i \log_2 p_i \quad i=1 \text{ [12] [13]}$$

2. Divide the dataset and calculate each branch entropy. The division benefit in knowledge is then determined, which is the differences in the initial entropy and the proportionate sum of the branch entropies.

$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

3. The highest Gain value attribute is selected as the decision node.

4. If an entropy of 0 is present in any of the branches of the chosen decision node, it becomes the blade node. Further separation is required for other branches.

5. The algorithm runs recursively until nothing is broken.

J48 is a R-packet ID3 algorithm and this is the feature we will use in our study.

Because of its simplicity, the method of the decision tree became popular. It can handle large datasets well and can handle noise very well in datasets. One advantage is that, as opposed to other algorithms, such as SVM or KNN, the "White Box" is used for decision trees, which means we can easily see what outcomes are reached and which decisions have contributed to them. These facts have made it popular for medical diagnoses, spam filters, security checks, etc.

3.5.2.5 Random forest random

Forest is one of the most well-known AI calculations. There is basically no requirement for information arranging and demonstrating, yet the tests are generally solid. In the last area the irregular woodlands depend on the choices taken. Random woods, in particular, have better predictions for clusters of decision-making trees. It is thus known as a 'wood' — eventually a series of decision-making bodies.

The basic idea is to build various option trees based on the free dataset sub-sets. Each hub selects n factors from the ability list, and the best split is found between them. The computation can be represented in simple words (Biau 2013)[61]: In simple words:

1. Different trees are fabricated generally on the two third of the preparation information (62.3%). Information is picked haphazardly.
2. A few indicator factors are arbitrarily chosen out of all the indicator factors. At that point, the best split on these chose factors is utilized to part the hub. As a matter of course, the measure of the chose factors is the square foundation of the all-out number of all indicators for grouping, and it is steady for all trees.
3. Utilizing the remainder of the information, the misclassification rate is settled. The all out blunder rate is settled as the generally standing up of-pack mess up rate.
4. Each prepared tree gives its own strategy result, giving its own "vote". The class that got the most "votes" is picked as the outcome. The game plan of the include is found in Figure 3.7.

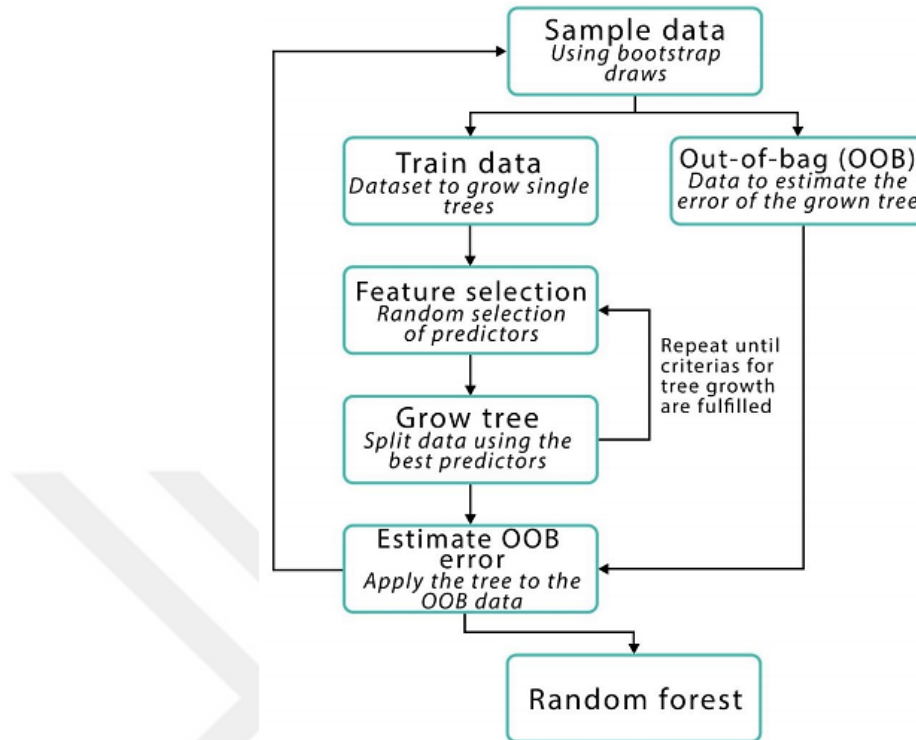


Figure 3.7: Random Forest scheme

Like in decision-making trees, this algorithm does not take into account the need for selecting the function to exclude irrelevant features. When dimensionality is needed, any functional selection with random forestry algorithms only occurs. In addition, the out - of-bag error rate, mentioned previously, can be regarded as a cross-validation method of the algorithm. This removes the need for repetitive, often required cross-validation steps. (Spanish version 1997).

Random forests are a legacy of many of the benefits of decision-making algorithms. We apply to regression and classification issues; we can easily calculate and fit. Typically they often lead to greater precision. Unlike decision trees, however, interpreting the findings is not very straightforward. Within decision-making bodies, we are able to learn within depth about the important variables and the consequences of their outcomes by analyzing the resulting tree. Random forests are not possible. It can also be considered a more robust algorithm than the decision treaties—decision-making will most likely change if we adjust the data a little. This will not occur in random forestry algorithms – the random forest will remain stable because it combines multiple decisions trees.

3.5.3 Cross-Validation

In the methods of precision assessment, they cannot say if the model operates on new data. The downside is how the methods are assessed. The solution for addressing this drawback is cross-validation. The goal is to separate the first dataset. The model is trained and tracked for the majority of the results. Instead the model is assessed. There are three separate cross-validation groups:

1. Unit Holdout—the informational index is partitioned into two regions: a preparation pack and a test set. The example is incredible for rehearsing. The model is then tried in the test go that has never been seen. The subsequent blunders are utilized to compute the mean outright test mistake for model assessment. The high speed is the advantage of this system. On the other hand, the results of the assessment depend strongly on the selection of the test set because the variance is generally high. The results of the assessment can therefore differ substantially between various test sets.
2. The k-fold technique is considered to boost the holdout process. In this case, the k sub-sets are chosen and the holdout process is repeated in k instances. If a sub-set of k is used as a training set and the k-1 subset is used as a test set. On all k holdout races the mean error is calculated. The variance is decreased with the increase of k so that the accuracy of different datasets does not improve. The downside is the difficulty and operating time that are higher than the holdout method.
3. The departure method is the extreme case of the k-fold strategy, where the k is as large as the sample universe. Data are trained at every holdout cycle on only one data point, and a single point is then evaluated. There is as little variation as possible in this case. At the other hand, the code complexity is high.

3.6 CONCLUSION

This part introduced foundation data on AI, which is significant for understanding the useful usage of the undertaking, as characterized in the following section. The standards for practical condition, useful extraction and determination strategies and the calculations for AI were talked

about. The calculations chose incorporate K-Nearest Neighbors, Vector Support, Decision Trees, Random Forests and Naive Bays.



4. PRACTICAL PART

The objective of the project is to create the best function and extraction procedures, the most accurate algorithm that allows families of malware to differentiate with the lowest error rate and current accuracy of the system. This chapter addresses the practical dimensions of project execution. This involves the collection of data, the determination of malware families, the selection of characteristics used for the algorithm, the best method to represent features, the method for evaluating and the implementation process.

4.1 DATA

A total of 2 140 files have been compiled for this project. In the majority of these cases, hashes that clearly identify files were contained in incident reports or reverse engineering software reports, which are then used for the use of external malware researchers to obtain the appropriate samples from the VirusTotal database. Nine malware families were utilized to work with an enormous dataset, creating 1.156 malevolent documents and 984 amiable records. The malevolent families that are utilized are Dridex, Locky, TeslaCrypt, Vawtrak, Zeus, DarkComet, CyberGate, Xtreme and CTB-Locker. Right now are examined in detail. Benign files primarily contained a.exe format program installer, but also various.pdf,.docx, etc. files because they were used often as malware vice versa. Only malware that emerged in the last two years is used to produce the most important and up-to-date tests.

4.1.1 Dridex

Dridex is the first family of malware with 172 single files. The malware, in particular, is part of the Trojan class. The consequence was a major outbreak in 2015, with a monthly infection level of 3 000-5 000.

Dridex comes from the 2012 malware Cridex. Cridex was also a bank credentials stealer, but in particular it was a worm used as a propagating tool for attached storage devices. In 2014, there was a new version, which shifted from power and control to peer-topic communication and therefore became more robust.

During a major spam campaign, Dridex Malware files were mostly distributable, using real

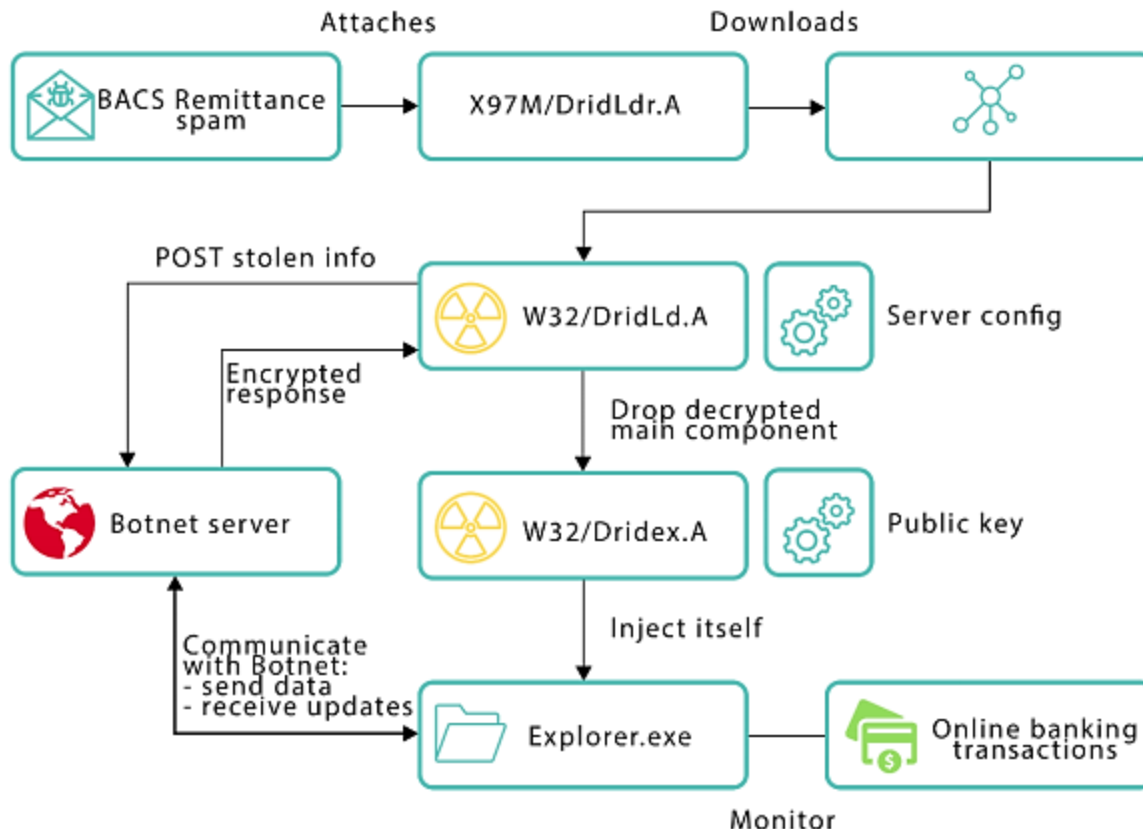


Figure 4.1: Dridex operation scheme (Aquino 2014)

Corporate names for a sender with fake high-level domains appropriate for the locations of target users. Much of the messages were either invoices or orders. Dridex attackers have shown great attention to detail: during business hours emails with real company names are also sent and individual employees are used.

Figure 4.1 defines the operating scheme of Dridex. Dridex ' malware has been used in Word document macros from a technical point of view. A.vbs file was executed and run after these macros, Dridex Trojan was downloaded and enabled. Dridex carries out a middle man attack, Integrate into web browsers like Chrome, FireFox or Internet Explorer, and monitor online banking link traffic and searches. After the data is found, Dridex steals keylogs, computer footage and input forms. (Brien 2016). O'Brien. Education. Practice.

4.1.2 Locky

Locky is the second class of malware, with 115 separate files. This ransomware encrypts all data on the machine of the user using ciphers RSA-2048 and AES-256 and adds a. Locky was created and vigorously spread in February 2016. Spam campaigns, especially fake invoices and phishing websites, constitute the most popular distribution vectors. The use of financial papers and macros indicates that Dridex group is responsible for this malware is close to those used to spread Dridex. The price of decrypting system files varies from 0.5 to 1 Bitcoin. (Sequence of the Lord).

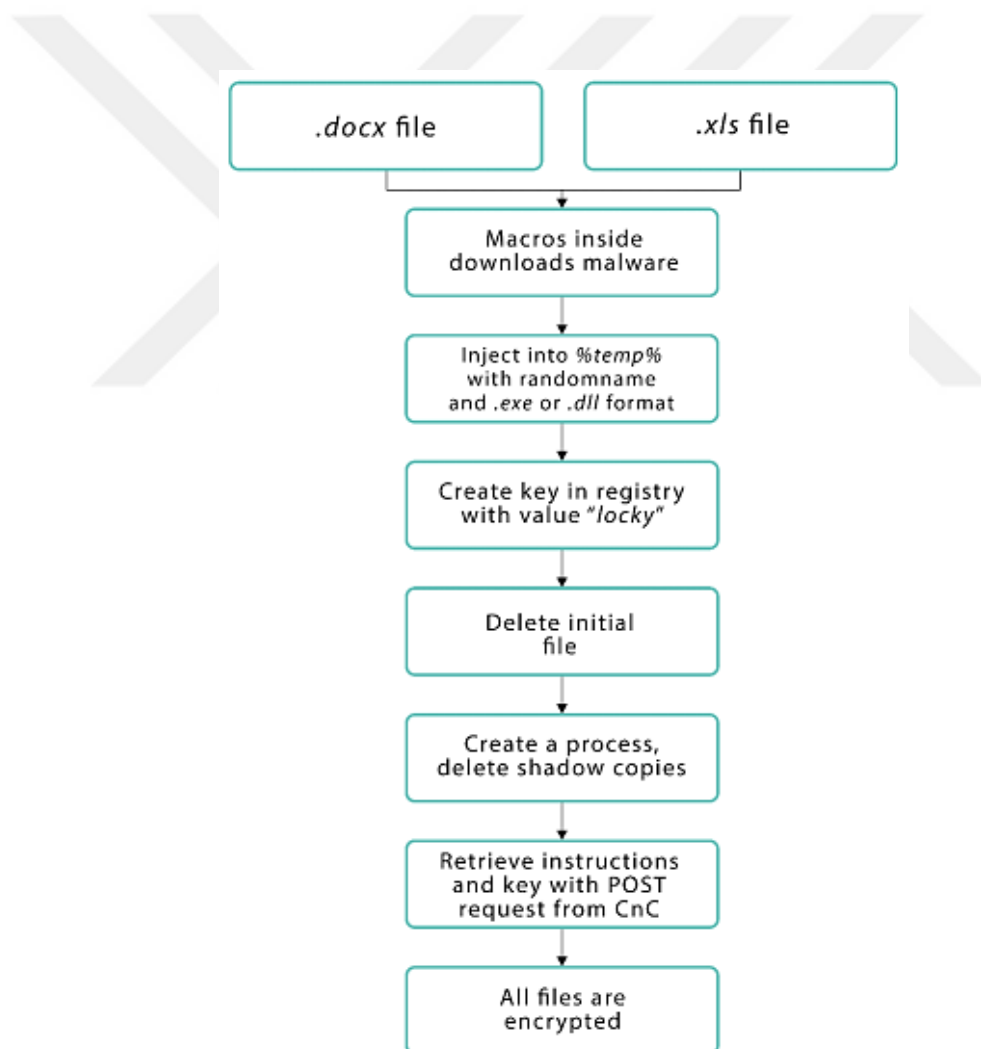


Figure 4.2: Locky operation

When the macros are sent to the device, it runs and installs the Locky malware stored in a.docxor.xls files. Malware, in effect, is loaded in a random name and a.exe or.dll file format into the percent temp percent tab. The registry instead adds a "run" registry key with "Locky" name, which shows the.exe file in the temp percent folder for the number. At this point, the original file is removed. After this, a new process will begin to analyze the volume characteristics and remove the copies of the shadows.

4.1.3 Teslacrypt

Teslacrypt is the third malware family of 115 files and ransom ware. Homepages and emails containing links to malicious websites that access this malware after it has been accessed are the main vehicle of distribution. The file is executed immediately following download.. Figure 4.3 displays the operating scheme for Teslacrypt.

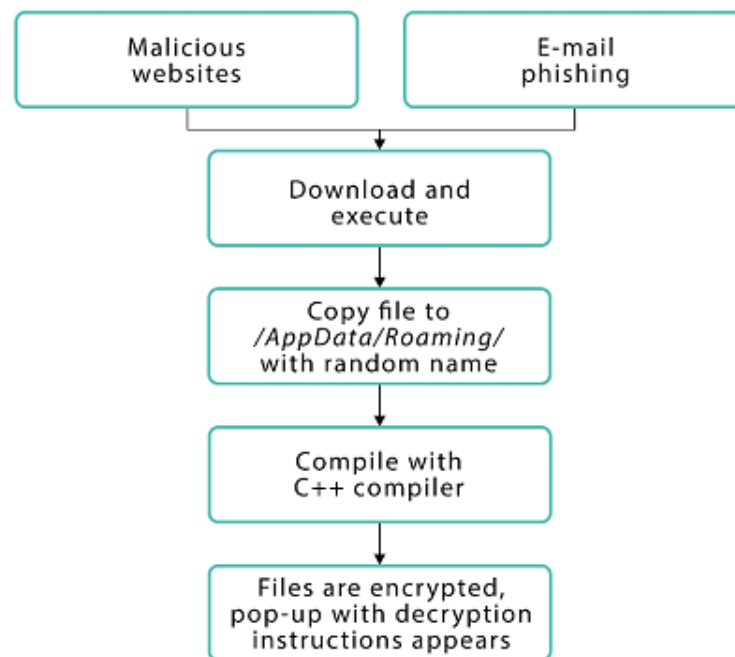


Figure 4.3: TeslaCrypt operation

4.1.4 Vawtrak

Vawtrak is the fourth single-file malware family. Vawtrak is another example of the banking Trojan, also known as Neverquest or Snifula. In the Czech Republic, USA, UK and Germany the most infections occurred. Vectors distributed include downloaders of malware, spam on malicious links or other drive-by downloads. Vawtrak can then access a person's bank accounts and rob identities, passwords, personal keys, and so on after download.

This family of malware is illustrated in Figure 4.4. The first file that is downloaded to the drive results in a percentage system data folder with an extension and the file name enabled. The first file will be removed after that. (The Spanish Province, 2015). This dropper is a DLL which disconnects and injects the Vawtrak module into running processes. With the hardcoded key the DLL will decrypt the payload to produce a new DLL to replace the original.

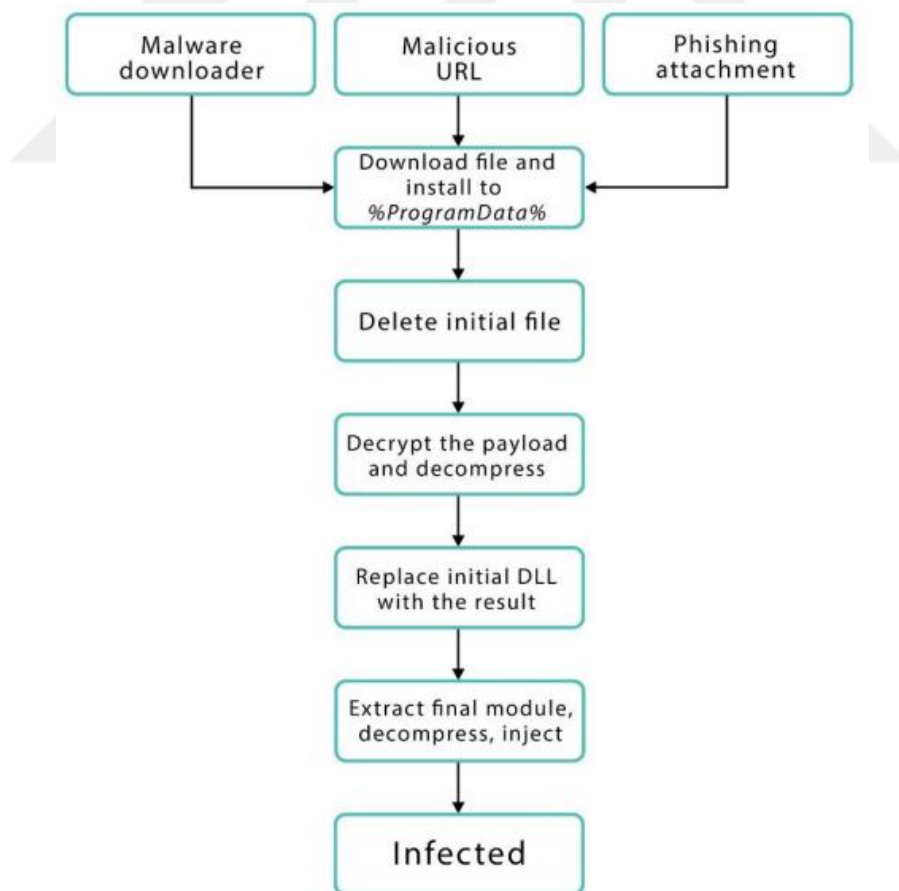


Figure 4.4: Vawtrak operation

4.1.5 Zeus

Zeus is the fifth class of malware and has 116 separate files. It is an easily tradable botnet kit for around USD 700 on the black market. Zeus has developed and continues to be one of the most common representatives for botnet malware following its launch in 2007.

Figure 13 offers a description of the Zeus activity. Zeus infection vectors vary greatly from spam e-mails to run drive-by. After download, the malware will insert the *sdra64.exe* file and change the registry values to be executed when the program is running. Zeus subsequently injects in the *winlogon.exe* method and ends the executable initial.

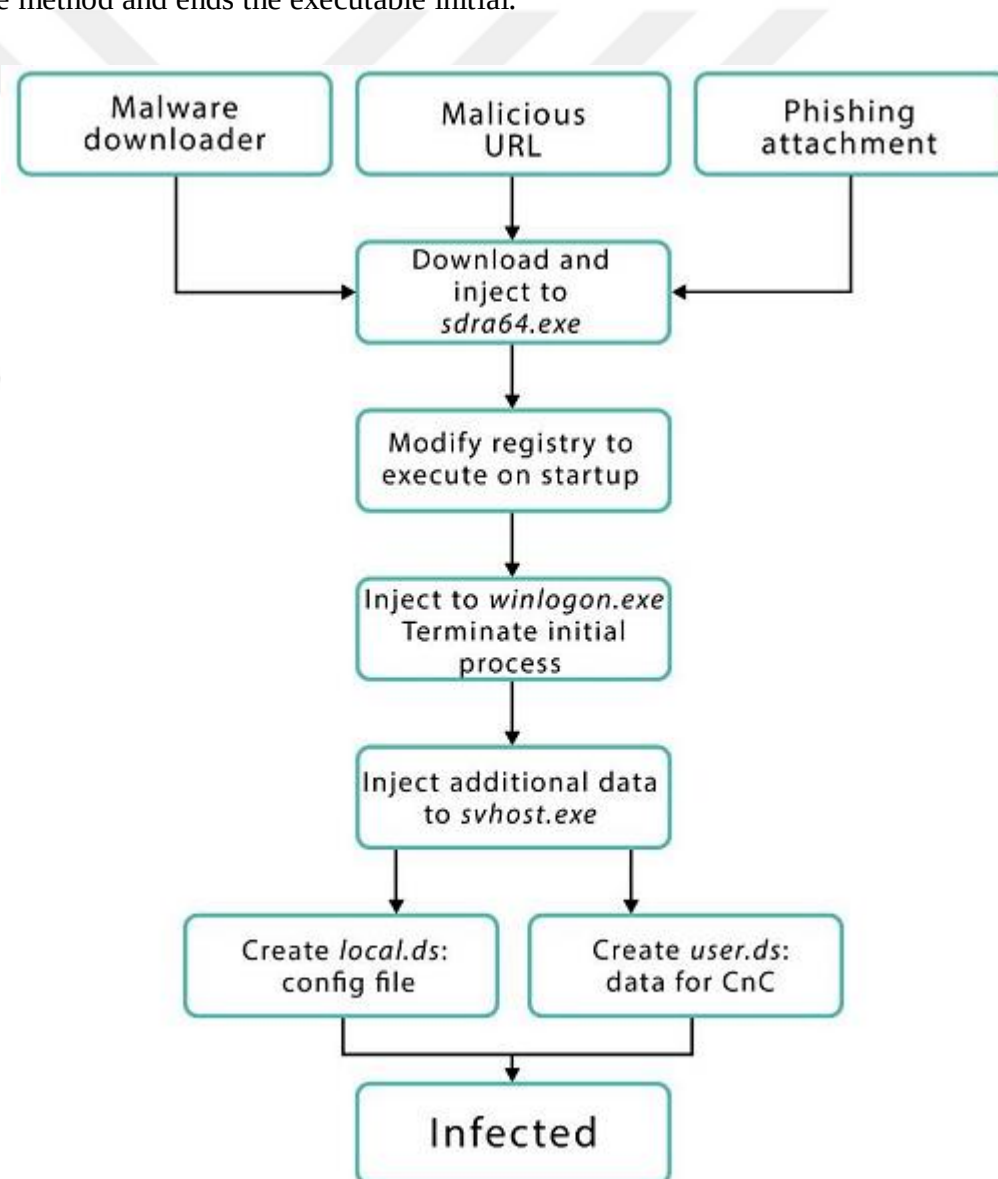


Figure 4.5: Zeus operation

4.1.6 DarkComet

DarkComet is a remote management tool (RAT) example. In 2012-2015, it was used in many attacks. Originally, DarkComet was not a malicious device, but it was ultimately employed by the Syrian government for the purpose of spying because of its existence and work, Followed in the years following by many other attacks.

The DarkComet contains the client and the server, like any other RATs. However, the attacker considers the server to be a malware machine and the intruder to be the consumer differently. The DarkComet relies on its remote architecture which makes it possible for the user to control the server before the server runs. (Kujawa 2012) Functionality of the DarkComet is wide, but not limited (Kujawa 2012).

- Webcam and sound capture
- Power off/Shutdown/Restart
- Remote Desktop functionality
- Active ports discovery
- LAN computers discovery
- URL download

Figure 4.6 demonstrates how the server interacts with the client..

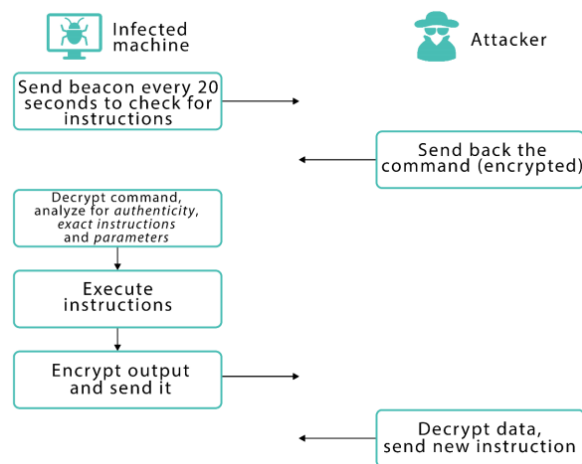


Figure 4.6: DarkComet communication scheme

4.1.7 CyberGate

The Remote Administration Tool (RAT) is another example of CyberGate. Written in Delphi, it is continually built and therefore robust and detailed features. CyberGate may be referred to as "legal" malware because it has first been created and used in legal cases. It should also be noted. This is often used however for illegal activities, such as spying.

CyberGate helps you to: • Sign in to the victim's device • Access system screenshots • Connect simultaneously to several users. Capturing data from connected input devices • Lock machine • Start, shutdown • Read and change registries • Communicate via shell •

4.1.8 Xtreme

Xtreme is a second RAT example. It has been developed in Delphi and is freely available with several other Delphi RAT malware, such as CyberGate.

A number of government warnings and attacks on Israel and Palestine have been used by Xtreme. Xtreme's architecture is based on a server model that includes a database. The settings are stored in the \Microsoft / Windows \APPDATA percent folder or in the mutex directory.

Xtreme function (Bennett and Villeneuve 2014) helps the attacker to:

- Using connected device data, such as a mobile device, camera, etc. Manipulate business processes • Read and change the register • Interact via remote shell • Remote Snap • Save files from connected devices •

4.1.9 CTB-Locker

The last malware family to be used was CTB-Locker, which used 79 different file settings. This is an instance of a ransomware program that encrypts user files that pay the decryption key for money. CTB is a Curve Tor Bitcoin acronym that refers to a crypt algorithm of the Elliptical Curve.

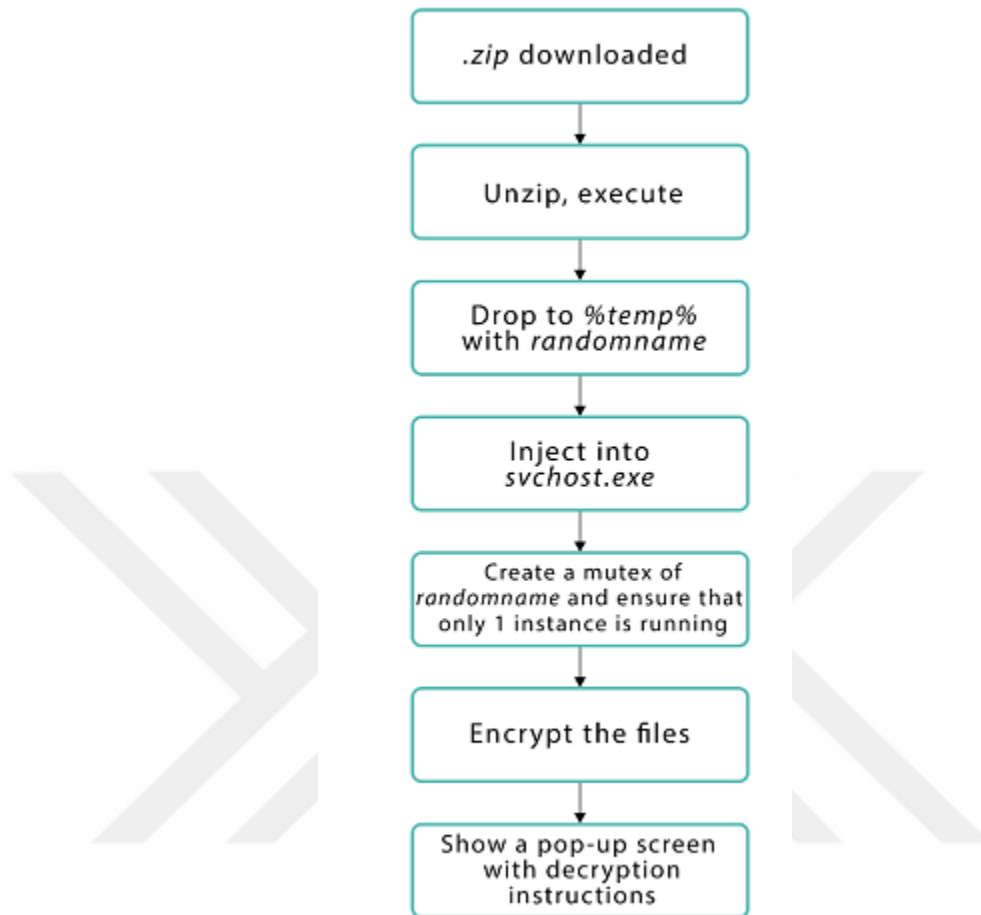


Figure 4.7: CTB-Locker operation

The CTB-Locker samples were distributed via e-mails using malicious appendices. Attachments shown in the downloader.zip archives. Figure 4.7 explains the initial process of CTB-Locker. After malware is executed, it drops to the randomly named temp percent folder and injects into the svchost.exe method. In addition, a random name mutex is created to ensure that only one CTB-Locker instance resides on the computer.

4.2 CUCKOO SANDBOX

The study is based on and aimed at Cuckoo Sandbox. Clearly, the data should be interpreted in every way to apply algorithms to machine learning to any problem. Cuckoo Sandbox has been used for this purpose. The reports created by the sandbox that describe and sample's behavioral data were pre-processed and malware was extracted here. However, the design of the sandbox and the report structure are important to understand first. Cuckoo Sandbox is a malware detection open-source tool that can detect any file or URL within seconds.

• DLL files • Microsoft Office documents • URLs and records with HTML • Wordpress contents • PHP documents • Visual Basic (VB), contents • ZIP-files • Java JAR • Python-Files • Practically whatever else permitted in document positions, as indicated by the establishment Cuckoo (2015).

Cuckoo has a very flexible modular architecture that can be used as an individual or integrated into a larger framework. The main components of the Cuckoo network are a host computer and a set of guest machines (virtual or physical research equipment). The situation is fundamental. Once the new file is forwarded, the virtual environment is allocated to the server, the file is executed and all activities carried out within the system are documented.

4.3 SCORING SYSTEM

The rating for the cuckoo analysis shows how dangerous a file is. The ranking is measured by the number of malicious measures taken. To order to detect malicious activity, Cuckoo uses the list of malicious actions called signatures. Each of these signatures has an indicator of the seriousness of the action carried out.

For total, there are 3 severity levels and each has a severity score: one for low, two for medium and three for high. The behavior of querying a name of the machine An example of a hallmark of low gravity. An example of a medium-specific signature is the creation of executable files. The removal of a shadow copy is a high level signature example.

All actions are processed for processing subsequently during analysis. In the end, multiple modules are used to analyze the stored behavior including the Signature Module. All collected data is analyzed in the Signatures module and patterns are found that suit a signature. When the signature matches, the strength of the signature (1, 2, or 3) will raise the counter. If all signatures are collected, the counter value is divided by 5.0 for a point point score that is floating. The Cuckoo study score is this number. Figure 4.8 gives an example of signatures of different gravity.

Signatures

Queries for the computername (5 events)
This executable has a PDB path (1 event)
The executable has PE anomalies (could be a false positive) (1 event)
One or more processes crashed (6 events)
Checks adapter addresses which can be used to detect virtual network interfaces (1 event)
Drops a binary and executes it (1 event)
Allocates read-write-execute memory (usually to unpack itself) (2 events)
A process attempted to delay the analysis task. (1 event)
Creates a suspicious process (4 events)
Executes one or more WMI queries (1 event)
One or more potentially interesting buffers were extracted, these generally contain injected code, configuration data, etc.
Performs some HTTP requests (6 events)
Creates an Alternate Data Stream (ADS) (3 events)
Deletes its original binary from disk (1 event)
Attempts to detect Cuckoo Sandbox through the presence of a file (1 event)
Installs itself for autorun at Windows startup (1 event)
Removes the Shadow Copy to avoid recovery of the system (1 event)
One or more of the buffers contains an embedded PE file (1 event)
Executed a process and injected code into it, probably while unpacking (14 events)
File has been identified by 44 AntiVirus engines on VirusTotal as malicious (44 events)

Figure 4.8: Severity levels of cuckoo signatures

Table 4.1 explains the total number of malware families used in the project. The color indicates the corresponding level of maliciousness to the score.

Table 4.1: Cuckoo scores for malware families

Family	Average Cuckoo score
Benign	1.04
Dridex	5.26
Locky	6.41
Teslacrypt	6.27
Vawtrak	2.66
Zeus	6.46
DarkComet	5.15
CyberGate	6.57
Xtreme	5.15
CTB-Locker	4.76

4.3.1 Reports and Features

We have to decide which kind of data should be collected and how it should be presented to apply machine learning algorithms to the problems.

Some field works use string properties or file format properties to represent features.. However, format-specific extraction of the functionalities is not the best solution, as examined file formats can be substantially different. (Position 2011).

The so-called n-grams are used for other plays. Bit n-gram is a sliding window accumulated with the overlap of substrates that dive one bit at a time Word N-grams and Character N-grams are commonly used. (Pujari and Reddy 2006).

Nonetheless, there are several drawbacks to such a strategy. The big issue is that the collection of n-grams derived from bytes from malware and from benign programs is very large, and that classification methods cannot be applied directly. The key problem is byte n-grams. byte n-grams. (Pujari and Reddy 2006). Furthermore, the approach also limits polymorphic malware detection capacity. In this case, different strings and therefore different n grams result in the samples that produce the same behavior.

In this analysis, it is determined that the actual file actions that the sandbox monitors will be based on the above factors. Due to this fact. In general, Files • Register key• Processes • IP addresses and API requests • API calls this section defines the role should be used in our work, as specified. The following functions can be defined extracted from the sandbox.

4.3.1.1 Files

The reports contain information on files that were opened, written files and produced. This information is good at predicting the family of malware, since every malware file causes many changes to the file system. It can in most cases be used to identify malware very accurately. Nevertheless, depending exclusively on file modifications could lead, for instance, to a lack of differentiation between different families in the case of ransom ware. This is because each system file is encrypted by ransom ware. Consequently the feature set consists mainly of the coded files. Variations among the ransom ware families are indicated by the malware files, which are far below the entire set of functions and, therefore, it would be very difficult to predict based on these details.

4.3.1.2 Registry keys

The registry holds low-level operating system configurations and programs on Windows systems. Every sample running on the device will cause a large number of changes to the registry. The reports on Cuckoo will display open, read, write, delete registry keys. The registry modification information can be a good source of data on malware-caused system changes and can be used to detect malware.

4.3.1.3 Mutexes

Mutual exclusion is the mutex. This is a software object which can share the same resource for several threads. A mutex with a unique name is generated every time you start a program. Mutex names may be strong sample malware identifiers. Nonetheless, they can not achieve the exact result on a broad scale for families, given that there are substantially fewer mutexes generated per sample than the data set. That is why the slight adjustment in the bug or the non-start phase will affect the prediction results dramatically.

4.3.1.4 Processes

The common malware sample identifier is the name of the process created. It can be used, however, very rarely for defining the malware family, because the process names in the common case are the same as the sample hash. The malware sample can be used as an option in the device process. Therefore, it is poor for the identity of the family.

4.3.1.5 IP addresses and DNS queries

Cuckoo offers network traffic information in PCAP format, from which data on contacts can be collected for both IP addresses and DNS queries. The IP addresses of command and control servers for attackers are accurately identified in this data, and in most cases the malware family can therefore be accurately identified. Nevertheless, attackers also alter or spoof their servers' domain names or IP addresses. Therefore, relying only on such information is inaccurate.

4.3.1.6 API calls

The API refers to a number of tools that provide an interface with software components. The application programming interface. API calls will be logged and link to the specific malware execution process. We describe all the operating system happenings, including file operations, registry operations, mutexes, processes and other functions referred to earlier. API calls, for instance, `OpenFileEx`, `CopyFileEx`, `CreateFile`, etc. Calls `OpenMutex`, `CreateMutex` and `CreateMutexEx`, describe opened and created mutexes, etc., and define file operations. Traces of the API call provide a detailed sample behavior description, including all the above features. Therefore, a large number of distinct values are included. In addition, they are simple to define and so they have been selected as features in numerical format. Here, the functional set is

specified by the number and return codes of the single API calls. The following section explains in more detail the representation method.

4.4 FEATURE REPRESENTATION

With our familiarization with the features in the study of Cuckoo Sandbox, we can now understand how the features for machine learning algorithms can be interpreted. As the feature set is very wide and includes the failed, active APIs and return codes, it must be displayed clearly, compactly and redundantly. This function is defined by the frequency (binary) matrix, outlined in the following section.

4.4.1 Binary Representation

Parallel portrayal is the least difficult and most effortless way to deal with getting to the fizzled and effective API calls. A grid wherein the lines speak to the examples and the segments are the API calls is produced right now. 0 is the "awful" call condition for the API and 1 is the real require the API.

$$API_{bin} = \begin{matrix} & API_1 & API_2 & \dots & API_n \\ \begin{matrix} S_1 \\ S_2 \\ \vdots \\ S_n \end{matrix} & \begin{bmatrix} 1 & 1 & \dots & 0 \\ 1 & 0 & \dots & 1 \\ \vdots & \ddots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{bmatrix} \end{matrix}$$

Although this method is simple and straightforward, the return codes produced are not taken into account and the same API call has been triggered many times, resulting in reduced accuracy.

4.4.2 Feature Selection

The goal of the feature selection is to exclude features that are not required because they are too large. Greater sets of functions are difficult to use, but other features do not affect the algorithm decision of this package and can therefore be omitted. In our case, for example, certain API calls could only be triggered once in a sample. This particular API call would play no role in the

algorithm for a large and varied set of function values and thus would have little effect on accuracy if it is removed.

We finished with 70518 features after extracting features and presenting them as a matrix. This is too much to process and forecast accurately. For example, with such a large feature set, loading the dataset, preprocessing and running the nearest algorithm on an x64 8 GB RAM machine takes approximately two-three hours. These services are inappropriate and redundant features need to be disabled.

4.5 IMPLEMENTATION

This phase is followed by the creation of the research plan and its implementation. In the following steps, the whole implementation process can be described: 1. Configuration of sandbox 2. Extraction of functionality (use of Python 2.7) 3. Selection of function (R) 4. 4. Machine-learning (using R) 5 applications. 5. Results assessment Each of these measures is further discussed in this chapter.

4.5.1 Sandbox Configuration

It is necessary to configure Cuckoo Sandbox to receive malware behavior report and make sure the malware is running correctly, including all the functionality. Various malware samples target various vulnerabilities in the real world that may include certain software products. Thus, it is important that the virtual machines that are produced with the sandbox include a wide range of services.

Virtualbox is the hypervisor used on Cuckoo's virtual machines. VMcloak, an automated generation of virtual machines and cloaking tool for Cuckoo Sandbox, builds the virtual machines. (French only). (French only). The following conditions apply to all virtual machines:

The supported program is: • Adobe FlashPlayer 11.7.700.169-Visual Studio redirection kit for the period 2005-2013, including a Service Pack 1 • Adobe PDP Reader 9.0. · Visual Studio Redirection kit for the year 2005-2013. • The.NET 4.0 screen for Java JRE 7.

4.5.2 Feature Extraction

A combined matrix which includes successful APIs, failed APIs and their return codes as discussed above is the feature representation approach chosen. The definition is taken from the papers in the sandbox.

Figure 4.9 explains the detailed process of removing features. The reports are saved locally in our implementation after the sandbox has been published. The.csv file that contains the combined matrix inside, is then used as an input to the extractor-function script. The algorithm must define the number of minimum API calls; for example, it is possible to omit all reports causing less than five API calls. The file contains the time stamp extraction and logs which record productive and failing operations will be stored in a separate file.

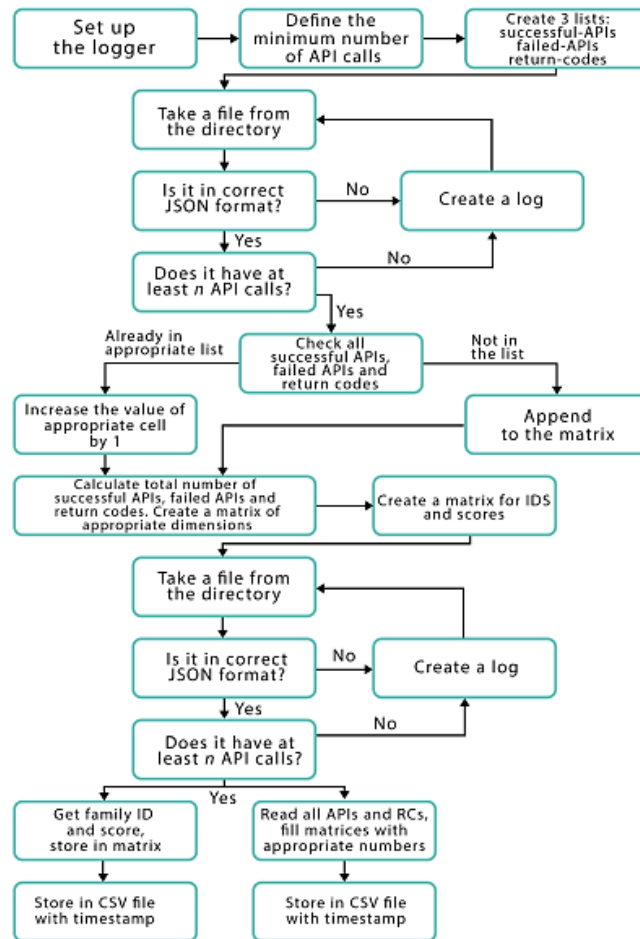


Figure 4.9: Feature extraction process

4.5.3 Feature Selection

The feature selection is used, as stated in the final chapter, to exclude redundant and irrelevant characteristics in order to improve predictive precision. In this case, the set of features is very large and the selection of features is therefore very important.

The language R is used for choosing and implementing the learning methods of the machine. R is a free computer and graphics software environment. It builds and operates on a number of UNIX, Windows and MacOS platforms. (Smith and Venables 2016).

The Boruta kit is a strong and easy algorithm for selecting features in classification issues. In general, the wrapper method works around the algorithm Random Forest. Their algorithm (Kursa and Rudnicki 2010) can be defined as follows:

1. Build shuffled copies (to add randomness) of all functions. This is called shadow copies.
2. Train the Random Forest rating for the new data series and take an important step in using the Mean Decline Accuracy algorithm. At this level, weight is determined and the value of individual functions is determined.
3. Check whether the weight of the original function is higher than the highest weight of the shadow copy on each iteration. Disable features that are not relevant at each iteration.
4. Avoid categorizing all features as 'selected' or 'met' or a random number of forest iterations.

In comparison to other methods of feature selection, Boruta enables the recognition of all features related to the tests. In addition, other methods rely on a small subset of features, which leads to minimal error. (Rudnicki and Kursa 2010). Once we continue using the function collection, the problem arises. The Boruta package, having 70 518 features, does not require sufficient memory and cannot work. We will split the data set accordingly.

The issue emerges when we begin actualizing the element choice. Having 70 518 highlights, the Boruta bundle debilitates, as it can't distribute enough memory and can't run. In this way, We will randomly divide the data set into the subsets that can fit in the memory and perform the highlight determination for each. We compile all the highlights that have become important and aggregate the subsets and forget all the irrelevant highlights. The next step is for all datasets to

return to the item choice. After the calculation of the item option, 306 highlights were completed. The presentation of this change was assessed dependent on the KNN precision with the given list of capabilities. KNN was picked for this issue, as it is the main calculation that can procedure the entire list of capabilities – it doesn't store some other data other than the dataset and doesn't manufacture models, in contrast to different calculations. In the wake of evacuating superfluous highlights, the precision of identification dependent on KNN improved by around 1% and the forecast took roughly three seconds.

4.5.4 Application of Machine Learning Methods

After the highlights were emptied and picked, we can apply the AI strategies to the information that we got. The AI procedures to be applied, as talked about starting at now, are K-Nearest Neighbors, Support Vector Machines, J48 Decision Tree, Naive Bayes, and Random Forest. The general approach is plot in Figure 4.10.

The bundles utilized for the usage of calculations are:

- K-Nearest Neighbors – class
- Support Vector Machines – kernlab
- J48 Decision Tree – RWeka
- Naive Bayes – e1071
- Random Forest – random Forest
- CrossTable plotting – gmodels

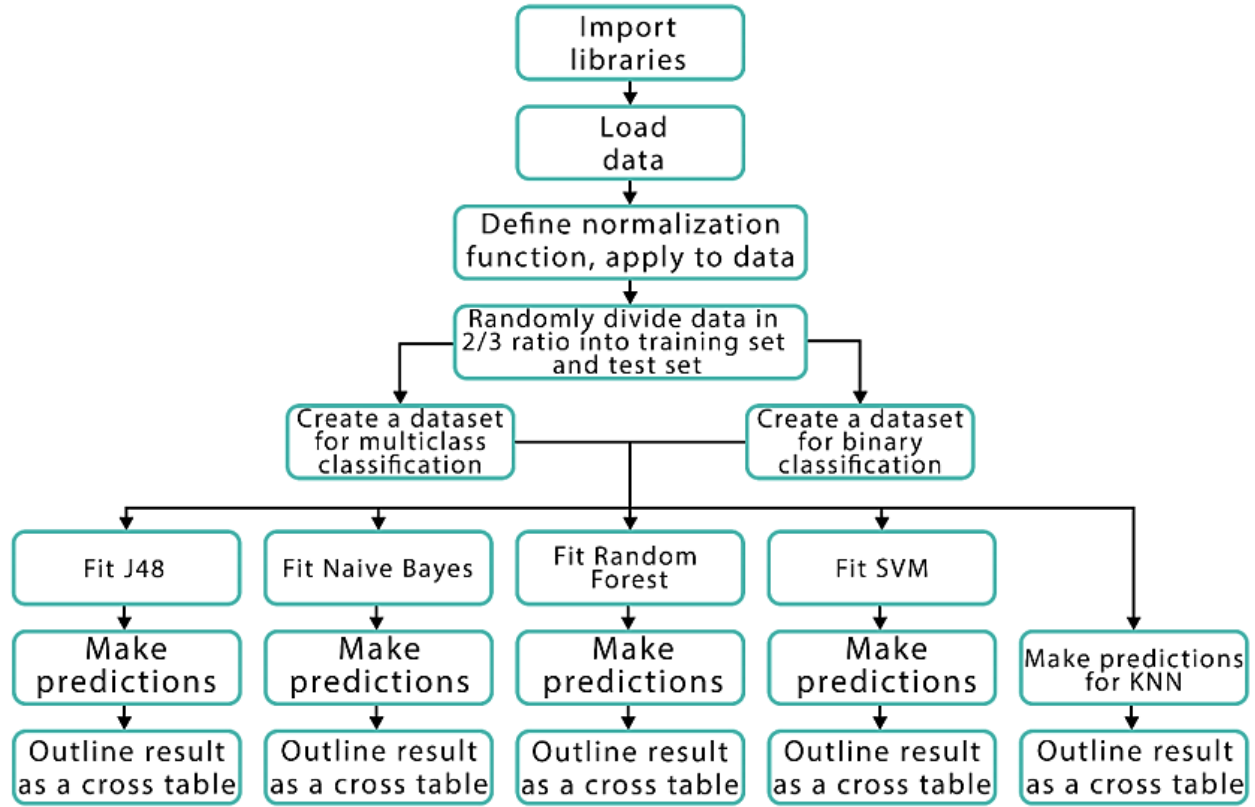


Figure 4.10: Machine learning classification scheme

4.6 RESULTS AND DISCUSSION

The results of tests of applied machine learning methods are discussed in this chapter. The identification sensitivity is calculated according to the percentage of correctly identified instances:

$$Accuracy = \frac{\text{count(Correctly identified samples)}}{\text{count(Total samples)}} \quad 5.1$$

4.6.1 K-Nearest Neighbors

In the cross table in Figure 4.11, the consequence of the K-Nearest process can be inferred. It is necessary to understand the results described here as follows: rows are the actual sample groups, whereas the columns indicate the expected values. Therefore, in the cell of first row and 1st column the number of proper examples of first class is indicated. The cells in row 1 and in column 2 show the number of 1st class instances defined as 2nd class[15].

Total Observations in Table: 371

selectedLapis.testlabels	model_pred										Row Total
	1	2	3	4	5	6	7	8	9	10	
0	0 0.000 0.000 0.000	0 0.000 0.000 0.000	1 1.000 0.034 0.003	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	1 0.003
1	49 0.803 0.860 0.132	0 0.000 0.000 0.000	1 0.016 0.034 0.003	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	2 0.033 0.038 0.005	2 0.033 0.049 0.005	5 0.082 0.132 0.013	2 0.033 0.091 0.005	61 0.164
2	1 0.027 0.018 0.003	31 0.838 0.969 0.084	2 0.054 0.069 0.005	0 0.000 0.000 0.000	1 0.027 0.045 0.003	2 0.054 0.059 0.005	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	37 0.100
3	2 0.074 0.035 0.005	0 0.000 0.000 0.000	22 0.815 0.759 0.059	1 0.037 0.023 0.003	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	2 0.074 0.091 0.005	27 0.073
4	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	43 0.977 0.977 0.116	1 0.023 0.045 0.003	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	44 0.119
5	2 0.111 0.035 0.005	1 0.056 0.031 0.003	0 0.000 0.000 0.000	0 0.000 0.000 0.000	15 0.833 0.682 0.040	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	18 0.049
6	3 0.075 0.053 0.008	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	4 0.100 0.182 0.011	30 0.750 0.882 0.081	0 0.000 0.000 0.000	1 0.025 0.024 0.003	1 0.025 0.026 0.003	1 0.025 0.045 0.003	40 0.108
7	0 0.000 0.000 0.000	0 0.000 0.000 0.000	1 0.020 0.034 0.003	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	47 0.959 0.904 0.127	0 0.000 0.000 0.000	1 0.020 0.026 0.003	0 0.000 0.000 0.000	49 0.132
8	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	38 1.000 0.927 0.102	0 0.000 0.000 0.000	0 0.000 0.000 0.000	38 0.102
9	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	0 0.000 0.000 0.000	1 0.029 0.029 0.003	2 0.059 0.038 0.005	0 0.000 0.000 0.000	31 0.912 0.816 0.084	0 0.000 0.000 0.000	34 0.092
10	0 0.000 0.000 0.000	0 0.000 0.000 0.000	2 0.091 0.069 0.005	0 0.000 0.000 0.000	1 0.045 0.045 0.003	1 0.045 0.029 0.003	1 0.045 0.019 0.003	0 0.000 0.000 0.000	0 0.000 0.000 0.000	17 0.773 0.773 0.046	22 0.059
Column Total	57 0.154	32 0.086	29 0.078	44 0.119	22 0.059	34 0.092	52 0.140	41 0.111	38 0.102	22 0.059	371

Figure 4.11: CrossTable for KNN multi-class classification

The test collection, as seen, includes 371 samples and 1 sample had a "0" class error. Table 4.2 displays the classification accuracy.

Table 4.2: KNN multi-class accuracy

Class	Family	Correctly classified	Incorrectly classified	Accuracy	Average Cuckoo score
1	Benign	49	12	80.3%	1.04
2	Dridex	31	6	83.8%	5.26
3	Locky	22	5	81.5%	6.41
4	TeslaCrypt	43	1	97.7%	6.27
5	Vawtrak	15	3	83.3%	2.66
6	Zeus	30	10	75%	6.46
7	DarkComet	47	2	95.9%	5.15
8	CyberGate	38	0	100%	6.57
9	Xtreme	31	3	91.2%	5.15
10	CTB-Locker	17	5	77.3%	4.76

We randomly selected the train and test set from the 2/3 ratio dataset. The data can not therefore be identical. The most probable cause is that the classes are so distributed that when the KNN algorithm is used, the boundaries are visible..

The total accuracy of the K-Nearest Neighbors depends on the k value. In our case, different values were tested. They produced the following accuracy:

- k=1: 87%
- k=2: 84.63%
- k=3: 81.3%
- k=4: 80%
- k=5: 80%
- k=6: 80%
- k=10: 77.8%

As it can be seen, the best accuracy was achieved with k=1, and the accuracy was 87%. This is an unusual case – when the best accuracy is achieved with k=1 it can be a sign of one of the following:

Total Observations in Table: 371

selected_apis.testlabels.twoway	model_twoway		Row Total
	1	2	
1	49 0.803 0.860 0.132	12 0.197 0.038 0.032	61 0.164
2	8 0.026 0.140 0.022	302 0.974 0.962 0.814	310 0.836
Column Total	57 0.154	314 0.846	371

Figure 4.12: CrossTable for KNN binary classification

There have been two-class malware classifications and benevolent files. Figure 4.12 displays the corresponding cross-table. Class 1 is the benign file in the table, while class 2 is the malicious file. Once, forecasts with various k values have been made:

- k=1: 94.6%
- k=2: 94.3%
- k=3: 93.5%
- k=5: 93.5%
- k=7: 92.7%

K=1-94.6 percent was the highest performance. Tables 4.3 display the thorough accuracy.

Table 4.3: KNN binary accuracy

Class	Correctly identified instances	Incorrectly identified instances	Accuracy
Benign	49	12	80.3%
Malicious	302	8	97.4%

Overall, 87 percent for multi-classe classification and 94.6 percent for two-class classification resulted in a good accuracy of the KNN algorithm. The algorithm provided good results, we can conclude. In multi-class classifications, grades shall be equally distributed, which also has an effect on the good accuracy of the forecasts. While the distribution in two groups (310 vs.61) is not yet right, the findings remain.

4.6.2 Support Vector Machines

Support for vector machines was the next algorithm tested. Figure 4.4 shows the results of the forecasts. For multiclass classification the overall accuracy obtained was 87.6 percent and for binary classification it was 94.6 percent.

Table 4.4: SVM CrossTable

Total Observations in Table: 371

selected_apis.testlabels	predictions	1	2	3	4	5	6	7	8	9	10	Row Total
0	1	0	0	0	0	0	0	0	0	0	0	1
	3.347	0.086	0.057	0.100	0.038	0.151	0.132	0.100	0.084	0.059	0.003	0.003
	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	0.014	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	0.003	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1	56	0	0	0	0	5	0	0	0	0	0	61
	164.742	5.261	3.453	6.084	2.302	1.923	8.057	6.084	5.097	3.617	0.164	0.164
	0.918	0.000	0.000	0.000	0.000	0.082	0.000	0.000	0.000	0.000	0.000	0.000
	0.778	0.000	0.000	0.000	0.000	0.089	0.000	0.000	0.000	0.000	0.000	0.000
	0.151	0.000	0.000	0.000	0.000	0.013	0.000	0.000	0.000	0.000	0.000	0.000
2	3	32	0	0	1	1	0	0	0	0	0	37
	2.434	260.056	2.094	3.690	0.112	3.764	4.887	3.690	3.092	2.194	0.100	0.100
	0.081	0.865	0.000	0.000	0.027	0.027	0.000	0.000	0.000	0.000	0.000	0.000
	0.042	1.000	0.000	0.000	0.071	0.018	0.000	0.000	0.000	0.000	0.000	0.000
	0.008	0.086	0.000	0.000	0.003	0.003	0.000	0.000	0.000	0.000	0.000	0.000
3	2	0	21	0	0	4	0	0	0	0	0	27
	2.003	2.329	248.084	2.693	1.019	0.001	3.566	2.693	2.256	1.601	0.073	0.073
	0.074	0.000	0.778	0.000	0.000	0.148	0.000	0.000	0.000	0.000	0.000	0.000
	0.028	0.000	1.000	0.000	0.000	0.071	0.000	0.000	0.000	0.000	0.000	0.000
	0.005	0.000	0.057	0.000	0.000	0.011	0.000	0.000	0.000	0.000	0.000	0.000
4	0	0	0	37	1	6	0	0	0	0	0	44
	8.539	3.795	2.491	242.365	0.263	0.062	5.811	4.388	3.677	2.609	0.119	0.119
	0.000	0.000	0.000	0.841	0.023	0.136	0.000	0.000	0.000	0.000	0.000	0.000
	0.000	0.000	0.000	1.000	0.071	0.107	0.000	0.000	0.000	0.000	0.000	0.000
	0.000	0.000	0.000	0.100	0.003	0.016	0.000	0.000	0.000	0.000	0.000	0.000
5	3	0	0	0	10	5	0	0	0	0	0	18
	0.070	1.553	1.019	1.795	127.901	1.918	2.377	1.795	1.504	1.067	0.049	0.049
	0.167	0.000	0.000	0.000	0.556	0.278	0.000	0.000	0.000	0.000	0.000	0.000
	0.042	0.000	0.000	0.000	0.714	0.089	0.000	0.000	0.000	0.000	0.000	0.000
	0.008	0.000	0.000	0.000	0.027	0.013	0.000	0.000	0.000	0.000	0.000	0.000
6	7	0	0	0	2	31	0	0	0	0	0	40
	0.075	3.450	2.264	3.989	0.159	103.203	5.283	3.989	3.342	2.372	0.108	0.108
	0.175	0.000	0.000	0.000	0.050	0.775	0.000	0.000	0.000	0.000	0.000	0.000
	0.097	0.000	0.000	0.000	0.143	0.554	0.000	0.000	0.000	0.000	0.000	0.000
	0.019	0.000	0.000	0.000	0.005	0.084	0.000	0.000	0.000	0.000	0.000	0.000
7	0	0	0	0	0	1	48	0	0	0	0	49
	9.509	4.226	2.774	4.887	1.849	5.531	266.483	4.887	4.094	2.906	0.132	0.132
	0.000	0.000	0.000	0.000	0.000	0.020	0.980	0.000	0.000	0.000	0.000	0.000
	0.000	0.000	0.000	0.000	0.000	0.018	0.980	0.000	0.000	0.000	0.000	0.000
	0.000	0.000	0.000	0.000	0.000	0.003	0.129	0.000	0.000	0.000	0.000	0.000
8	0	0	0	0	0	0	1	37	0	0	0	38
	7.375	3.278	2.151	3.790	1.434	5.736	3.218	291.027	3.175	2.253	0.102	0.102
	0.000	0.000	0.000	0.000	0.000	0.000	0.026	0.974	0.000	0.000	0.000	0.000
	0.000	0.000	0.000	0.000	0.000	0.000	0.020	1.000	0.000	0.000	0.000	0.000
	0.000	0.000	0.000	0.000	0.000	0.000	0.003	0.100	0.000	0.000	0.000	0.000
9	0	0	0	0	0	1	0	0	31	0	0	34
	6.598	2.913	1.925	3.391	1.283	0.886	4.491	3.391	279.106	2.016	0.092	0.092
	0.000	0.000	0.000	0.000	0.000	0.088	0.000	0.000	0.912	0.000	0.000	0.000
	0.000	0.000	0.000	0.000	0.000	0.054	0.000	0.000	1.000	0.000	0.000	0.000
	0.000	0.000	0.000	0.000	0.000	0.008	0.000	0.000	0.084	0.000	0.000	0.000
10	0	0	0	0	0	0	0	0	0	22	0	22
	4.270	1.898	1.245	2.194	0.830	3.321	2.906	2.194	1.838	328.305	0.059	0.059
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.059	0.000	0.000
Column Total	72	32	21	37	14	56	49	37	31	22	371	371
	0.194	0.086	0.057	0.100	0.038	0.151	0.132	0.100	0.084	0.059	0.003	0.003

Detailed information on each class' accuracy is available in Table 4.5.

Table 4.5: SVM multiclass accuracy

Class	Family	Correctly classified	Incorrectly classified	Accuracy	Average Cuckoo score
1	Benign	56	5	91.8%	1.04
2	Dridex	32	5	86.5%	5.26
3	Locky	21	6	77.8%	6.41
4	TeslaCrypt	37	7	84%	6.27
5	Vawtrak	10	8	55.6%	2.66
6	Zeus	31	9	77.5%	6.46
7	DarkComet	48	1	98%	5.15
8	CyberGate	37	1	97.4%	6.57
9	Xtreme	31	3	91.2%	5.15
10	CTB-Locker	22	0	100%	4.76

Cell Contents

N
Chi-square contribution
N / Row Total
N / Col Total
N / Table Total

Total Observations in Table: 371

selected_apis.testlabels.twoway	predictions		Row Total
	1	2	
1	41	20	61
	174.102	21.631	
	0.672	0.328	0.164
	1.000	0.061	
	0.111	0.054	
2	0	310	310
	34.259	4.256	
	0.000	1.000	0.836
	0.000	0.939	
	0.000	0.836	
Column Total	41	330	371
	0.111	0.889	

Figure 4.13: SVM binary classification CrossTable

The binary classification cross-table is illustrated in Figure 4.13. Examples of binary classification can also be found in Tables 4.6. As you see, 41 malicious instances were correctly identified (true negatives) –310, benevolent instances incorrectly recognized (false positives) –20, malicious instances wrongly identified (false negatives) –0

Table 4.6: SVM binary classification accuracy

Class	Correctly identified instances	Incorrectly identified instances	Accuracy
Benign	41	20	67.2%
Malicious	310	0	100%

The corresponding accuracies are almost the same as those of the K-Nearest neighbours, which were 87.6 percent for multi-class rating and 94.6 percent for binary rating. This algorithm, in effect, resulted in a binary classification of 0 false negatives—that means there were no malware samples classified as benign. This means it is more effective than K-nearest neighbors to prevent malware infection.

4.6.3 J48 Decision Tree

The choice tree J48 was the third calculation to be tried. The advantage of the strategy for the Decision Tree is that it works with the "white box" and we can perceive what choices prompted our expectation. The multi-class and differential rating choice tree is found in Figures 4.14.

Total Observations in Table: 371

selected_api.testlabels	predictions										Row Total
	1	2	3	4	5	6	7	8	9	10	
0	1	0	0	0	0	0	0	0	0	0	1
	4.447	0.111	0.065	0.121	0.059	0.100	0.127	0.111	0.089	0.059	0.003
	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
	0.017	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
	0.003	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
1	54	0	0	1	4	0	0	0	1	0	61
	202.294	6.741	3.946	5.534	0.040	4.248	7.728	6.741	3.610	3.617	0.164
	0.885	0.000	0.000	0.016	0.066	0.016	0.000	0.000	0.016	0.000	
	0.915	0.000	0.000	0.022	0.182	0.027	0.000	0.000	0.030	0.000	
	0.146	0.000	0.000	0.003	0.011	0.003	0.000	0.000	0.003	0.000	
2	0	37	0	0	0	0	0	0	0	0	37
	5.884	264.894	2.394	4.488	2.194	3.690	4.687	4.089	3.291	2.194	0.100
	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
	0.000	0.902	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
	0.000	0.100	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
3	2	0	24	0	0	0	0	0	0	1	27
	1.225	2.984	283.524	3.275	1.601	2.693	3.420	2.984	2.402	0.226	0.073
	0.074	0.000	0.889	0.000	0.000	0.000	0.000	0.000	0.000	0.037	
	0.034	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.045	
	0.005	0.000	0.065	0.000	0.000	0.000	0.000	0.000	0.000	0.003	
4	0	0	0	44	0	0	0	0	0	0	44
	6.997	4.863	2.846	280.092	2.609	4.388	5.574	4.863	3.914	2.609	0.119
	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
	0.000	0.000	0.000	0.978	0.000	0.000	0.000	0.000	0.000	0.000	
	0.000	0.000	0.000	0.119	0.000	0.000	0.000	0.000	0.000	0.000	
5	1	1	0	0	16	0	0	0	0	0	18
	1.212	0.492	1.164	2.183	208.906	1.795	2.280	1.989	1.601	1.067	0.049
	0.056	0.056	0.000	0.000	0.889	0.000	0.000	0.000	0.000	0.000	
	0.017	0.024	0.000	0.000	0.727	0.000	0.000	0.000	0.000	0.000	
	0.003	0.003	0.000	0.000	0.043	0.000	0.000	0.000	0.000	0.000	
6	1	3	0	0	2	33	0	1	0	0	40
	4.518	0.456	2.588	4.852	0.058	210.975	5.067	2.647	3.558	2.372	0.108
	0.025	0.075	0.000	0.000	0.050	0.825	0.000	0.025	0.000	0.000	
	0.017	0.073	0.000	0.000	0.091	0.892	0.000	0.024	0.000	0.000	
	0.003	0.008	0.000	0.000	0.005	0.089	0.000	0.003	0.000	0.000	
7	0	0	0	0	0	0	47	2	0	0	49
	7.792	5.415	3.170	5.943	2.906	4.887	268.065	2.154	4.358	2.906	0.132
	0.000	0.000	0.000	0.000	0.000	0.000	0.959	0.041	0.000	0.000	
	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.049	0.000	0.000	
	0.000	0.000	0.000	0.000	0.000	0.000	0.127	0.005	0.000	0.000	
8	0	0	0	0	0	0	0	38	0	0	38
	6.043	4.199	2.458	4.609	2.253	3.790	4.814	272.053	3.380	2.253	0.102
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.927	0.000	0.000	
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.102	0.000	0.000	
9	0	0	0	0	0	2	0	0	32	0	34
	5.407	3.757	2.199	4.124	2.016	0.570	4.307	3.757	277.620	2.016	0.092
	0.000	0.000	0.000	0.000	0.000	0.059	0.000	0.000	0.941	0.000	
	0.000	0.000	0.000	0.000	0.000	0.054	0.000	0.000	0.970	0.000	
	0.000	0.000	0.000	0.000	0.000	0.005	0.000	0.000	0.086	0.000	
10	0	0	0	0	0	1	0	0	0	21	22
	3.499	2.431	1.423	2.668	1.305	0.650	2.787	2.431	1.957	297.344	0.059
	0.000	0.000	0.000	0.000	0.000	0.045	0.000	0.000	0.000	0.955	
	0.000	0.000	0.000	0.000	0.000	0.027	0.000	0.000	0.000	0.955	
	0.000	0.000	0.000	0.000	0.000	0.003	0.000	0.000	0.000	0.057	
column Total	59	41	24	45	22	37	47	41	33	22	371
	0.159	0.111	0.065	0.121	0.059	0.100	0.127	0.111	0.089	0.059	

Figure 4.15: Decision Tree multi-class CrossTable

In Table 4.7 you can find detailed results of every malware unit.

Table 4.7: Decision Tree multi-class accuracy

Class	Family	Correctly classified	Incorrectly classified	Accuracy	Average Cuckoo score
1	Benign	54	7	88.5%	1.04
2	Dridex	37	0	100%	5.26
3	Locky	24	3	88.9%	6.41
4	TeslaCrypt	44	0	100%	6.27
5	Vawtrak	16	2	88.9%	2.66
6	Zeus	33	7	82.5%	6.46
7	DarkComet	47	2	95.9%	5.15
8	CyberGate	38	0	100%	6.57
9	Xtreme	32	2	94.1%	5.15
10	CTB-Locker	21	1	95.5%	4.76

The algorithm has led to a correct 46 cases of benign (true negatives) samples for the binary classification problem, 305 incorrectly recognized (true positives) malware samples, 15 incorrectly identified (false positive samples), and 5 incorrectly classified (false negatives). Figure 30 and Tables 4.8 provide information.

selected_apis.testlabels.twoway	predictions		Row Total
	1	2	
1	46	15	61
	168.727	26.891	
	0.754	0.246	0.164
	0.902	0.047	
	0.124	0.040	
2	5	305	310
	33.201	5.291	
	0.016	0.984	0.836
	0.098	0.953	
	0.013	0.822	
Column Total	51	320	371
	0.137	0.863	

Figure 4.16: Decision Tree binary classification CrossTable

Table 4.8: Decision Tree binary classification accuracy

Class	Correctly identified instances	Incorrectly identified instances	Accuracy
Benign	46	15	75.4%
Malicious	305	5	98.4%

The general precision of J48 Decision Tree was extraordinary: 93.3% for multi-class and 94.6% for parallel positioning. This outcome is better than that of the K-Nearest neighbors and multi-class vector machines. In any case, for paired characterization, the outcome is the equivalent.

4.6.4 Naive Bayes

Naive Bayes was the fourth algorithm studied. The corresponding precision for multi-class classification was 72.23 percent and binary classification was 55 percent. Figure 4.16 displays the cross table for the Naive Bayes classification.

selected_api.testlabels	predictions										Row Total
	1	2	3	4	5	6	7	8	9	10	
0	0	0	0	0	0	0	1	0	0	0	1
	0.094	0.005	0.084	0.102	0.054	0.075	4.662	0.100	0.127	0.205	
	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.003
	0.000	0.000	0.000	0.000	0.000	0.000	0.018	0.000	0.000	0.000	
	0.000	0.000	0.000	0.000	0.000	0.000	0.003	0.000	0.000	0.000	
1	34	0	2	0	1	0	2	0	4	18	61
	138.633	0.329	1.882	6.248	1.593	4.604	5.799	6.084	1.798	2.424	
	0.557	0.000	0.033	0.000	0.016	0.000	0.033	0.000	0.066	0.295	0.164
	0.971	0.000	0.065	0.000	0.050	0.000	0.035	0.000	0.085	0.237	
	0.092	0.000	0.005	0.000	0.003	0.000	0.005	0.000	0.011	0.049	
2	0	1	4	0	11	0	0	0	3	18	37
	3.491	3.213	0.267	3.790	40.658	2.792	5.685	3.690	0.607	14.326	
	0.000	0.027	0.108	0.000	0.297	0.000	0.000	0.000	0.081	0.486	0.100
	0.000	0.500	0.129	0.000	0.550	0.000	0.000	0.000	0.064	0.237	
	0.000	0.003	0.011	0.000	0.030	0.000	0.000	0.000	0.008	0.049	
3	0	0	25	0	0	0	0	0	0	2	27
	2.547	0.146	229.287	2.765	1.456	2.038	4.148	2.693	3.420	2.254	
	0.000	0.000	0.926	0.000	0.000	0.000	0.000	0.000	0.000	0.074	0.073
	0.000	0.000	0.806	0.000	0.000	0.000	0.000	0.000	0.000	0.026	
	0.000	0.000	0.067	0.000	0.000	0.000	0.000	0.000	0.000	0.005	
4	0	0	0	33	0	0	0	0	8	3	44
	4.151	0.237	3.677	180.145	2.372	3.321	6.760	4.388	1.056	4.012	
	0.000	0.000	0.000	0.750	0.000	0.000	0.000	0.000	0.182	0.068	0.119
	0.000	0.000	0.000	0.868	0.000	0.000	0.000	0.000	0.170	0.039	
	0.000	0.000	0.000	0.089	0.000	0.000	0.000	0.000	0.022	0.008	
5	0	0	0	2	8	0	1	0	0	7	18
	1.698	0.097	1.504	0.013	50.926	1.358	1.127	1.795	2.280	2.976	
	0.000	0.000	0.000	0.111	0.444	0.000	0.056	0.000	0.000	0.389	0.049
	0.000	0.000	0.000	0.053	0.400	0.000	0.018	0.000	0.000	0.092	
	0.000	0.000	0.000	0.005	0.022	0.000	0.003	0.000	0.000	0.019	
6	0	0	0	3	0	28	2	0	1	6	40
	3.774	0.216	3.342	0.294	2.156	206.719	2.796	3.989	3.265	0.587	
	0.000	0.000	0.000	0.075	0.000	0.700	0.050	0.000	0.025	0.150	0.108
	0.000	0.000	0.000	0.079	0.000	1.000	0.035	0.000	0.021	0.079	
	0.000	0.000	0.000	0.008	0.000	0.075	0.005	0.000	0.003	0.016	
7	0	0	0	0	0	0	49	0	0	0	49
	4.623	0.264	4.094	5.019	2.642	3.698	228.458	4.887	6.208	10.038	
	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.132
	0.000	0.000	0.000	0.000	0.000	0.000	0.860	0.000	0.000	0.000	
	0.000	0.000	0.000	0.000	0.000	0.000	0.132	0.000	0.000	0.000	
8	0	0	0	0	0	0	1	37	0	0	38
	3.585	0.205	3.175	3.892	2.049	2.868	4.010	291.027	4.814	7.784	
	0.000	0.000	0.000	0.000	0.000	0.000	0.026	0.974	0.000	0.000	0.102
	0.000	0.000	0.000	0.000	0.000	0.000	0.018	1.000	0.000	0.000	
	0.000	0.000	0.000	0.000	0.000	0.000	0.003	0.100	0.000	0.000	
9	1	1	0	0	0	0	1	0	31	0	34
	1.519	3.639	2.841	3.482	1.833	2.566	3.415	3.391	165.418	6.965	
	0.029	0.029	0.000	0.000	0.000	0.000	0.029	0.000	0.912	0.000	0.092
	0.029	0.500	0.000	0.000	0.000	0.000	0.018	0.000	0.660	0.000	
	0.003	0.003	0.000	0.000	0.000	0.000	0.003	0.000	0.084	0.000	
10	0	0	0	0	0	0	0	0	0	22	22
	2.075	0.119	1.838	2.253	1.186	1.660	3.380	2.194	2.787	67.901	
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.059
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.289	
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.059	
Column Total	35	2	31	38	20	28	57	37	47	76	371
	0.094	0.005	0.084	0.102	0.054	0.075	0.154	0.100	0.127	0.205	

Figure 4.17: Naive Bayes multi-class classification cross-table

Table 4.9 displays detailed findings that illustrate the accuracy of each malware family.

Table 4.9: Naive Bayes multi-class classification accuracy

Class	Family	Correctly classified	Incorrectly classified	Accuracy	Average Cuckoo score
1	Benign	34	27	55.8%	1.04
2	Dridex	1	36	2.7%	5.26
3	Locky	25	2	92.6%	6.41
4	TeslaCrypt	33	11	75%	6.27
5	Vawtrak	8	10	44.4%	2.66
6	Zeus	28	12	70%	6.46
7	DarkComet	49	0	100%	5.15
8	CyberGate	37	1	97.4%	6.57
9	Xtreme	31	3	91.2%	5.15
10	CTB-Locker	22	0	100%	4.76

The algorithm performed poorly for binary classification. There were 61 malware instances correctly identified (true negatives) 143 malware instances incorrectly identified (false positives) 0, and 167 malware instances incorrectly identified (falsely negatives). Figure 4.18 and Tables 4.10 display detailed results.

Total Observations in Table: 371

selected_apis.testlabels.twoway	predictions		Row Total
	1	2	
1	61	0	61
	14.747	23.512	
	1.000	0.000	0.164
	0.268	0.000	
	0.164	0.000	
2	167	143	310
	2.902	4.627	
	0.539	0.461	0.836
	0.732	1.000	
	0.450	0.385	
Column Total	228	143	371
	0.615	0.385	

Figure 4.18: Naive Bayes binary classification cross-table

Table 4.10: Naive Bayes binary classification accuracy

Class	Correctly identified instances	Incorrectly identified instances	Accuracy
Benign	61	0	100%
Malicious	143	167	46.1%

All in all, the Naive Bayes algorithm was inadequate. Multi-class ratings were 72,23 percent reliable and only 55 percent of binary ratings. This result can not be used to characterize the real world. Furthermore, a number of false-nots, i.e. malware files wrongly labelled benign files, amounted to 167–45% of the files total. This result would cause enormous malware epidemics in a real world in just a short period of time.

It most definitely is because of high reliability between apps that such a bad accuracy occurs. The main disadvantage of the Bayes Naive algorithm, as we know, is that each function is handled independently. In our case, some APIs rely most likely on each other, i.e. without API_m, APIs are not activated. This is the most likely source of the Naive Bayes algorithm's bad result.

4.6.5 Random Forest

The Random Forest algorithm was the last algorithm adopted. The algorithm resulted in a good predictive accuracy, 95.69% of multi-class scoring, and 96.8% of binary classification. Figure 4.19 provides a cross-table for multi-class predictions.

selected_apis.testLabels	predictions										Row Total
	1	2	3	4	5	6	7	8	9	10	
0	1	0	0	0	0	0	0	0	0	0	1
	4.151	0.097	0.070	0.119	0.059	0.100	0.132	0.102	0.092	0.062	
	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.003
	0.016	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
	0.003	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	
1	58	1	0	0	2	0	0	0	0	0	61
	224.190	4.088	4.275	7.235	0.723	6.084	8.057	6.248	5.590	3.782	
	0.951	0.016	0.000	0.000	0.033	0.000	0.000	0.000	0.000	0.000	0.164
	0.935	0.028	0.000	0.000	0.091	0.000	0.000	0.000	0.000	0.000	
	0.156	0.003	0.000	0.000	0.005	0.000	0.000	0.000	0.000	0.000	
2	0	35	0	0	1	0	0	0	0	1	37
	6.183	274.788	2.593	4.388	0.650	3.690	4.887	3.790	3.391	0.730	
	0.000	0.946	0.000	0.000	0.027	0.000	0.000	0.000	0.000	0.027	0.100
	0.000	0.972	0.000	0.000	0.045	0.000	0.000	0.000	0.000	0.043	
	0.000	0.094	0.000	0.000	0.003	0.000	0.000	0.000	0.000	0.003	
3	1	0	25	0	1	0	0	0	0	0	27
	2.734	2.620	282.198	3.202	0.226	2.693	3.566	2.765	2.474	1.674	
	0.037	0.000	0.926	0.000	0.017	0.000	0.000	0.000	0.000	0.000	0.073
	0.016	0.000	0.962	0.000	0.045	0.000	0.000	0.000	0.000	0.000	
	0.003	0.000	0.067	0.000	0.003	0.000	0.000	0.000	0.000	0.000	
4	0	0	0	44	0	0	0	0	0	0	44
	7.353	4.270	3.084	288.218	2.609	4.388	5.811	4.507	4.032	2.728	
	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.119
	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	
	0.000	0.000	0.000	0.119	0.000	0.000	0.000	0.000	0.000	0.000	
5	1	0	0	0	15	2	0	0	0	0	18
	1.341	1.747	1.261	2.135	181.863	0.023	2.377	1.844	1.650	1.116	
	0.056	0.000	0.000	0.000	0.833	0.111	0.000	0.000	0.000	0.000	0.049
	0.016	0.000	0.000	0.000	0.682	0.054	0.000	0.000	0.000	0.000	
	0.003	0.000	0.000	0.000	0.040	0.005	0.000	0.000	0.000	0.000	
6	1	0	1	0	3	35	0	0	0	0	40
	4.834	3.881	1.160	4.744	0.166	241.067	5.283	4.097	3.666	2.480	
	0.025	0.000	0.025	0.000	0.075	0.875	0.000	0.000	0.000	0.000	0.108
	0.016	0.000	0.038	0.000	0.136	0.946	0.000	0.000	0.000	0.000	
	0.003	0.000	0.003	0.000	0.008	0.094	0.000	0.000	0.000	0.000	
7	0	0	0	0	0	0	49	0	0	0	49
	8.189	4.755	3.434	5.811	2.906	4.887	279.472	5.019	4.491	3.038	
	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.132
	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	
	0.000	0.000	0.000	0.000	0.000	0.000	0.132	0.000	0.000	0.000	
8	0	0	0	0	0	0	0	38	0	0	38
	6.350	3.687	2.663	4.507	2.253	3.790	5.019	298.892	3.482	2.356	
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	0.102
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000	
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.102	0.000	0.000	
9	0	0	0	0	0	0	0	0	34	0	34
	5.682	3.299	2.383	4.032	2.016	3.391	4.491	3.482	306.116	2.108	
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.092
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.092	0.000	
10	0	0	0	0	0	0	0	0	0	22	22
	3.677	2.135	1.542	2.609	1.305	2.194	2.906	2.253	2.016	312.233	
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.059
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.957	
	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.059	
Column Total	62	36	26	44	22	37	49	38	34	23	371
	0.167	0.097	0.070	0.119	0.059	0.100	0.132	0.102	0.092	0.062	

Figure 4.19: Random Forest multiclass classification cross-table

Detailed information on each class ' output is available in Table 4.11.

Table 4.11: Random Forest multiclass classification accuracy

Class	Family	Correctly classified	Incorrectly classified	Accuracy	Average Cuckoo score
1	Benign	58	3	95%	1.04
2	Dridex	35	2	94.6%	5.26
3	Locky	25	2	92.6%	6.41
4	TeslaCrypt	44	0	100%	6.27
5	Vawtrak	15	3	83.3%	2.66
6	Zeus	35	5	87.5%	6.46
7	DarkComet	49	0	100%	5.15
8	CyberGate	38	0	100%	6.57
9	Xtreme	34	0	100%	5.15
10	CTB-Locker	22	0	100%	4.76

The result reached 96.8 percent in the binary classification problem. More precisely, 52, malware instances correctly identified (real negatives) hit 307, benign instances wrongly identified (false positives) 9, and malware instances wrongly identified (false negatives) 3. 3. Figures 4.20 and Tables 4.12 provide more detailed information.

Total Observations in Table: 371

selected_apis.testlabels.twoway	predictions		Row Total
	1	2	
1	52 204.055 0.852 0.945 0.140	9 35.516 0.148 0.028 0.024	61 0.164
2	3 40.153 0.010 0.055 0.008	307 6.989 0.990 0.972 0.827	310 0.836
Column Total	55 0.148	316 0.852	371

Figure 4.20: Random Forest binary classification cross-table

Table 4.12: Random Forest binary classification accuracy

Class	Correctly identified instances	Incorrectly identified instances	Accuracy
Benign	52	9	85.2%
Malicious	307	3	99%

The algorithm of Random Forest provided the highest precision between other algorithms. The results for multi grade and binary grading respectively were 95,69 percent and 96,8 percent accuracy. Nonetheless, those false negatives still occur – they are three.

4.7 CONCLUSIONS

In general, the targets set for this study were achieved. The required methods of abstraction and representation were defined and the machine learning algorithms were applied and tested. A composite matrix for the desired method of presenting functions, consisting of the frequency of positive and failing APIs and return codes, was chosen. This was chosen because the file behavior is described. This includes information about various system modifications, including registry updates, mutexes, data, etc, as opposed to other approaches.

5. CONCLUSION AND FUTURE WORK

5.1 CONCLUSIONS

In total, the objectives listed have been accomplished for this review. Selection of the necessary extraction and representation methods and implementation and evaluation of the machine learning algorithms.

A combined matrix was chosen for the desired function representation method to show the frequency of effective API and failed calls along with their return codes. This has been chosen as the actual behavior of the file is described. This incorporates information on various system changes, including registry updates, silent files, and so on, unlike other approaches.

More precise than sand box explore are the Random Wood results. The discoveries can scarcely be estimated quantitatively since the sandbox doesn't classify the examples into unsafe and benevolent. The meaning of malware goes past its capacity. On the other hand, record perniciousness is viewed as a relapse issue, and the earnestness score is its output. However, you can without much of a stretch see the distinction in precision. In Chapter 4.2.1, Table 5.1, no Malware Family was labeled as "red" and no Malware Families as 'green' were labeled. In contrast with 95.69% and 96.8% reached by Rando this figure is very misleading

In light of the past outcomes, the Random Forest Classification Method is prescribed as the outcome was the most ideal precision and superior. Based the outcomes are portrayed previously. While this methodology has additionally acquired the most noteworthy outcome in double grouping, it is suggested that help vector machines be presented. This is on the grounds that 0 bogus negatives came about in no malware tests being set apart as big-hearted. In spite of the fact that the precision of the parallel issue remains the significant concern, the quantity of bogus negatives is additionally critical, since they can prompt gigantic contaminations. Arbitrary Forest, in spite of its high precision, brought about 3 bogus negatives. Therefore, Bolster Vector Machines have caused 0 false negatives, while the accuracy is just 2 percent lower. That is why the update of Support Vector Machines in pair order is necessary.

Table 5.1: Results

Classifier \ Performance		KNN	SVM	Naive Bayes	J48	Random Forest
Multi-class	Accuracy	87%	87.6%	72.34%	93.3%	95.69%
Binary	Accuracy	94.6%	94.6%	55%	94.6%	96.8%
	False-positives	12	20	0	15	9
	False-negatives	8	0	167	5	3
	True-positives	302	310	143	305	307
	True-negatives	49	41	61	46	52

5.2 FUTURE WORK

The investigation acted right now a proof-of-idea. Consequently, a few future enhancements identified with the commonsense execution of this undertaking can be distinguished:

- Implement include extraction in the inline mode

At the moment, after running the documents on the sandbox, extraction of the part takes place and the reports are produced. If done, these approaches can result in delays in the record check. Rather, the highlights should be omitted as the sandbox tests them, so that there are no convincing reasons why the findings should be heard yet again.

- Use a wider dataset

While the dataset used in this study is broad and covers the most important malware forms in the contemporary world, it does not cover all types. It takes a lot of time and energy to compile a malware dataset. To assess the predictors more accurately, it is recommended that models be checked on all potential malware types: spyware, adware, rootkits, backdoors, bank malware, etc. Therefore, it is important to understand that the model can only predict the family samples it saw in the past. In other words, the maximum possible number of families should be used for real-world scenarios in a real-world application before launching the project.



REFERENCES

- [1] "Anonymous Official Website," [Online]. Available: <http://anonofficial.com/>. [Accessed 7 Juny 2015].
- [2] R. Nightingale, "4 Top Hacker Groups And What They Want," MakeUseOf Limited, [Online]. Available: <http://www.makeuseof.com/tag/4-top-hackergroups-want/>. [Accessed 4 March 2015].
- [3] M. Danniels, "Denial of Service Attacks," Imperva, 2015. [Online]. Available: <https://www.incapsula.com/ddos/ddos-attacks/denial-of-service.html>. [Accessed 17 Juny 2015].
- [4] G. Garza, "Catalogs.com," [Online]. Available: <http://www.catalogs.com/info/travel-vacations/top-10-worst-computerviruses.html>. [Accessed 21 3 2015].
- [5] J. Poushter. (2016, Feb 24). Smartphone Ownership and Internet Usage Continues to Climb in Emerging Economies [Online]. Available: <http://www.pewglobal.org/2016/02/22>
- [6] The Statistics Portal. Number of mobile phone users worldwide from 2013 to 2019 (in billions) [Online]. Available: <http://www.statista.com/statistics/274774/forecast-of-mobile-phone-usersworldwide/>
- [7] Yanfang Y, Wang, D, Li, T, Ye, D & Jiang, Q 2008, 'An intelligent PE-malware detection system based on association mining', Journal in Computer Virology, vol.4, no.4, pp.323-334.
- [8] Sami, A, Yadegari, B, Rahimi, H, Peiravian, N, Hashemi, S & Hamze, A 2010, 'Malware detection based on mining API calls', Proceedings of the ACM Symposium on Applied Computing, pp. 1020-1025.
- [9] Freund, Y & Schapire, RE 1996, 'Experiments with a new boosting algorithm', In: Machine learning: Proceedings of the thirteenth international conference, pp. 325-332.

- [10] Anderson, B, Storlie, C & Lane, T 2012, 'Security and artificial intelligence', Proceedings of the 5th ACM workshop AISEC'12, North Carolina, USA, pp. 3-14.
- [11] Szor 2005, 'The Art of Computer Virus Research and Defense', Pearson Education, New Jersey, United States.
- [12] Ralph Langner. Stuxnet: Dissecting a cyberwarfare weapon. Security & Privacy, IEEE, 9(3):49–51, 2011.
- [13] Danilo Bruschi, Lorenzo Martignoni, and Mattia Monga. Using code normalization for fighting self-mutating malware. In Proceedings of the International Symposium on Secure Software Engineering, pages 37–44, 2006.
- [14] Saurabh Anandrao Shivale. Cryptovirology: Virus approach. arXiv preprint arXiv:1108.2482, 2011.
- [15] Ilsun You and Kangbin Yim. Malware obfuscation techniques: A brief survey. In BWCCA, pages 297–300, 2010.
- [16] John Bethencourt, Dawn Song, and Brent Waters. Analysis-resistant malware. 2008.
- [17] Juan Caballero, Chris Grier, Christian Kreibich, and Vern Paxson. Measuring pay-perinstall: The commoditization of malware distribution. In Usenix security symposium, page 15, 2011.
- [18] Sami Zhioua. The middle east under malware attack dissecting cyber weapons. In 2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops, pages 11–16. IEEE, 2013.
- [19] Savan Gadhiya, Kaushal Bhavsa , Techniques for Malware Analysis, International Journal of Advanced Research in Computer Science and Software Engineering.[online] Available at:http://www.ijarcsse.com/docs/papers/Volume_3/4_April2013/V3I4-0371.pdf[Accessed: July 2016]

- [20] Practical Malware Analysis, Kris Kendall. [online] Available at: www.blackhat.com/presentations/bh-dc-07/Kendall_McMillan/Paper/bh-dc-07-Kendall_McMillanWP.pdf [Accessed: August 2016]
- [21] "Microsoft Malware Protection Center - Malware Naming Conventions". Microsoft.com. N.p., 2016. [online] Available at: <https://www.microsoft.com/security/portal/mmpc/shared/malwarenaming.aspx> [Accessed: September 2016].
- [22] Peng, S, Yu, S & Yang, A 2013, 'Smartphone Malware and Its Propagation Modeling: A Survey', IEEE Communications Surveys and Tutorials, vol. 16, no.2, pp. 925-941.
- [23] Google. (). Android security 2016 year in review. Accessed at: 2017-11, [Online]. Available: https://source.android.com/security/reports/Google_Android_Security_2016_Report_Final.pdf.
- [24] Kaspersky. (). It threat evolution q1 2017. Accessed at: 2017-11, [Online]. Available: <https://securelist.com/it-threat-evolution-q1-2017-statistics/78475/>.
- [25] Gator 2005, Gator ewallet - the smart online companion, March 2005. Available from :. [23 March 2005].
- [26] Jianyong Dai, Ratan K Guha, and Joohan Lee. Feature set selection in data mining techniques for unknown virus detection: a comparison study. In CSIIRW, page 56, 2009.
- [27] Dmitry Komashinskiy and Igor Kotenko. Using low-level dynamic attributes for malware detection based on data mining methods. In International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security, pages 254–269. Springer, 2012.
- [28] Guanhua Yan, Nathan Brown, and Deguang Kong. Exploring discriminatory features for automated malware classification. In Detection of Intrusions and Malware, and Vulnerability Assessment, pages 41–61. Springer, 2013a.

- [29] Asaf Shabtai, Robert Moskovitch, Yuval Elovici, and Chanan Glezer. Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey. *Information Security Technical Report*, 14(1):16–29, 2009b.
- [30] Shadi A Aljawarneh, Raja A Mofteh, and Abdelsalam M Maatuk. Investigations of automatic methods for detecting the polymorphic worms signatures. *Future Generation Computer Systems*, 60:67–77, 2016.
- [31] Jithu Raphel and P Vinod. Heterogeneous opcode space for metamorphic malware detection. *Arabian Journal for Science and Engineering*, pages 1–22, 2016
- [32] Manuel Egele, Theodoor Scholte, Engin Kirda, and Christopher Kruegel. A survey on automated dynamic malware-analysis techniques and tools. *ACM Computing Surveys (CSUR)*, 44(2):6, 2012.
- [33] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, J. Nieves, P. G. Bringas, and G. Álvarez Maraón, “Mama: Manifest Analysis for Malware Detection in Android,” *Cybern. Syst.*, vol. 44, no. 6–7, pp. 469–488, 2013.
- [34] Saba Awan and Nazar Abbas Saqib. Detection of Malicious Executables Using Static and Dynamic Features of Portable Executable (PE) File, pages 48–58. Springer International Publishing, Cham, 2016. ISBN 978-3-319-49145-5. doi: 10.1007/978-3-319-49145-5_6. URL http://dx.doi.org/10.1007/978-3-319-49145-5_6.
- [35] Leyla Bilge and Tudor Dumitras. Before we knew it: an empirical study of zero-day attacks in the real world. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 833–844. ACM, 2012.
- [36] S Murugan and K Kuppusamy. System and methodology for unknown malware attack. In *Sustainable Energy and Intelligent Systems (SEISCON 2011)*, International Conference on, pages 803–804. IET, 2011.

- [37] Mihai Christodorescu and Somesh Jha. Testing malware detectors. In ACM SIGSOFT Software Engineering Notes, volume 29, pages 34–44. ACM, 2004.
- [38] Nwokedi Idika and Aditya P Mathur. A survey of malware detection techniques. Purdue University, page 48, 2007.
- [39] K. Tam, A. Feizollah, N. B. Anuar, R. Salleh, and L. Cavallaro, “The evolution of android malware and android analysis techniques”, ACM Computing Surveys (CSUR), vol. 49, no. 4, p. 76, 2017.
- [40] P. Yan and Z. Yan, “A survey on dynamic mobile malware detection”, Software Quality Journal, pp. 1–29, 2017.
- [41] Baskaran, Balaji, and Anca Ralescu. 2016. A Study of Android Malware Detection Techniques and Machine Learning. Proceedings of the 27th Modern Artificial Intelligence and Cognitive Science Conference, 15-23.
- [42] Rieck, Konrad, Philipp Trinius, Carsten Willems, and Thorsten Holz. 2011. Automatic Analysis of Malware Behavior using Machine Learning. Journal of Computer Security.
- [43] S. Dua and X. Du, Data mining and machine learning in cybersecurity. CRC press, 2016.
- [44] M. G. Schultz, E. Eskin, F. Zadok, S. J. Stolfo, Data mining methods for detection of new malicious executables, in: Security and Privacy, 2001. S P 2001. Proceedings. 2001 IEEE Symposium on, 2001, pp. 38–49.
- [45] J. Z. Kolter, M. A. Maloof, Learning to detect and classify malicious executables in the wild, J. Mach. Learn. Res. 7 (2006) 2721–2744.
- [46] M. Gharacheh, V. Derhami, S. Hashemi, S. M. H. Fard, Proposing an hmm-based approach to detect metamorphic malware, in: Fuzzy and Intelligent Systems (CFIS), 2015, pp. 1–5.
- [47] T. Lee, J. J. Mody, Behavioral classification, in: EICAR Conference, 2006, pp. 1–17.

- [48] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, J. Nazario, Automated classification and analysis of internet malware, in: Recent advances in intrusion detection, Springer, 2007, pp. 178–197.
- [49] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, E. Kirda, Scalable, behavior-based malware clustering, in: NDSS, Vol. 9, 2009, pp. 8–11.
- [50] S. Attaluri, S. McGhee, M. Stamp, Profile hidden markov models and metamorphic virus detection, Journal in Computer Virology 5 (2) (2009) 151–169.
- [51] Y. Ye, T. Li, D. Adjero, S. S. Iyengar, A survey on malware detection using data mining techniques, ACM Computing Surveys (CSUR) 50 (3) (2017) 41.
- [52] N. Kawaguchi, K. Omote, Malware function classification using apis in initial behavior, in: Information Security (AsiaJCIS), 2015 10th Asia Joint Conference on, IEEE, 2015, pp. 138–144.
- [53] M. Asquith, Extremely scalable storage and clustering of malware metadata, Journal of Computer Virology and Hacking Techniques (2015) 1–10.
- [54] M. Egele, T. Scholte, E. Kirda, C. Kruegel, A survey on automated dynamic malware-analysis techniques and tools, ACM computing surveys (CSUR) 44 (2) (2012) 6.
- [55] M. G. Schultz, E. Eskin, F. Zadok, S. J. Stolfo, Data mining methods for detection of new malicious executables, in: Security and Privacy, 2001. S P 2001. Proceedings. 2001 IEEE Symposium on, 2001, pp. 38–49.
- [56] I. Santos, F. Brezo, X. Ugarte-Pedrero, P. G. Bringas, Opcode sequences as representation of executables for data-mining-based unknown malware detection, Information Sciences 231 (2013) 64–82.
- [57] M. Eskandari, Z. Khorshidpour, S. Hashemi, Hdm-analyser: a hybrid analysis approach based on data mining techniques for malware detection, Journal of Computer Virology and Hacking Techniques 9 (2) (2013) 77–93.

- [58] J. Z. Kolter, M. A. Maloof, Learning to detect and classify malicious executables in the wild, *J. Mach. Learn. Res.* 7 (2006) 2721–2744.
- [59] M. Siddiqui, M. C. Wang, J. Lee, Detecting internet worms using data mining techniques, *Journal of Systemics, Cybernetics and Informatics* (2009) 48–53.
- [60] Thirumuruganathan, Saravanan. 2010. A Detailed Introduction to K-Nearest Neighbor (KNN) Algorithm. WWW document. Available at: <https://saravananthirumuruganathan.wordpress.com/2010/05/17/a-detailedintroduction-to-k-nearest-neighbor-knn-algorithm/>. [Accessed 15 February 2017]
- [61] Biau, G´erard. 2013. Analysis of a Random Forests Model. *Journal of Machine Learning Research*, 1063-1095.