CAPITAL UNIVERSITY OF SCIENCE AND
TECHNOLOGY, ISLAMABAD

A Machine Learning Based Hybrid Approach to Classify and Detect Windows Ransomware

by

Sana Aurangzeb

A thesis submitted in partial fulfillment for the
degree of Master of Science

in the

Faculty of Computing
Department of Computer Science

2018

Copyright © 2018 by Sana Aurangzeb

All rights reserved. No part of this thesis may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, by any information storage and retrieval system without the prior written permission of the author.

This effort is dedicated to Allah Almighty in whose hand
is the control of my life.
Then to my respected teacher Dr. Muhammad Aleem who guides me and
helps me in every step of my work
&
My spiritual parents for their support and love.
I also dedicated this study to
my husband for
his endless support and encouragement.

CAPITAL UNIVERSITY OF SCIENCE & TECHNOLOGY
ISLAMABAD

CERTIFICATE OF APPROVAL

A Machine Learning Based Hybrid Approach to Classify and Detect Windows Ransomware

by

Sana Aurangzeb
MCS163008

THESIS EXAMINING COMMITTEE

S. No.	Examiner	Name	Organization
(a)	External Examiner	Dr. Kashif Munir	FAST-NU
(b)	Internal Examiner	Dr. Muhammad Arshad Islam	CUST
(c)	Supervisor	Dr. Muhammad Aleem	CUST

Dr. Muhammad Aleem
Thesis Supervisor
October, 2018

Dr. Nayyer Masood
Head
Dept. of Computer Science
October, 2018

Dr. Muhammad Abdul Qadir
Dean
Faculty of Computing
October, 2018

Author's Declaration

I, **Sana Aurangzeb** hereby state that my MS thesis titled "**A Machine Learning Based Hybrid Approach to Classify and Detect Windows Ransomware**" is my own work and has not been submitted previously by me for taking any degree from Capital University of Science and Technology, Islamabad or anywhere else in the country/abroad.

At any time if my statement is found to be incorrect even after my graduation, the University has the right to withdraw my MS Degree.

(Sana Aurangzeb)

Registration No: MCS163008

Plagiarism Undertaking

I solemnly declare that research work presented in this thesis titled "*A Machine Learning Based Hybrid Approach to Classify and Detect Windows Ransomware*" is solely my research work with no significant contribution from any other person. Small contribution/help wherever taken has been dully acknowledged and that complete thesis has been written by me.

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

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

(Sana Aurangzeb)

Registration No: MCS163008

List of Publications

It is certified that following publication(s) have been made out of the research work that has been carried out for this thesis:-

1. **S. Aurangzeb**, M. Aleem, M. A. Iqbal and M. A. Islam, "Ransomware: A Survey and Trends," *Journal of Information Assurance & Security*, vol. 12(2), pp. 48-58, 2017.

2. **S. Aurangzeb**, M. Aleem, M. A. Islam, Ahmad. U. "On the Classification of Ransomware Using Hardware Performance Counters," *Journal of Computers & Security*, Under Review.

3. **S. Aurangzeb** and M. Aleem. "Proposed Hybrid Framework for the Classification of Windows-Based Ransomware Using Machine Learning," *Turkish Journal of Electrical Engineering & Computer Sciences* 2018, Under Review.

Sana Aurangzeb
(MCS163008)

Acknowledgements

I pay my gratitude to ALLAH (S.W.T) The Creator and Sustainer of all seen and unseen worlds, who gave me power and strength to conduct this study without His help I could never write even a single word. Secondly, I would like to express my sincerest appreciation to my supervisor **Dr. Muhammad Aleem** for his directions, assistance, and guidance. I sincerely thanked for his support, encouragement and whose thoughtful suggestions and offering of time is greatly appreciated. Nobody has been more important to me in the pursuit of this research than my parents, whose love and guidance has been with me throughout every choice I have made. So, I thank to my beloved parents for their continual love, affection, support and encouragement in this journey. No words can ever be sufficient for the gratitude I have for my brothers and for my family. A special thanks to my husband for his support and encouragement to complete this Master of Science degree. I pray to ALLAH (S.W.T) that may He bestows me with true success in all fields in both worlds and shower His blessed knowledge upon me for the betterment of all Muslims and whole Mankind. I thank all those persons who have cooperated with me during my laborious work.

Abstract

Ransomware - a subcategory of malware - is a malicious software and a class of self propagating malware, whose specific goal is to hold victim's data by using encryption techniques, until a ransom is being paid. With the proliferation of internet and computers in any domain, Windows based ransomware is now becoming a great threat that emerging drastically with the passage of time. It installed on a victim's device without the knowledge of the owner and perform malicious activities such as stealing personal information, encrypt data, lock the machine, makes the data inaccessible to the user and demands a ransom amount to be paid in the form of bitcoin or other forms of untraceable currency. With the rise of new malware categories, it has now become difficult to differentiate between an ordinary malware and a ransomware. Therefore, it is essential to analyze the behaviour of ransomware samples to know their malicious nature that differs from other malwares. There are two ways to analyze ransomware. The first type is to analyze statically, which means analyzing the code without executing. The second type is to analyze dynamically, which means observing the behaviour of ransomware by actually executing the application. In most cases, static analysis is not sufficient enough to identify the malicious behaviour between ransomwares and non-ransomware. Due to the shortcoming of static analysis, we proposed two hybrid approaches which is a combination of static analysis, dynamic analysis, and performance counters to classify and detect ransomware using machine learning methods. The machine learning analyzer is used to train both static and dynamic features using different families of ransomware and non-ransomware applications. Our experiment results show that the proposed ransomware classification and detection mechanism is able to classify and detect unknown ransomware that exhibits similar static or dynamic behaviour. Moreover, we find that data from performance counters can be used to classify and identify ransomware.

Contents

List of Figures

List of Tables

Abbreviations

API	Application Programming Interface
AV	Anti-virus
CC	Command & Control
CPU	Central Processing Unit
DLLs	Dynamic-Link Libraries
FN	False Negative
FPR	False Positive Rate
HHRC	Hybrid-Hierarchy Ransomware Classification
HCRC	Hybrid-Combined Ransomware Classification
MD5	Message-Digest Algorithm 5
ML	Machine Learning
NB	Naive Bayes
NRW	Non-Ransomware
RF	Random Forest
PE	Portable Executable
PEiD	Portable Executable Identifier
ROC	Receiver Operating Characteristics
RW	Ransomware
SSDT	System Service Descriptor Table
TN	True Negative
TPR	True Positive Rate
WEKA	Waikato Environment for Knowledge Analysis

Chapter 1

INTRODUCTION

With the increasing number of devices such as laptops and cell phones, the cyber-attacks are emerging drastically that leads the researcher to raise many security questions. Malware stands for malicious software (that used to perform harmful activities), which is installed in users system without his/her knowledge. Malware comes in many forms such as viruses, trojans, worms, and ransomware, etc. Recently a specific form of malware known as ransomware (RW) being the most dangerous malicious software among all. Ransomware is known to be scareware, which is a type of malware that can be a used as a tool for blackmailing user via secretly infecting victims device and later demands a ransom payout in order to decrypt files. Recently, Windows ransomware have become a great threat to the computer and smart device users. Ransomware can be referred as scarewares, which are coded in a way to frighten victims and convince them to quickly purchase the software. A ransomware encrypts the data of infected machines, and asks the user to pay a ransom usually, in Bitcoins [1] to regain full access to the attached system. Many victims pay ransom in order to save their important data for which they do not have any backup.

Ransomware are not only affecting businesses, individuals but public economy exponentially as well as private companys assets [2]. A recent trendy example of 2017 year about WannaCry ransomware attack, where more than 300,000 computer devices were affected in 150+ countries and ransom demands were rendered, which

indicates that these predictions might be beaten in the coming year [3]. Another example is CryptoWall version-3 [4, 5], which caused an estimated $325 million damage in the US alone during the period from November 2015 to June 2016. CryptoWall version 4 reached up to $7.1 million damage globally [4]. Considering Sony ransomware attack [6] that even took US government attention which claims that North Korea was responsible for the damage [7]. In 2016 and 2017, similar ransomware attacks were increasing gradually from the last couple of years. Such type of ransomware makes headlines day-by-day via extracting large amounts from victims ranging from $10,000 to restore a public school districts records to $17,000 to restore patient records at a hospital [8] to $24 million in the year of 2015. The FBI reported in 2016 that the ransomware cost within the 03 months of 2016 in United States growing onwards to an estimated $209 million.

Ransomware usually pass through three phases 1) finding a target to victimize; 2) preventing access to local information; and then 3) displaying some scary or ransom message to get amount from users. According to [5], there are two basic types of ransomware available today, (a) locker-ransomware and, (b) crypto-ransomware. Locker-ransomware family is used to lock the victims machine and ultimately prevent the user from using it. Whereas, the crypto-ransomware, seems most common now-a-days, encrypts personal files to make them inaccessible to its victim. Ransomware attacks have been reported on both the desktop and mobile platforms. Although the mobile ransomware attacks are not that common as compare to desktop ransomware. For the reason that users are more concerned with their data placed on desktop systems than mobiles. For example, the desktop Trojan Kenzero [9] not only steals the users browser-history and also publishes it publicly on the Internet along with the persons name. It typically demands 1500 yen to take down the victims browser history [10]. There have not yet been any mobile malware that seriously threatens or publicly embarrasses the user for profit except for one piece of mobile ransomware that demanded a ransom e.g., a Dutch worm [11] locks iPhone screens and later demands 5 euros to unlock the screens of the infected phone [10, 11]. There might be a behavior difference among users that

makes one platform a more valuable ransom target than the other. Screenshot of Windows ransomware is shown in 1.1.

Windows malware are found to be a great threat throughout years, however, ransomware being the most severe threat among all due to its attacking and demanding nature (i.e., demands ransom in return). Therefore, ransomware classification from other malware is now become mandatory because of its harmful and demanding nature, which affects financially and economically by damaging hospitals, police, and schools record drastically. This work is based on a machine learning approach to classify and differentiate actual ransomware from non-ransomware (NRW) i.e., an ordinary malware, which is not that threatening as ransomware.

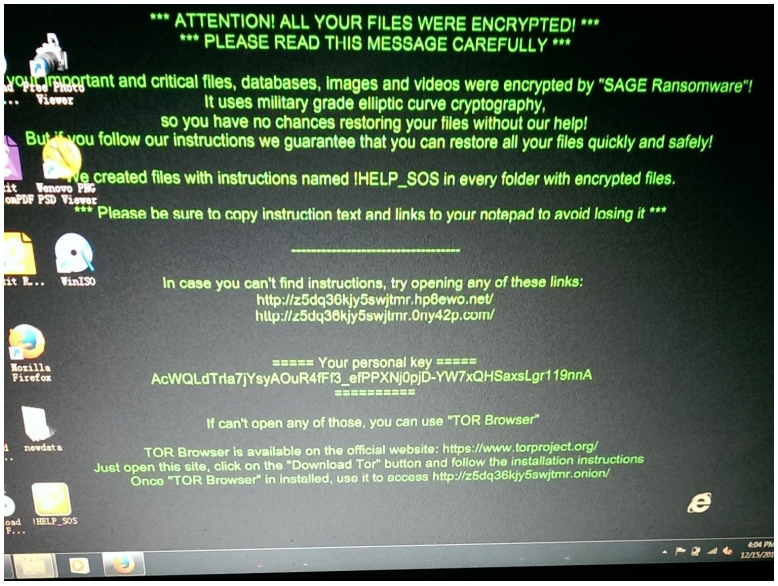


FIGURE 1.1: Screenshot of Windows Ransomware Attack

We propose to use a hybrid classification method consists of a static and dynamic method. Static method which is used for analyzing an applications malicious activities via code without viewing the actual instructions i.e., without execution, and a dynamic analysis starts by actually executing the specific application. Static methods are used to classify ransomware behavior by observing some features e.g., strings, information get through PEview and exploring list of DLLs functions that imported by a piece of malware [12]. Dynamic methods, the second step of our work performed after basic static analysis, considered those features that can only be obtained after executing malware to get the behavior of applications (memory,

CPU, cache-misses, branches, instructions, statistics on API calls, file monitoring operations, registry keys and some other hardware performance counters). Static approaches are less computationally intense than dynamic methods but they are typically less effective in detecting ransomware [7, 13]. Whereas, dynamic analysis are effective in classifying and detecting new malwares. Thus outperforming static techniques, however, dynamic analysis need applications to be in execution mode thus, potentially infecting the device [13]. The motivation behind implementing a hybrid approach is to get the advantages of both static and dynamic techniques by minimizing their disadvantages.

Keeping this in mind, the main contributions of this thesis are as follows:

1. Design and Evaluating - hybrid approach - based on static features (strings, and DLL files) and dynamic features (file system monitoring operations, hardware performance counters, API calls and registry keys) in classifying windows ransomware in section 3.1;

2. Design and Evaluating - hybrid (combined vs hierarchy-based) approach in Section 3.2.

1.1 PURPOSE

The purpose of this study is to look deeper into Windows-based features and provide a new way to classify ransomware. This thesis helps in providing a framework for effective ransomware classification and detection. This framework will be a stepping stone for Antivirus (AV) vendors and malware experts to take these features into an account that have been neglected so far to develop and improve new ransomware detection tools and antiviruses. We proposed a hybrid machine-learning based ransomware classification and detection technique that utilizes several static and dynamic features to increase ransomware detection rate for the Windows platform.

1.2 PROBLEM STATEMENT

A lot of solutions have been developed against malware and ransomware that significantly improved the users security. Previous researches [2, 5, 7, 14–18] has shown that there is a lack of behavioral analysis that use hybrid technique to classify among ransomware and non-ransomwares collectively using these features i.e., API Calls, File operations, Registry keys, and Hardware performance counters (processor usage, cache-misses, memory usage, page faults, instructions, branches etc.). Whereas hardware performance counters were analyzed on malware and benign apps in [17] but authors have not considered it for ransomware classification. Classification of ransomware has not been done previously, taking combination of these multiple features into consideration, which might give us accurate results for classification and detection. These aspects need to be considered:

a. Hardware performance counters such as cache-misses, branches, branch-misses, CPUs utilized, task clocks, cycles, context-switching, CPU migrations, instruction per cycles, and page faults are not observed previously for classification of ransomware.
b. Classification of malware into RW or NRW never been done collectively using these features.
i. File Operations;
ii. API Calls;
iii. Registry Activities;
iv. Performance Counters (disk resources).

1.3 RESEARCH QUESTIONS

The problem discussed in section 1.2 has led us to explore the answers for the following questions:

Research Questions:

1. How to classify known and zero-day malicious apps using hybrid approach into two categories RW or NRW?
2. Which of the static and dynamic features play a vital role in the detection of ransomware from malicious app?
3. Which of the hardware performance counters are more useful for the detection of ransomware?
4. Which classifier plays a role in detecting and classifying ransomware?
5. How malicious application classification rate can be improved by employing two different hybrid frameworks?

1.4 PROPOSED SOLUTION

This thesis proposes two different hybrid techniques that combine the features from both static and dynamic approaches to classify Windows-based ransomware from non-ransomware (i.e., other malicious applications) using single analyzer in one approach and two analyzers in another approach. In this proposed security mechanism, fingerprinting technique is conducted to analyze the malicious behavior of applications statically. File operations, API Calls, registry activities and hardware performance counters are observed during the dynamic testing to observe the runtime behavior of applications. This approach covers the static as well as the dynamic behavior of the applications considering the combination of static analysis and other dynamic features along with Hardware Performance Counter of Windows-based applications that have not been considered in this combination previously.

1.5 SIGNIFICANCE OF THE SOLUTION

The utmost line of defense against ransomware are antivirus products that recognize attacks using the hashing-based technique. However, ransomware writers

actively develop new techniques to evade the existing solutions. Ransomware being the deadliest among other malware therefore, it is essential to develop a new defensive mechanism that is harder to bypass, and are capable of mitigating the novel threats. This proposed mechanism observes vulnerabilities that help to effectively classify and detect ransomware from other malwares that are actually non-ransomware and are less threatening which might help in revealing more security points that need to improve by antivirus solution providers. The proposed technique provides deeper insight into hardware features to provide a framework against new ransomware attacks and tools. The combination of these analyses; hashing, DLLs file static analysis, and dynamic analysis using hardware features along with the behavioral analysis of monitoring files helps to inform future prevention strategies for Windows-based ransomware and improve security solution.

1.6 TOOLS AND TECHNIQUES

Following tools and techniques are used in this work.

1. Host Operating System: Ubuntu (version LTS 14.04) 32 bit;
2. Virtual Environment: VMWare Workstation Pro and Cuckoo sandbox having Windows XP;
3. Machine Learning Classifiers: Random Forest, J48, and Naive Bayes;
4. Perf: for hardware performance counter;
5. Wine: to analyze windows .exe files on Ubuntu;
6. WinMD5Free v1.20: to calculate and display hashes to identify ransomware;
7. PEiD Program (UPX version 0.89.6-1.02): to detect packed files;
8. Dependency Walker: to explore DLLs and functions imported by malwares;
9. PEview: to display the structure and content of the PE

10. Windows Filesystem Minifilter Driver Framework: a standard kernel-based approach to achieve system wide filesystem monitoring in multiple windows;
11. SSDT: for API functions and system calls monitoring;
12. WEKA: for training and testing machine classifiers;
13. VirusTotal: to report where applications were flagged as ransomware [19];
14. Cuckoo Sandbox: to analyze generating reports;
15. Python 2.7

Chapter 2

BACKGROUND AND LITERATURE REVIEW

2.1 INTRODUCTION

Ransomware is a sub category of malware and malware comes in many forms which are explained below:

2.1.1 MALWARE

Malware is a malicious piece of code which not only takes control over a system but steals personal information and tries to damage it. There are many types of malware which threat victims differently, so can be classified into following categorized:

A.PersonalSpyware
Spyware tries to collect personal information such as login credentials, geographical location, browsing habits, license keys or text message history over a particular period of time. With personal spyware, it transmits the stolen information to a remote server and installs the software without the users knowledge to have a physical access to the device [10].

B.Ransomware

Malware that can be a used as a tool for blackmailing is referred as ransomware. Ransomware is malicious software that secretly infects victims device, and suddenly demands a ransom payment in order to decrypt the encrypted data.

In this work, we present a study of different ransomwares features using a hybrid technique. This study of classification and detection of ransomware aims to assist researchers and individual users to counter cybercriminal activities. In our knowledge, there does not exist any recent study related to these ransomwares features that use System calls, API Calls, registry activities and performance counter features collectively in a single study.

C.Backdoors

Backdoor contains malicious instructions and a piece of code that installs itself onto a device to get access from users systems. It allows remote access to an infected host, and let the attacker connect to the device typically bypassing normal authentication and security mechanisms [12].

D.Bots

Bots provide the ability to the attacker to enable remote control of a host. Botnets are used for performing distributed operations that present a dangerous security threat, such as sending spam e-mails, Denial of Service (DoS), mining cryptocurrency, and many more [20].

E.Downloaders

Downloader program is a malicious code that downloads additional malware from remote servers, usually configured by cyber-criminals when they take control over the device and get access to all files and folders.

F.Rootkits

Another type of malware that consists of malicious code designed to pair with other malware code, such as a backdoor, to interfere with security software of the infected device to evade detection.

G.Scareware

Scareware is just a piece of software that uses to spread fear among users and to sell their product, such as, a very well-known scareware is rogue antivirus (AV).

H.Worm or virus

A malicious piece of code capable of copying itself and can infect other devices [12].

2.1.2 RANSOMWARE

Ransomware, a malicious software that locks users system until a ransom amount is being paid off. The first ransomware program was originated in 1989 and was labeled as AIDS Trojan [21]. In the beginning, few cases were reported back in 2005 in Russia and were attacked by its neighboring countries like Belarus, Ukraine, and Kazakhstan [22]. Later on, ransomware spread widely by employing different social engineering techniques and uses more advanced encryption techniques to conceal user data. In 2007, first time Locker-ransomware was appeared in Russia. In 2013 CryptoLocker [23] ransomware was observed that spreads rapidly. FBI reported that CryptoWall causes $18 million from April 2014 to June 2015 [24]. CryptoWall version 3 [4, 5], caused an estimated $325 million whereas CryptoWall version 4 causes $7.1 million damage globally [4]. Ransomware victimization has increased by 3500% more than the fourth quarter of 2015 within the starting period of 2016 [21]. According to CNN News (2016), $209 million dollars were paid within a few months of 2016 and it reaches up to 80% at the end of year [25]. Recently, the new type of ransomware emerges in 2017 named as WannaCry that effects widely over more than 150 countries. [3]. It has been stated that the top six countries impacted by all types of ransomware in 2015 are the United States, Japan, United Kingdom, Italy, Germany, and Russia [26]. In 2015, a report generated from the Cyber Threat Alliance explained the total damage caused by ransomware was $325,000,000 [22].

The most common families of ransomware along with their originality are listed below in Table 2.1 [27]:

TABLE 2.1: Ransomware and their origin years.

Name	**Year**
PCCYBORG Trojan	1989
One Half Virus	1994
Trojan.GPCode	2005
Trojan.Cryzip	2006
Locker ransomware	2007
GPcode.AK	2008
Citadel toolkit	2012
Reveton	2012
CryptoLocker	2013
CryptoWall	2014
CryptoDefense	2014
PoshCoder	2014
Virlock	2014
TeslaCrypt	2015
CryptoFortress	2015
CryptoTorLocker2015	2015
CTB-Locker	2015
CryptoWall	2016
Xorist	2016
JavaScript-only ransomware-as-a-service	2016
Filecoder	2017
Petya	2017
JAFF	2017
WannaCrypt	2017

2.1.3 HOW RANSOMWARE WORKS?

Ransomware is a longlasting problem and Windows ransomware are increasing quantitatively and qualitatively. In general, all ransomwares go through similar stages. A ransomware usually attacks the victims device by some means of user activity such as; user either open an email received from an unknown source or have a visit on malicious websites, clicked on email attachments, or press a malicious web-link, or spread through other infected devices. The system gets infected after the ransomware acquires administrative privileges. It then tries to contact Command & Control server in order to steal victims information and send back to the attacker. It also locks the user device and demands victims to pay ransom in the form of untraceable payment methods like Bitcoins by changing the desktop background wallpaper or by generating a new folder on a desktop which shows payment methods to regain system access as shown in figure 2.1.

Ransomware thus searches for different files and folders to start encrypting all types of files(.doc, .xlsx, .pdf, .jpeg, .png, etc) by generating symmetric key from the command & control server using asymmetric (RSA) encryption algorithm. RSA algorithm uses two keys, public key for encryption and private key for decrypting

FIGURE 2.1: Screenshot of Windows Ransomware Payment

data [28]. Meanwhile, ransomware removes all the backup points, backup folders, and shadow volume copies [20].

2.2 LITERATURE REVIEW

Numerous techniques have been suggested to identify the vulnerability of ransomware and these techniques based on either static or dynamic analysis helps in classifying and detect ransomware with the help of some tools and techniques.

One of the crucial and complicating factors in ransomware static analysis is the emerging of new programs that modify an executable file to hide its data and the actual program logic that can only be found from reverse engineering techniques. Those programs that can alter other program files to compress their data and all contents are known to be as "executable packers" or just "packers". Considering pattern-matching based approaches, the program Portable Executable (PE) Identifier (PEiD) [29] found out as a widely used tool for detecting binaries that exhibit unpack-execute behavior. This type of static analysis helps in finding cryptographic algorithms in PE files, and access the capability to extract the information to another tool that is IDA pro. [5] suggests that static based detection technique as used by [30] can help in evading AV, usually at the start of ransomware.

String is another helpful tool used in static analysis for analyzing files having ASCII and Unicode strings in binary data. With the help of strings, the analyzer can get a quick overview of malware capacity and ability. Few strings are embedded that can be extracted from executable files using a wide variety of tools.

Another way to analyze statically is to examine the lists of dynamically linked functions in an executable with the help of Dependency Walker program [12].

Both methods static and dynamic have their own benefits and limitations. Following is the literature review of some state-of-the-art techniques in ransomware analysis.

2.2.1 STATIC ANALYSIS

In [31], authors elaborate that detection of malware using static-based analysis without executing in order to find out malware's malicious activities. If the analysis configures out any malicious code, the executable will be stopped from launching. Authors presented a case study of CryptoLuck ransomware which was first seen in 2016, to highlight the importance of behavioral based ransomware detection. They find out that the goodate.dll was located in its current directory which means that CryptoLuck hijacks Google DLL with its own DLL file.

In [32], authors provide an experimental analysis on windows and android platform by selected ransomware variants from existing ransomware families. Mainly focusing on their characterization. They selected 17 Windows and 08 Android ransomware families for analysis. Authors compare variants of ransomware with each other and they find out with the analysis that ransomware variants show similar characteristics, however, they show different ways of payload. They used reverse Engineering process to analyze ransomware. Features used for the Windows ransomware are file system activities, registry activities, device control and communication, network activity and locking mechanism. For detection in windows operating system, they have checked MD5 checksum values for each analyzed sample and checked it against antivirus search engines. The experimental results revealed that Windows platform ransomware detection is possible by observing abnormal filesystem and registry activities.

In [33], authors observed targeted malware by applying considering various features n-gram, byte sequence, opcode, and portable executable header and apply machine learning methods. The major problem faced by authors in using machine learning for analysis is the lack of training dataset. Due to the shortcomings of static analysis, it is alone not sufficient enough to analyze malware. Therefore, researchers prefer to do either dynamic analysis or hybrid analysis for ransomware detection.

[19] authors proposed R-PackDroid a supervised machine learning based 10 fold cross-validation technique to detect Android ransomware statically from benign and other malware applications using API packages. The proposed approach is a lightweight technique because it does not require prior knowledge of ransomwares encryption mechanisms. However, the proposed approach cannot analyze applications having feature code required to be dynamically loaded at runtime or that are fully encrypted. Authors explained that the proposed approach can be integrated with other dynamic analysis techniques.

In [34], the effective ransomware prevention technique was performed statically using process monitoring. The proposed technique is based on three modules: Configuration module, monitoring module and processes module on file events that occurred when the ransomware accesses and copies files using statistical methods on processor usage, and I/O rates. The hash information method is used for detection of Cryptolocker type ransomware. Random Forest classifier come up with good results, however, employing unsupervised methods such as clustering revealed unsatisfactory results.

In [5], authors suggest that static based detection technique as used by [30] can help in evading AV especially only in the first phases of new ransomware campaign.

2.2.2 DYNAMIC ANALYSIS

Dynamic analysis technique is extremely powerful in a way that it is performed by executing any malware application prior to investigation. Dynamic analysis usually performed after basic static analysis has reached a dead end, whether due to obfuscation or packing [12]. Dynamic analysis let you monitor ransomware applications by executing it in the controlled environment. Unlike static analysis, dynamic analysis unveils the real functionality that might be not possible to be observed by implementing only static approaches [12].

In [27], authors performed ransomware behavioral analysis on windows platform of 14 strains of ransomware and perform a comparison among ransomware API calls

with baselines of normal operating system behavior. They observed the individual behavioral pattern of ransomware and clean applications. They identify the system activities and events that were executed during each Win/32 baseline operation. The results showed that ransomware used a small subset of all system calls logged during normal baseline operations.

In [35], authors present an automated detection and analysis of ransomware based on dynamic ransomware detection system using data mining techniques like RF, SVM, SL, and NB. They monitor dynamic behavior by generating API calls flow graph CFG and generate feature sets to distinguish between benign and ransomware. The results show that the proposed methodology can improve in ransomware detection using SL algorithm that achieved 98.2% detection rate and 1.2% false positive rate.

Authors in [7], perform a dynamic analysis system which they named as 'UNVEIL', designed specifically for the detection of ransomware by automatically generating an artificial user environment. Later, detect ransomware when it interacts with user data. Authors keep a record of the changes found in the systems desktop that indicate ransomware like behavior i.e., locking Desktop. They claim that UNVEIL is capable enough to identify previously unknown ransomware that was not detected by the antimalware industry. Their system is implemented through custom Windows kernel drivers that allow monitoring filesystem, I/O Data Buffer Entropy used for every read and write request to a file captured in an I/O trace. They performed a long-term study analyzing 148,223 recent general malware samples in the wild.

In [18] authors present an automated detection and analysis of ransomware based on dynamic ransomware detection system by producing behavioral logs with Cuckoo sandbox. They only considered pre-encryption features for application to be ransomware. Their experimental results revealed that TF-IDF gives better results as far as WannaCry is concerned. Authors validate that the method can be able to extract features to differentiate malware using logs.

There are several other research efforts which are based on a machine learning approach to detect ransomware exploiting the dynamic or runtime features of executing applications. In [36], evaluation of machine learning classification was performed on malware. Authors observed network traffics and extract its features while filtering out TCP packets and trained it using five classifiers (Bayes, RF, KNN, J48, and MLP) for evaluation.

Another proposed study of dynamic analysis of ransomware through monitoring file system activity of windows platform was conducted by [5]. They used a classification technique to classify goodware and ransomware using three machine learning classifiers (EldeRan, SVM, and NB). Authors present EldeRan, which they refer as a machine learning approach for dynamically analyzing and classifying ransomware from clean ones. They considered a wide range of features such as Windows API calls, Registry Key Operations, File System Operations, file operations performed per File Extension, Directory Operations, Dropped Files, and Strings. Feature selection was performed using MI criterion. Their evaluation shows that EldeRan achieves ROC curve of 0.995 whereas achieving a detection rate of 96.3%supporting dynamic analysis for ransomware detection. Authors concluded that registry key and API calls are the two classes with most relevant features with the Logistic Regression that outperforms Naive Bayes.

Hardware performance counters are typically used by programmers to measure the performance of the under investigation software with the aim of improving it on a target platform [37]. In [17] uses a dynamic approach to classify hardware malware based on their performance counters. They applied machine classifiers that were KNN and Decision tree with the result of 90% accuracy and 3% FP.

Performance counters represent the true execution behaviors of the application. However, none of the existing machine learning ransomware detection techniques uses hardware performance counter for ransomware classification. Malware can employ obfuscation techniques to deceive antivirus that relies on static analysis; however, a malware cannot hide its malicious intent that is hidden in the hardware

performance counter pattern. Therefore, we argue that hardware performance counters can act as a vital source for differentiating malware and ransomware.

This thesis focuses on the use of hardware performance counters to analyze the runtime behavior and detect ransomware in order to find out how accurately can hardware performance counters are able to classify malware and ransomware. We argue that ransomware are inclined to reveal in the form of hardware performance counter similar detectable dynamic features (e.g., Page Faults, branches, branch-misses, instructions etc.). Multiple machine learning classifier, i.e., Decision Tree(J48), Random Forest, and Naive Bayes are evaluated using the performance counters, in an order to get the answer of the research questions mentioned above in section 1.3.

2.3 CRITICAL ANALYSIS

After a comprehensive study of the state-of-the-art techniques for Windows ransomware analysis, we summaries strengths and weaknesses of the current approaches as shown in Table 2.2:

TABLE 2.2: Literature Survey of Different Methodologies

Ref	Methodology	Strengths	Weaknesses
[36]	Filter TCP packets, extract network traffic features, labeled data Use 5 ML classifers (Bayes, Random Forest, KNN, J48, MLP)	Perform a good comparative analysis between 5 ML and with other papers	No classification between malicious applications For android only
[32]	Present the life cycle of Windows based Ransomware using basic static and basic dynamic analysis MD5 method, Cuckoo Sandbox used Analyze Filesystems, registry activities and network operations	Explains in detail working and functionality of RW and malicious applications PEiD tool is used for windows RW detection	No ML No classifier Lack of experimental results
[18]	Dynamic behavioral analysis of wannacry RW	Study helps in further manual analysis of logs	Considered only one RW i.e., wannaCry

	Present a method to extract features of RW from hosts logs and analysis pattern generation Monitors files, folders, memory dumps, network traffic, processes, API calls via Cuckoo	Accurately presents pattern generation logs and dynamic analysis	No ML classifier Very basic and trivial results
[7]	Dynamic long term study to Monitors file system, I/O activity Use Windows File system Mini filter Driver framework Detect screen lock on windows platform	Successfully detect 1 unknown ranswomware Submitted to VirusTotal 96.3% TP rate and 0% FPs	Based on kernel level activities only Risk RW run at kernel level may stop to monitor some files
[5]	Dynamic analysis to Classify between goodware and ransomware using Machine Learning Monitors file system activity on windows platform Features used: API calls, Registry Key, File System Operations, Directory operations, Strings	Come up with automated tool to analyze new software and evaluated the performance of the 3 ML classifiers (EldeRan, SVM, and NB) Concluded that registry key and API calls are important features Can enhance the detection capabilities of AV software	The experimental setup for RW analysis was not natural (commonly used apps were not installed in VM) Initial dataset was larger Unable to analyze RW that show silent behavior, or wait for a user to do something
[27]	Analyzed 14 strains of windows ransomware. Compared API calls through RW processes with baselines of normal operating system behaviour.	Concluded that API calls having helpful feature	Considered only single feature i.e., API calls Ignore all other features
[38]	Analyzed 15 ransomware families Presented detailed analysis of payment methods and the use of Bitcoin.	Detailed evolution-based study from 2006-2014 Proposed high-level mitigation strategies such as the use of decoy resources to detect suspicious file access.	Proposed only general methodology
[17]	Android Malware detect with performance counters using a Dynamic approach	A major support is that runtime behavior can be captured using HW performance counters are essential to detect malware	No static analysis and Implement only with malware dataset, not ransomware

	Applied ML classifiers (KNN, Decision tree)		Ignores other important features
[35]	Automated Dynamic approach using ML techniques RF, SVM, SL and NB Monitors API calls flow graph CFG using k fold cross validation Distinguish between benign and ransomware	Proposed methodology improved RW detection rate SL algorithm achieved 98.2% detection rate and 1.2% false positive rate	Extract only the features of 168 software samples for estimating. Considered only API features
[34]	Statically analyze process monitoring and I/O rates features. The Proposed technique is based on three modules: Configuration module, monitoring module and processes module on file events	The hash information method is used for detection of Cryptolocker type ransomware. Random Forest reveal good results	Only for android platfom Results for unsupervised approach (clustering) were not satisfactory

It has been observed from the literature work that most of the techniques [5] can either only observe system calls, or API calls [18, 19, 27, 35] or file operations [7], processor usage [34], or registry activities [32]. Some research work is based on static analysis [31] whereas other proposed techniques that mainly focuses on dynamic analysis for classification. There exists no such work that considers all these important aspects in a single methodology that could improve ransomware detection rates. Therefore, we propose a ransomware analysis methodology to combine all these features(i.e., cache-misses, branches, branch-misses, CPUs utilized, task clocks, cycles, context-switching, CPU migrations, instruction per cycles, page faults, statistics on API calls, file monitoring operations, strings, list of DLL functions imported and registry keys) to classify ransomware from non-ransomware so that ransomware classification accuracy can be improved. In addition to that, no other existing approach analyze hardware execution profile for the classification of ransomware.

Chapter 3

METHODOLOGY

In this chapter, we cover the methodology of our proposed framework, details of a dataset, features extraction process using static and dynamic analysis and training of the machine learning classifiers. We have performed our experimental analysis by proposing two methods: Hybrid Hierarchy-based Ransomware Classification (HHRC) and Hybrid-Combined Ransomware Classification (HCRC)) analysis as explained in Section 3.1 and Section 3.2, followed by the training phase of HHRC analyzer in Section 3.1.1. Specification of data collection in Section 3.1.2, static feature extraction in Section 3.1.3 whereas dynamic feature extraction is explained under Section 3.1.3. Training phase of HCRC is explained under Section 3.2.1.

3.1 PROPOSED HYBRID HIERARCHY RANSOMWARE CLASSIFICATION (HHRC) ANALYZER

Presenting the overall methodology used for the proposed hybrid analyzer known to be as Hybrid Hierarchy-based Ransomware Classification (HHRC) analyzer. HHRC analyzer is a combination of static analysis feature vector separately trained on machine learning classifiers. After the output, all those applications which are

labeled as ransomware in static analysis machine learning analyzer are considered to be ransomware as a final output. Therefore, such feature vectors are eliminated in dynamic analysis. Thus, reducing computational time. We called it HHRC analyzer due to the hierarchy it followed as explained in Figure 3.1. Therefore, in dynamic analysis, only those applications are trained on a machine learning classifier that are either mispredicted or not predicted in static phase in order to give output as ransomware or non-ransomware.

The proposed hybrid Windows ransomware classification and detection approach based on machine learning classifiers as explained in Figure 3.1 starts with a set of malware consisting of ransomware and non-ransomware. The first phase is known as static analysis performed by analyzing applications in a virtual environment using the hashing technique. All the malicious applications execute via hashing programs that produce a unique hash that identifies that malware (a sort of fingerprint), winMD5 is used to calculate that hash values [12]. We called it process 1 (P1). Figure 3.2 and 3.3 shows hashing process. This hash value can be used for online search if the file has already being identified or not by any AV. These applications are then further analyzed in process 2 (P2) to make sure either they are packed or obfuscated. PEiD can be used to detect such type of packer or compiler employed to build an application, which makes analyzing the packed file much easier [12]. Next, we explore applications hierarchical tree diagram of functions and DLLs and embedded strings. The most useful piece of data that we can able to find out at run-time are the list of functions that the executables imports. Above all the linking methods, dynamic linking is the most common and the most interesting for malware analysts [12], where libraries are dynamically linked, then the host Operating System searches for the relevant libraries after the application is loaded. When the application calls the linked library function, that function executes within the library [12]. This can be possible using DependencyWalker tool.

These features then act as feature vectors that are being used to train machine learning algorithms. Once algorithms are trained, these trained algorithms can

classify the given applications into ransomware and non-ransomware only if P1 and P2 both labeled it as ransomware. The application is labeled to be as ransomware as output. This whole process is referred to be as phase 1. All those applications which are identified as ransomware in phase 1 are eliminated.

Applications which are labeled only as non-ransomware are then further analyzed dynamically in phase 2 which are then executed under a controlled environment using Cuckoo sandbox that records the dynamic behavior of the applications and produces recorded activity logs as output. Different dynamic features (such as registry key operations, API calls, file operations, and hardware features) are extracted to produce feature vectors. These vectors are used to train different machine learning classifiers. Once these algorithms are trained, they can classify applications into ransomware and non-ransomware on the basis of dynamic features. Detailed analysis of our proposed framework of dynamic analysis is as follows:

1. Those applications which are labeled as non-ransomware in static analysis phase 1 are again checked dynamically under a controlled environment using Cuckoo sandbox that records the dynamic behavior such as API calls, Registry keys and behavioral analysis of the applications are produced and recorded activity logs as output.

2. Hardware behavior of executed applications are analyzed on Ubuntu environment 14.04, an operating system via perf tool to analyze the run-time behavior. Wine is used to run Windows-based executable files on ubuntu whereas SSDT is used to extract other file system monitoring features. Perf [39] command of Ubuntu is used to collect hardware features during execution of the application.

3. The feature vector input along with a label ransomware/non-ransomware (i.e., application category as 1/0) is used to train machine learning algorithms (J48, Random Forest, and NB). The reason for using these classifiers

is that they are well-known in the detection of ransomware and are used throughout years by experts in analyzing malicious software [5, 17, 34–36].

4. After the training of classifiers, they are used to classify applications into ransomware and non-ransomware in the testing phase as shown in Figure 3.5 which explains the training phase of the analyzer which employs 50% ransomware and 50% non-ransomware applications.

3.1.1 TRAINING PHASE OF HHRC ANALYZER

The training phase of HHRC analyzer starts by analyzing a set of malware consists of 50% ransomware and 50% non-ransomware applications that are analyzed under a controlled environment. The features are then extracted by employing tools and techniques as mentioned in section 1.6. These feature vectors along with their category (ransomware labeled as 1 and non-ransomware labeled as 0) are then given as input to machine learning static analyzer. Once these algorithms are trained, they can classify applications into ransomware and non-ransomware on the basis of static features as shown in Figure 3.4

The second phase of HHRC analyzer starts by analyzing applications dynamically under virtual environment to get sets of feature vectors. These feature vectors along with their category (ransomware labeled as 1 and non-ransomware labeled as 0) are then given as input to machine learning dynamic analyzer. Once these algorithms are trained, they can classify applications into ransomware and non-ransomware on the basis of dynamic features as shown in Figure 3.5

There are few important steps to achieve the objective of training machine learning algorithms for ransomware detection. These steps are discussed in the forthcoming sections.

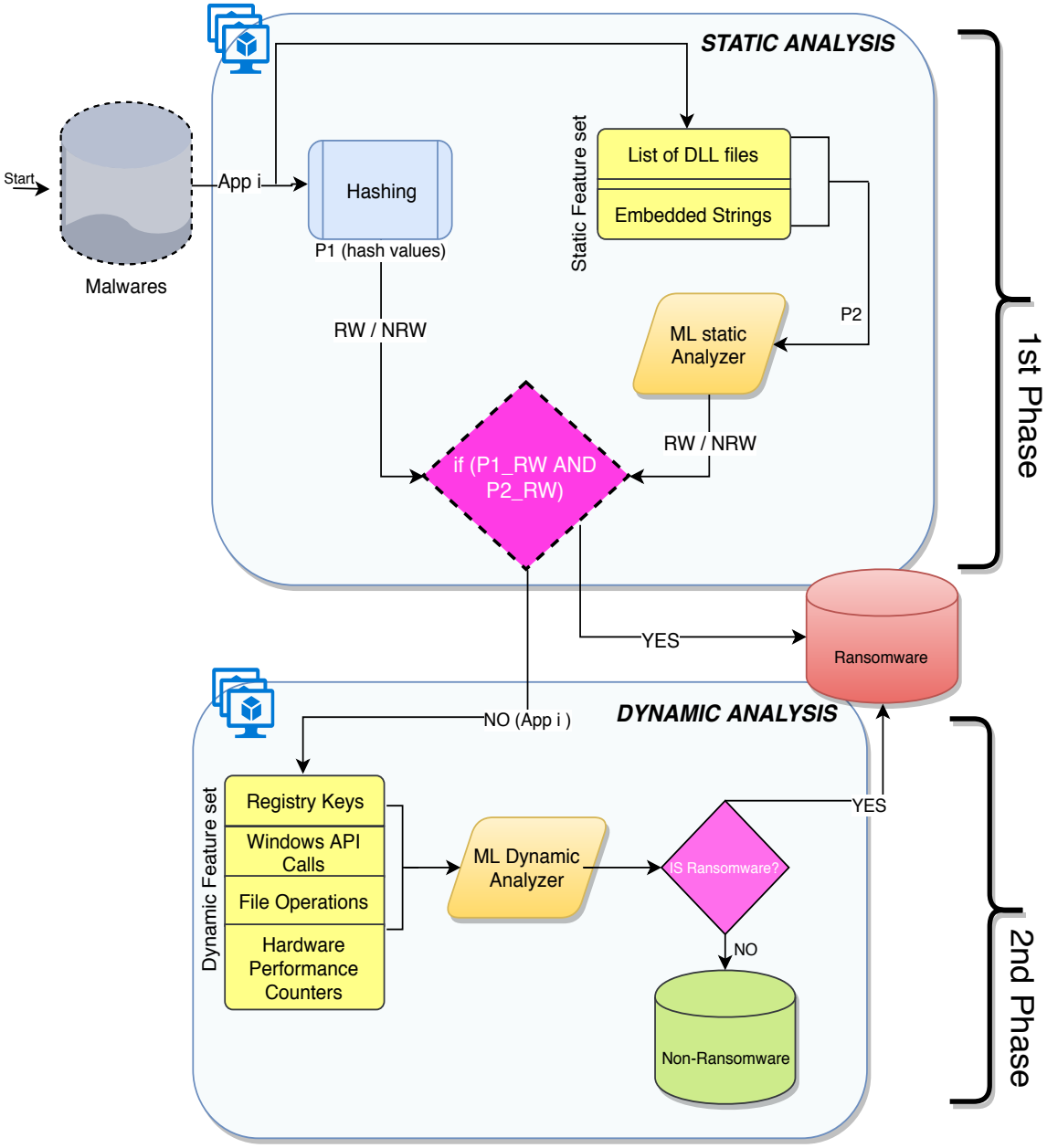


FIGURE 3.1: HHRC Methodology

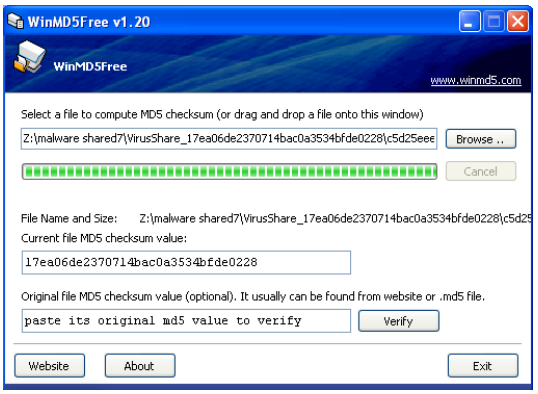


FIGURE 3.2: Produces hash value.

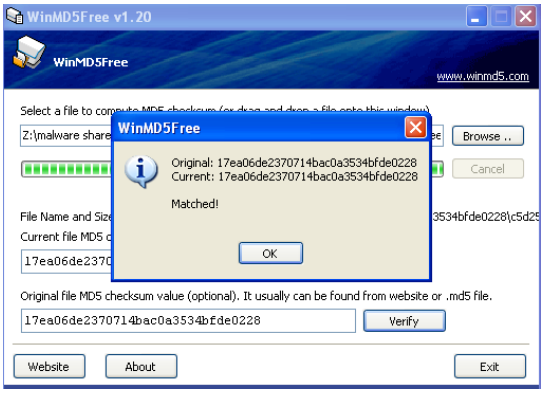


FIGURE 3.3: Screenshot shows matched md5 values.

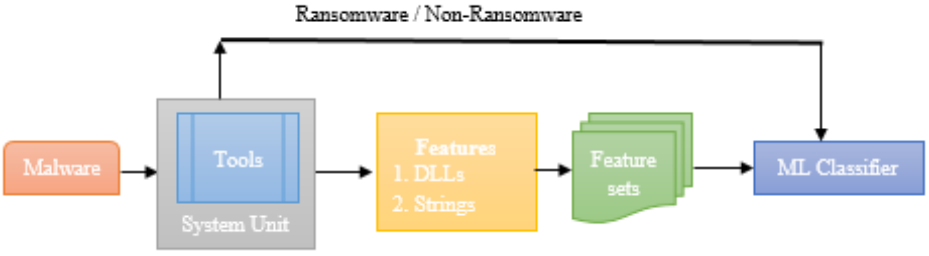


FIGURE 3.4: Workflow of training classifier for HHRC analyzer.

3.1.2 DATA COLLECTION

For the sake of experimentation, we collected our dataset comprises of 500 ransomware and non-ransomware executables (that are not clean apps but other malware) randomly downloaded from Virusshare.com [40] and from [5] dataset based on several families. Virusshare is an archive of malware samples to provide researchers, digital forensic experts, and interest groups the access to samples of malicious code. All the collected dataset was labeled by top-performing antivirus

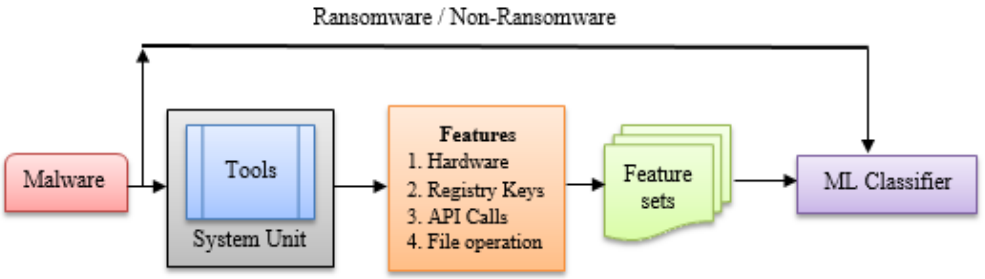


FIGURE 3.5: Workflow of training classifier for dynamic analyzer.

according to the definition of a malware i.e., a malicious code that may be a ransomware or non-ransomware program. The employed classifier was trained using the behavioral features for both the ransomware and non-ransomware with an explicit labeling (i.e., Ransomware /Non-Ransomware). For training and validation, a disjoint data set is used. Table 3.1 gives a brief overview of the dataset used for the experimentation for this research.

TABLE 3.1: Details of Dataset Used

Application Type	Label	No of Applications used for experimentation
Ransomware	1	250
Non-Ransomware	0	250

3.1.3 EXTRACTION OF STATIC FEATURES

During static analysis, we extracted Windows ransomware by analyzing applications in a virtual environment using the hashing technique, PEiD, PEView, Strings and DependencyWalker tool to extract all the features statically. Strings are another useful part to collect some basic information as far as static analysis is concerned. A string in a program is a sequence of characters such as "the, url, .dll." A program contains strings in case it print out a message that join with a URL, or can move a file to a specific location. Utilizing strings throughout the program can be a simple way to get hints about the functionality of a particular program [12].

The information gathered through static analysis then act as feature vectors. We extracted 1713 features by performing static analysis. These feature vectors along with the application category label (ransomware, non-ransomware) are used to train machine learning algorithms. Once algorithms are trained, these trained algorithms can then classify the given applications into ransomware and non-ransomware.

3.1.4 EXTRACTION OF DYNAMIC FEATURES

Dynamic analyzer starts with the second phase of our proposed hybrid analysis, which is to analyze the behavior of those suspicious applications that are either mispredicted or not predicted at static analyzer phase 1. Dynamic analysis performed by executing the applications in a controlled environment such as Cuckoo sandbox [41].

It is quite efficient to detect the malicious application behaviors that remain unidentified during static analysis. In earlier work, a lot of attention has been paid to study the API calls during dynamic analysis. Nevertheless, there are other techniques that can lead to an efficient and less expensive dynamic ransomware detection, for example, considering only single feature. In our proposed framework, we explore multiple dynamic features (API calls, Registry operations, Files operations, and hardware features etc) that are less expensive but are more helpful in ransomware detection. Cuckoo sandbox [41] is selected in a Linux platform for automated dynamic analysis of Windows executable ransomware because of its an open source software. It automatically runs and analyze files and collect comprehensive analysis results that outline what the ransomware does while running inside an isolated operating system. All processes and file changes are tracked and logged.

The first step is to configure cuckoo sandbox in a controlled environment in order to run executable applications. The dataset is in the form of .exe files is located

in the host Operating System (Ubuntu 14.04 LTS). Cuckoo is used to run an application for automated dynamic analysis. Figure 3.6(a) and 3.6(b) shows the execution of an application within the Cuckoo sandbox.

Cuckoo output log is saved for each monitored application in the host Operating system. This is a lightweight format used fort data-interchange and easily interpreted by the humans.

(a)

(b)

FIGURE 3.6: Execution of Cuckoo

The information gathered through dynamic analysis then act as feature vectors. We extracted 10985 total features by performing dynamic analysis. These feature vectors along with the application category label (ransomware, non-ransomware) are used to train machine learning algorithms. Once algorithms are trained, these trained algorithms can then classify the given applications into ransomware and non-ransomware.

Generated logs and behavioral analysis reports are recorded by Cuckoo as shown in Figure 3.7, 3.8 and 3.9.

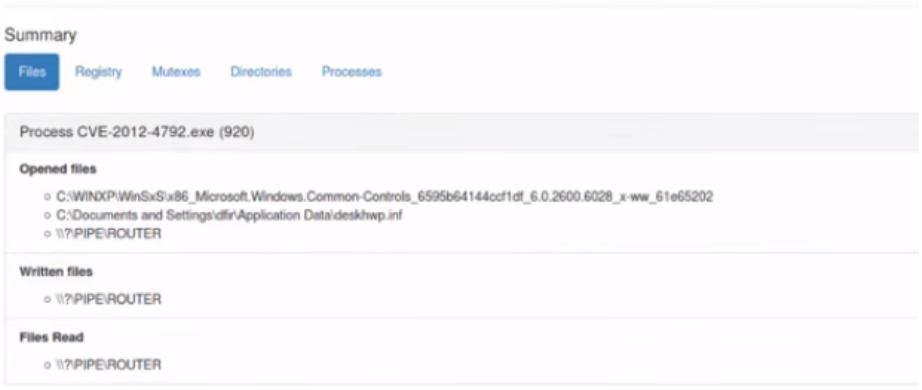


FIGURE 3.7: Summary of the executed malware

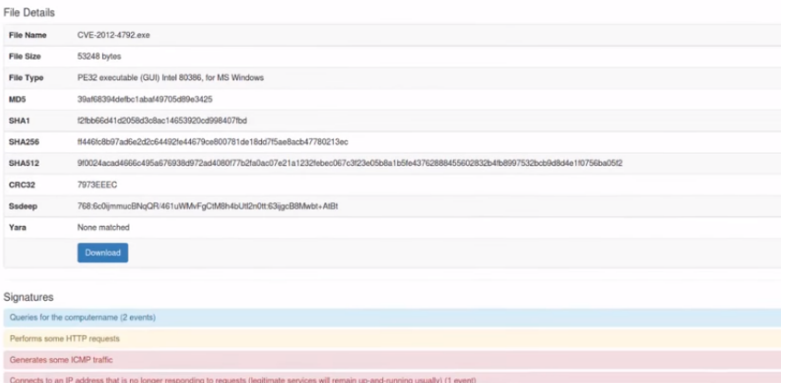


FIGURE 3.8: Details of the files executed

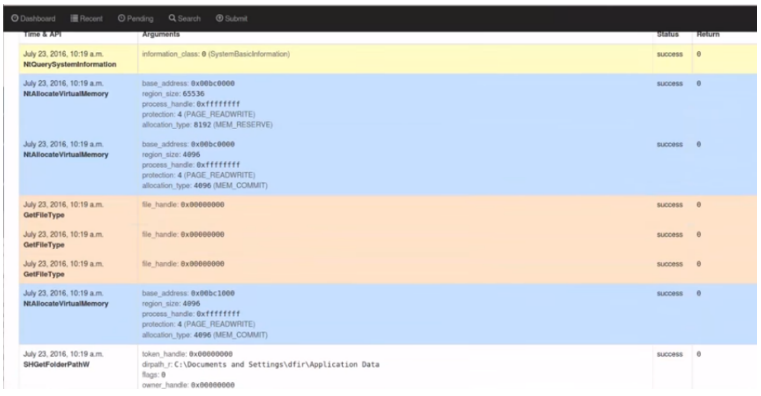


FIGURE 3.9: Behavioral Analysis of the files executed

The dynamic features of an application show a big picture of control flow and data flow information about the application. This information is used to feature vectors for the training of machine learning algorithms.

Table 3.2 shows some of the features we have collected during the dynamic analysis of applications. These features are based upon the events that are generated during execution of the application in a virtual environment.

TABLE 3.2: List of Few Collected Dynamic Features

API Calls	Registry Activites	File Monitoring	Hardware
GetSystemDirectoryA	DELETE:DHKEY _LOCAL_MACHINE \SYSTEM \ControlSet001 \Services\pfmfs _640smx	DELETED:C: \Program Files \WinCalendar V4\	cache misses
WriteConsoleW	DELETE:DHKEY _CURRENT _USER \Software \APN PIP \SFFZ	DELETED:C: \Program Files \ChatSend Toolbar\tbunsy3.tmp\	task Clock
NtReadVirtualMemory	DELETE:DHKEY _CURRENT _USER\Software \Microsoft \Windows \CurrentVersion \Run\	DELETED:C: \WINDOWS\Temp\	CPUs Utilized
RemoveDirectoryA	DELETE:DHKEY _LOCAL_MACHINE \SOFTWARE \Classes \.bz\	DELETED:C: \Documents and Settings\MyUser \Application Data\Microsoft\	context Switches
GetKeyState	DELETE:DHKEY _LOCAL_MACHINE \SOFTWARE \Classes\.r03\	OPENED:C: \Documents and Settings\All Users\Application Data\	CPU Migrations
FindFirstFileExA	DELETE:DHKEY _CLASSES _ROOT\	OPENED:C: \Program Files\DebugMode\	page Faults
NtQueryKey	OPENED:HKEY _LOCAL_MACHINE \Software \Google \Picasa\	OPENED:c: \d5a412dfc95bd9972 eaf87bf6a49\update\	cycles
HttpOpenRequestA	OPENED:HKEY _CURRENT_USER \Software\d18656c1d 1d6e4d5de54423754 2640205da94d30308 2605b5cbdc5ca60d4 ab37\	OPENED:C: \Program Files\MSECache \wordview\	instructions
HttpSendRequestA	OPENED:HKEY _CURRENT_USER \Software \Local AppWizard-Generated Applications\StuDormMS\	READ:c:\Documents and Settings\MyUser\Favorites\	branches

GetUserNameA	OPENED:HKEY _LOCAL_MACHINE \SOFTWARE \Classes \CLSID\7b8a2d94-0ac9-11d1-896c-00c04Fb6bfc4\	READ:C:\Documents and Settings\MyUser\Local Settings\Temp\is-80O4H.tmp\	branchMisses
HttpOpenRequestW	READ:HKEY _LOCAL_MACHINE \SYSTEM\ControlSet001 \Services\Messenger\	WRITTEN:C: \Program Files \ImproveSpeedPC\	seconds Time Elapsed
WriteConsoleA	READ:HKEY _CURRENT_USER \Software\Microsoft \Windows\CurrentVersion \Internet Settings \5.0\Cache\Cookies\	WRITTEN:c:\a6a468 1e30cf844c1dc05d21 580a99\pkg\	
NtOpenFile	READ:HKEY_LOCAL _MACHINE\SOFTWARE \Classes\.wma\	WRITTEN:c:\b56 a85006c8424892 58be9e8cbda \mousekeyboardcenter \setup64\files\1033 \rtf \mouse \gaming_mice\	
NtCreateProcessEx	READ:HKEY_LOCAL _MACHINE\SOFTWARE \Microsoft\Windows NT\CurrentVersion \Image File Execution Options\photohse.EXE\		
GetSystemInfo	WRITTEN:HKEY _CURRENT_USER \Software\5ab2c353\		
NetShareEnum	WRITTEN:HKEY _CURRENT_USER \Software\HuluDesktop\		
FindWindowExW	WRITTEN:HKEY _CURRENT_USER \Software\WinRAR \Setup\.uu\		

3.1.5 FEATURE VECTOR

The choice of a good feature set is the initial phase of any data mining approach. Some of them were inspired by previous work [5], but we added our features too i.e., Hardware performance counters, DLLs and strings.

Let V be a vector comprising of a set of 500 ransomware applications. For every application at *ith* location in the dataset, we produce a binary sequence

Vi= s1, s2...sj and

sj= 1, if *jth* feature exists.

0, otherwise

The recognized features are then arranged as a sequence of 0 or 1 in a comma separated list. The presence of a specific feature is denoted by 1 and the absence is denoted by 0 in the list.

Following sequence represents an example of the feature vector for ransomware applications and non-ransomware applications:

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

Multiple features as shown above in Table 3.2 obtained during dynamic analysis are used to generate the binary feature vector. These features are also used to train

the dynamic analyzer (machine learning algorithm) that identifies the malware sample as ransomware and non-ransomware in the given dataset.

3.1.6 FEATURE SELECTION

Due to a large number of attributes in the dataset, there is a need to reduce those attributes in order to find the most relevant among to solve predictive modeling problem. A large number of features in the dataset should be filtered and refined. In addition, some features may contain redundant information from other features. That lead towards increasing computational cost and reduce accuracy.

The aim of this step is to moderate the high-dimension of each feature instance in our collected dataset by introducing subsets of features. Subset features are helpful in predicting class label accurately. Features selection helps in improving the accuracy of the classification models and reducing time complexity. Additionally, feature selection minimizes the factors of overfitting; the time required for training/testing and increases the accuracy to generate simple interpreted models.

For this purpose, we have used info gain method to get the features list playing a most significant role in the classification important features were selected out of 1713 total 25 produced by static analysis. These features consist of PEview info, DLL and string collectively. Therefore, following the recommendation of [42], the information gain criterion [43] will be used in this thesis to select a subset of features, that finds the most appropriate features by assigning weights to the information to emphasize the effectiveness of the features.

A specific method called *InfoGainAttributeEval* from Weka machine learning tool was applied to attribute selection. We selected *25* features out of 1713 after applying feature selection algorithm based on the static results and selected *47* features out of 10985 based on dynamic results.

The information gain of an attribute C on sample data S can be calculated as below:

$InfoGain(C, rj) = entropy(C) - entropy(C \mid rj)$

Figure 3.10 describes the feature selection of top-ranked static features (ranked by the algorithm) and plays a key role in ransomware identification in a dataset. It has been seen clearly that static analysis features reach maximum range to 0.1285 which is not that important as we required. The top feature is SizeofStackReverse attained 0.1285 scores, then MajorOperatingSystemVersion achieving 0.11983, and then Subsystem which remains on 0.09727 as explained in Appendix B that shows the brief description of these top-ranked features obtained through feature selection method.

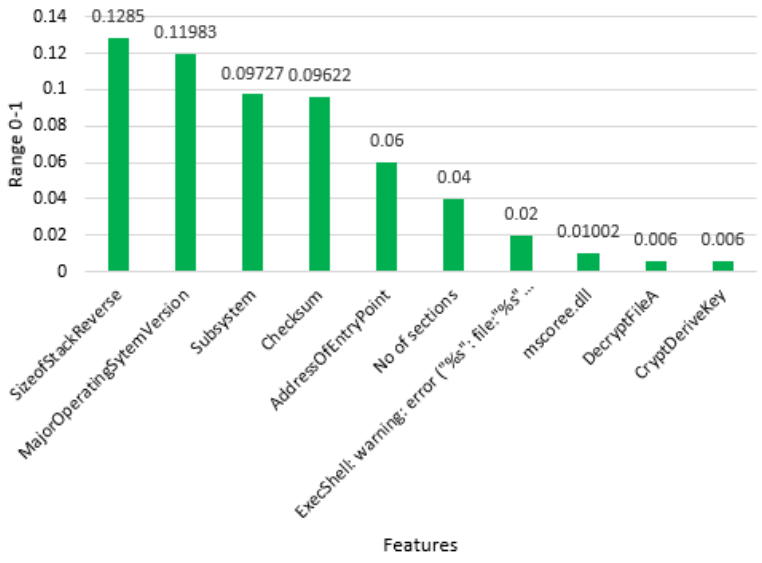


FIGURE 3.10: Rankwise Static Feature Selection using InfoGain method

The features ranked as top-ranked from dynamic data using Info Gain method are shown in Figure 3.11 with their corresponding scores respectively. 50 Features were produced by dynamic analysis out of 10985 whereas 47 features were selected for HHRC. The figure depicts that the highest range attained through info gain method is 0.8 which is cache-misses, then branches achieved 0.73, instructions get

a score of 0.71, and pagefaults is on 0.65. These are the hardware features that achieved the highest score during the feature selection method.

The registry key operations also play a role during feature selection by gaining 0.6, 0.59, and 0.314. Whereas API calls CreateDirectoryW attained 0.414 and NtTerminateProcess is on last of top-ten ranked features gets 0.2 scores.

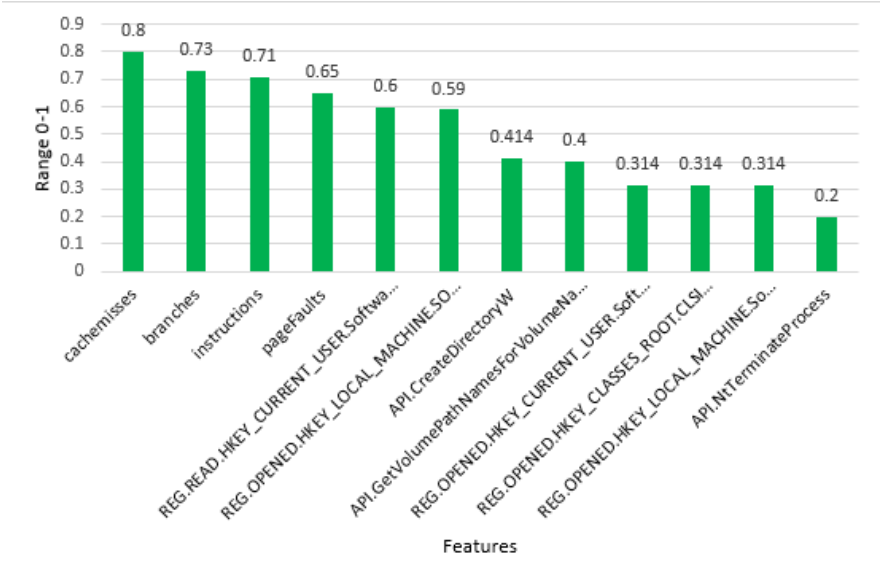


FIGURE 3.11: Rankwise Dynamic Feature Selection using InfoGain method

The features ranked by Info Gain method in dynamic phase is explained in Appendix B along with their description. It has been seen that hardware features and registry key operations play a role in the classification and detection of ransomware.

3.2 PROPOSED HYBRID COMBINED (HCRC) ANALYZER

The second methodology that we have tried to perform in an experiment is to analyze the performance of single classifier instead of two separate machine learning classifiers. The experiment is based on training a single classifier using features

obtained after static analysis separately and collecting dynamic features separately. Then both features were combined to make a feature set, we refer it as Hybrid-Combined Ransomware Classification (HCRC) analyzer. These feature vectors than given as input to a machine learning HCRC analyzer for training and validating data. Machine learning algorithm will decide the applications to be ransomware or non-ransomware.

Figure 3.12 shows the methodology of training HCRC analyzer. It shows that applications are given as input and static analysis (strings, DLLs and PEView information) is performed separately on all those applications. Later dynamic analysis is performed on all those applications to extract features such as Hardware features, Registry keys, file monitoring operations, and API calls. These features are then given as input to a machine learning classifier to train analyzer. This combined analyzer is then tested on applications for ransomware classification and detection. Results of this are depicted in Chapter 4.

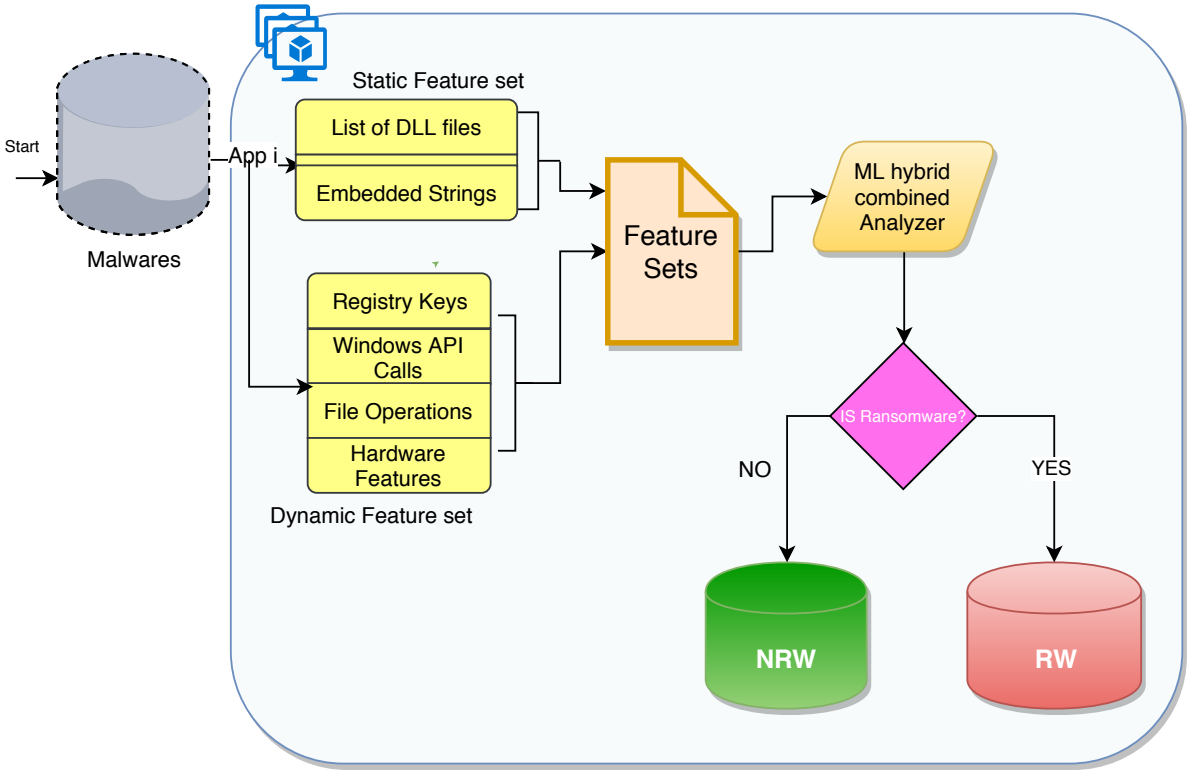


FIGURE 3.12: Methodology of HCRC analyzer

3.2.1 TRAINING PHASE OF HCRC ANALYZER

We further study the impact of combined consideration of both the static and dynamic features for ransomware classification and detection simultaneously. We collected static and dynamic features to find out which features are effective and plays a role in the classification and detection of ransomware. Moreover, how they can strengthen the ransomware analysis together or separately.

Training phase starts by collecting features from both statically and dynamically on all applications. The collected features collectively along with their corresponding labels (ransomware or non-ransomware) are provided to train the machine learning based application analyzer as shown in Figure 3.13.

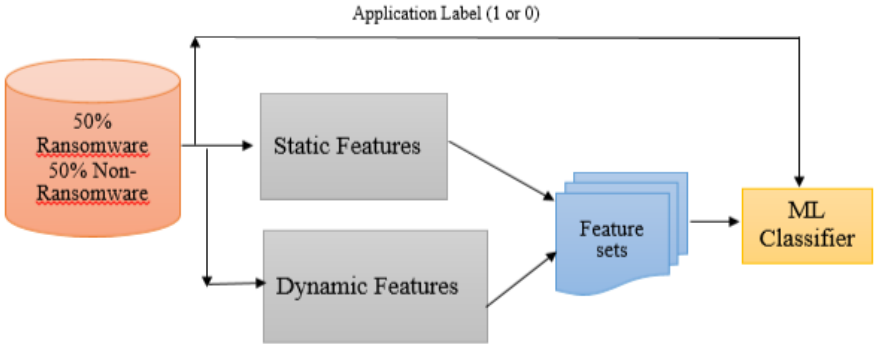


FIGURE 3.13: Workflow of training HCRC analyzer

3.3 CLASSIFIERS USED FOR TRAINING PHASES

Selection of correct classifier for training phase is the most crucial phase in our work. Previous studies [5, 17, 34–36] suggests different classifier on the basis of results they yield. Therefore, we have selected three classifiers for our training phase that are: Naive Bayes (NB) [44], Random Forest (RF) [45], Decision Tree (J48) [46] . The reason of using these classifiers is that for numerical data, choices are too many - starting from basic decision trees, Naive Bayes, SVM etc whereas for categorical data Naive Bayes, decision trees and their ensembles including Random

forest, are good techniques [47]. The data we have used considers both numerical values& categorical values. Since, we have class labels, therefore, the problem is called classification problem.So, one option is to go with decision trees, other possibilities are naive Bayes where you model numeric attributes by a Gaussian distribution or else Random Forest that combines bagging and random subspace. With mixed data, choices are limited and therefore need to be cautious and creative with different choices [47].

We performed a 10-fold cross validation using the whole dataset whereas 80% of the dataset is used for training purpose and 20% of the dataset is used for testing purpose.

3.3.1 NAIVE BAYES

The Naive Bayes Classifier [48] is expected to perform well, being relatively simple to implement and has good detection rates [49]. NB is used to be included in our work, because of its success being mentioned in the malware classification studies [50] from many years. We have compared NB performance with other classifiers such as J48 and RF.

Bayes theorem used to follow a method of calculating the posterior probability, P(c |r), from P(c), P(r), and P(r |c). Naive Bayes classifier suggests that the value of a predictor (r) on a given class (c) is independent of the values of other attributes. This assumption is called class conditional independence.

$$P(c|r) = \frac{P(r|c)P(c)}{P(r)}$$

$$P(c|R) = P(r_1|c) * P(r_2|c) * ... * P(r_n|c) * P(c)$$

- P(c |r) is the posterior probability of target class given attribute.
- P(c) is the prior probability of class.
- P(r |c) is the likelihood which is the probability of attribute given class.

- P(r) is the prior probability of attribute.

3.3.2 RANDOM FOREST

Random Forest (RF) is a classifier that used multiple decision tree predictors in a way that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest [51]. From the studies, its been observed that RF has excellent and very high accuracy seemed so far among current classifier algorithms [52] because of its execution effectively on huge datasets and ultimately come up with an estimation of what variables are important in the classification. This algorithm generates an internal unbiased estimate of the error as the forest building progresses [51]. Random Forest is capable of making the assumption of missing values in data effectively whereby maintaining accuracy when a large proportion of the data are missing [52].

3.3.3 DECISION TREE (J48)

We have used another machine learning predictive classifier known to be as Decision Tree or J48 that was employed in WEKA, in order to utilize the concept of information entropy to make tree and targets the value of a new sample based on different attribute values of the particular dataset. Dataset is used to split them into smaller features of subsets dependent (the attribute to be predicted) and independent (attribute that helps predict the value of the dependent variable) [36]. Later, the normalized information gain is calculated. The feature with highest information gain is selected for decision making [52]. According to [53] it gives very high detection rates.

Algorithms Basic Steps:

Steps of J48 algorithm are: [54]

(i) If the instances belong to the same class then the tree represents a leaf so the leaf is returned with a label of particular.

(ii) Using the attribute from test data, information is then calculated against every feature. After that infogain algorithm is implemented to get the result from test data.
(iii) In the end, the best feature is found on the basis of the present selection criterion and that feature selected for branching.

3.3.4 SUPPORT VECTOR MACHINE (SVM)

Support Vector Machine is effective in cases where number of dimensions are greater than number of samples. According to [17] categorization of classifier could be done as: linear and non-linear. Linear algorithms tries to draw an optimal hyperplane class X can be pointed as support vectors placed on one side of the hyperplane and class Y referred as points on the opposite side of the hyperplane as illustrated in Figure 3.14. Separating classes via n-dimensional data points, where n is considered as the total number of features. Points on the line are referred as support vectors. Non-linear classifiers however, have no such restrictions; any operation to derive a classification can be applied [17]. We choose to focus on linear algorithms in this work to see either SVM is capable enough to classify ransomware and other malicious programs. Is it be detectable? Does SVM do proper classification as we expect in other cases somehow? The classifier is trained by giving labeled data, the algorithm outputs an optimal hyperplane. After setting up some parameters, we train the SVM to build SVM model to do our predictions. However, the results were not as satisfactory as we were expecting. The reason behind such results were due to dataset not aligned according to SVM, as SVM is suitable for numeric data. Therefore, we haven't proceeded SVM for further analysis and no results were being reported.

3.4 EVALUATION MEASUREMENTS

For performance evaluation of different classifiers, we use the following metrics.

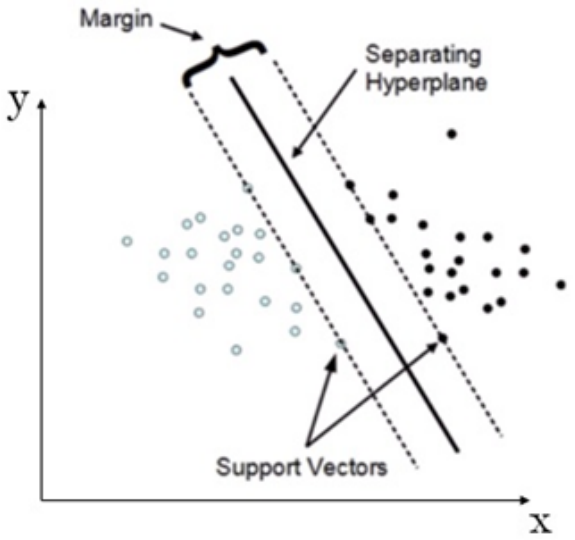


FIGURE 3.14: Illustration of hyperplane [55]

ROC curve: Over the past few years, Receiver Operating Characteristic (ROC) curves [56] are now used widely by many researchers to measure performance which is quite obvious [57]. Furthermore, ROC curves are being used in many statistical methods that merge many clues, test results etc., and have been plotted and evaluated qualitatively. ROC is basically a plot where True Positive Rate (TPR) is on Y-axis and False Positive Rate (FPR) is plotted on X-axis to calculate performance. For every possible classification, TPR rate relies on the scenario where the actual classification is positive and how often the classifier has predicted as positive. The FPR relies on that when the actual classification is negative how often the classifier incorrectly predicted positive. Both the TPR and FPR range between 0 - 1. The area covered below the ROC curve is known as Area Under the ROC Curve (AUC), is widely utilized for weighing classifier performance [36]. The AUC values represent different levels and many research efforts classified into Table 3.3

Accuracy: We have used accuracy for evaluating results which is the fraction of the total number of correctly classified applications as ransomware or non-ransomware

TABLE 3.3: AUC Levels

Levels	Definition
1.0	excellent prediction
0.9	great prediction
0.8	good prediction
0.7	better prediction
0.6	poor prediction
0.5	random prediction
< 0.5	unreliable prediction

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{3.1}$$

Precision: Precision denotes the proportion of the predicted correctly classified applications to the total of all applications that are correctly real positives.

$$Precision = \frac{TP}{TP + FP}, \tag{3.2}$$

Recall: is the fraction of the actual apps that are classified correctly to the total number of the apps that are classified correctly or incorrectly

$$Recall = \frac{TP}{TP + FN}, \tag{3.3}$$

F-Measure: The harmonic mean of precision and recall. F measure represents the value that tells how much the model is capable of making fine distinctions.

$$FMeasure = 2 \times \frac{Precision * Recall}{Precision + Recall}, \tag{3.4}$$

Confusion Matrix: Indicates the number of correctly and incorrectly classified applications. For our dataset, there were two types of data i.e ransomware and

non-ransomware. The confusion matrix shows the results in the following terms:

True Positive (TP): It's the scenario, where the actual classification is positive and how often the classifier has predicted as positive i.e., the count of the ransomware applications that are correctly classified as ransomware.

False Negative (FN): It relies on where the actual classification is negative and how often the classifier incorrectly predicted positive i.e., the count of the ransomware applications that are incorrectly classified as non-ransomware.

True Negative(TN): This indicates that the count of the non-ransomware applications that are truly classified as non-ransomware.

False Positive (FP): It indicates the count of all those non-ransomware applications that were incorrectly classified as ransomware.

Chapter 4

RESULTS

In this chapter, we evaluate our proposed framework for machine learning based ransomware detection and classification system. We have used improved hybrid analysis (HHRC and HCRC) to prepare the dataset for the training of machine learning classifiers. The aim of this section is to discuss the preparation of dataset using improved hybrid analysis, dataset, experimental setup and performance of the proposed framework by evaluating obtained results.

4.1 EXPERIMENTAL SETUP

We carried out our experiments on a machine with specification shown in Table 4.1 below.

4.2 DATASET

Dataset used in our experiments consists of two categories of application sets. Ransomware applications that were collected from Virusshare.com [40] and [5] dataset. The ransomware dataset consists of 250 executables belongs to different

TABLE 4.1: System Configuration

CPU	Intel core 2 duo 2.13GHz
System Type	32 bit
OS	Ubuntu 14.04 LTS
Data Mining Tool	WEKA 3.8
Platform	Windows XP
RAM	3GB
Sandbox	Cuckoo sandbox
Virtual Machine	VMWare

ransomware families. Appendix A indicates all the hash values of ransomware used in our dataset.

Non-ransomware are collected from virusshare.com consists of different malware families, it consists of 250 executables. See Appendix A for the hash values of non-ransomware used in our dataset.

We setup two types of experimental environment (static and dynamic) each for analysis, details of both experimental setups can be seen in Chapter 3, 3.1 and 3.2. The features extracted from both analyses are recorded. Features count obtained as result of static analysis is 1713 where 25 were selected after feature selection method as explained in section 3.1.6; whereas dynamic features count is 10987 and 47 were selected at the end.

4.3 CLASSIFICATION

In our work, we have used three classifiers to differentiate between ransomware and non-ransomware applications. These classifiers are Naive Bayes [44], Random Forest [45] and Decision Tree(J48) [46]. Classifiers are built using data whose category is provided as label i.e., ransomware or non-ransomware.

Cross-validation is a statistical method of assessing and comparing machine learning algorithms. It is implemented by splitting the data into two equal portions: one portion used to train a model and the other portion is used to test the model. One form that used widely in cross-validation is k-fold cross-validation [58]. For every classifier, we consider its optimal parameters. This optimal parameters produces better classification accuracy whilst considering a *k-fold cross-validation* [58].In this validation form, the data is arbitrarily divided into k equally sized folds.

Subsequently, a classifier is trained on k-1 iterations for learning purpose while the rest of the data are used for validation purpose such that within each iteration a different fold of the data is held out for validation. This experiment is performed k *times* repeatedly, considering a different part for testing each time. After these k experiments, the weighted average of classification accuracy indicates the appropriateness of the parameters of a classifier. Finally, we select the algorithm for classification that yields best results in terms of accuracy and whose ROC curve (discussed in Section 3.4) have a better value.

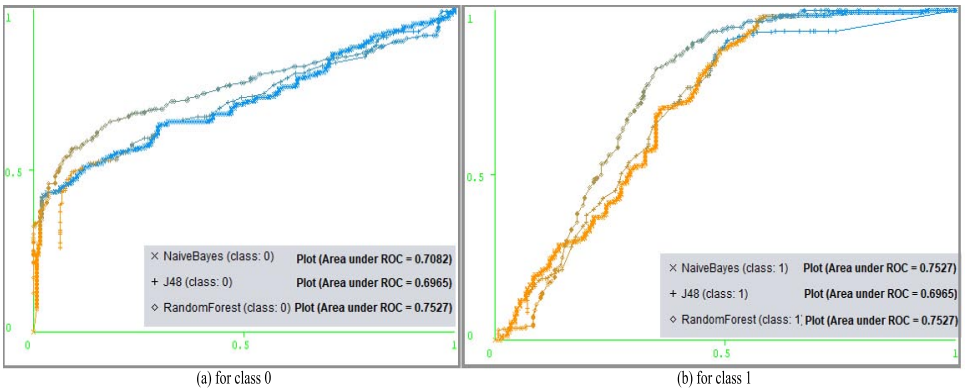
4.4 CLASSIFIERS EVALUATION

Figure 4.1 shows the performance of classifiers on static features through ROC curves. ROC curves are explained clearly under section 3.4. Interpretation and comparisons of ROC curves is easy among dissimilar data sets. ROC plots True Positive Ratio (TPR) against the False Positive Ratio (FPR) for different thresholds of the data with different classification methods. Moreover, it shows that any escalation in TPR goes along with a decrease in the FPR. AUC is used to evaluate the performance of a classifier and is commonly applied for model comparison. Optimal AUC value is 1.0 which means a good performance and classification.

Figure 4.1 shows the ROC curves for static data (i.e., strings, DLLs and PEView info). It indicates the ROC curve values of all three classifiers i.e., RF, NB and J48 in both classes (0 for non-ransomware and 1 for ransomware). Y-axis indicates

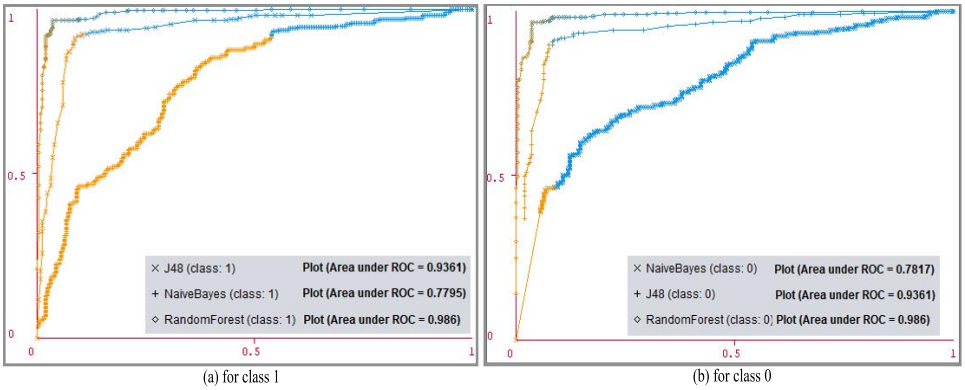
TPR and x-axis refers to FPR. The more the line closer to the y-axis specifies that the classifier model produces the best results. Its clear from the Figure that Random Forest produces ROC curve for both classes 0 and 1 as 0.7527 for static analysis whereas NB produces the same results as Random Forest for class 1 which is 0.7527 but shows less ROC curve for class 0 that is 0.7082.

In Figure 4.2 the detection rate for ransomware is significant as the TPR is high and FPR is low. The figure indicates that the Area under the ROC curve (AUC) value is 0.986 that yields good ransomware classification and detection under dynamic analysis for both classes 0 and 1, produced by Random Forest. Whereas we have seen that J48 produces the same result 0.9361 for both classes and NB gives not satisfactory results which are 0.7795 for class 1 and 0.7817 for class 0.



(a) for class 0 (b) for class 1

FIGURE 4.1: ROC curve for Static Data with classifiers



(a) for class 1 (b) for class 0

FIGURE 4.2: ROC curve for Dynamic Data with classifiers

Figure 4.3 shows that the AUC value of HHRC for all the three classifiers. RF is prominent as compared to other classifiers which produce ROC curve 0.9816

against both classes. Moreover, we can see that NB gives results as 0.791 for class 0 and 0.7879 for class 1 similar to dynamic analysis. J48 performs better produces 0.9817 for classes 0 and 1. Consequently, the AUC for the Random Forest is prominent as compared to others produces AUC value as 0.98 that indicates the best performance by the Random Forest classification technique. If we compare the performance of the same classifiers on the whole data set, we find that the AUC value of Random Forest for dynamic features is highest while for static features this value is 0.75 which is quite low. The J48 shows nearly similar values that indicates both classifiers perform well for ransomware identification. The Naive Bayes has the lower AUC values comparatively.

We have performed hybrid analysis by implementing two different strategies i.e., HHRC and HCRC as explained in section 3.1 and 3.2. Therefore Figure 4.4 shows the results of the performance of HCRC by employing all the three classification algorithms in WEKA using the AUC values. The goal of this test is to compare both the techniques. HCRC produces ROC curve value for Random Forest as 0.9873 whereas it was 0.9816 in HHRC which is not that different. J48 yields 0.934 whereas NB gives 0.78 AUC values. Both techniques come up with similar results.

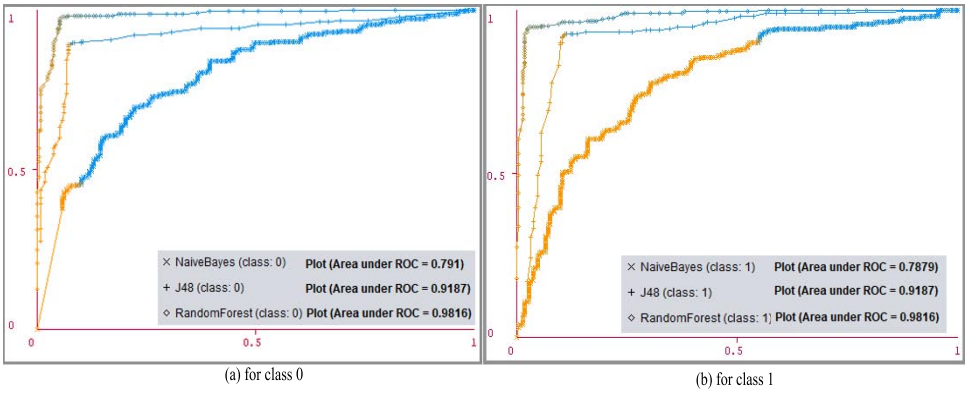
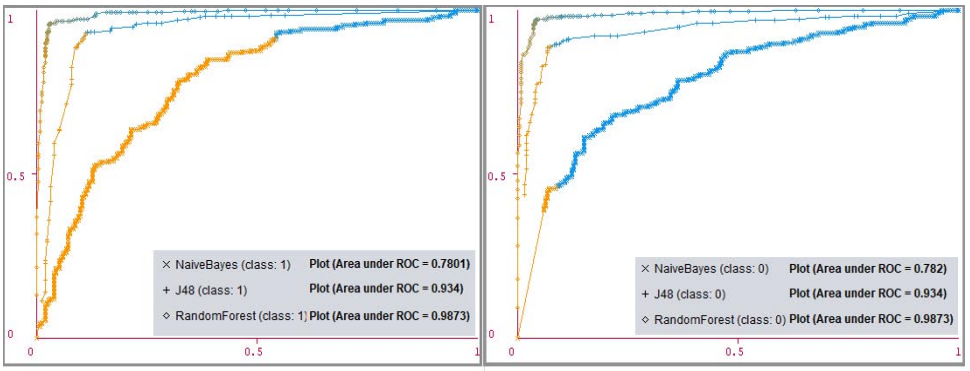


FIGURE 4.3: ROC curve for HHRC Data with classifiers

We have applied three different classifiers to test their performance against ransomware data. Figure 4.5 shows the results of different classifiers with one another to check the selection of accurate machine learning algorithms. From the results



(a) for class 1 (b) for class 0

FIGURE 4.4: ROC curve for HCRC Data with classifiers

of this experiment, we can see that the Random Forest algorithm gives an AUC value closer to optimum value 1 on static data.

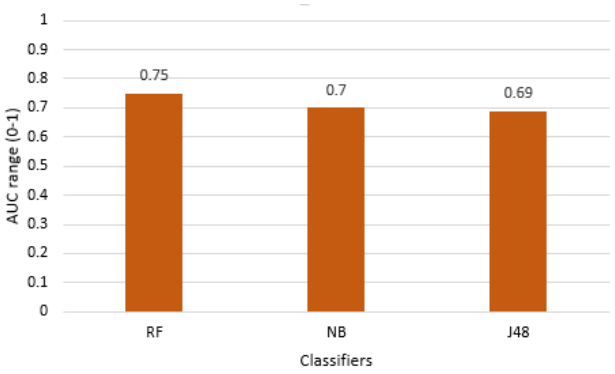


FIGURE 4.5: Performance of the classifiers on static data

Figure 4.6 and 4.7 shows the comparison of three classifiers on both type of hybrid data. The comparison result shows that Random Forest performance is at its peak producing 0.981 results as discussed above on both types of data while the J48 is second best producing AUC value 0.91 which is good prediction whereas NB produces low results of 0.79 prediction which is a better prediction as explained in section 3.4. When applying HCRC approach it come up with 3.08% improvement while comparing with HHRC generating 0.987 indicating that HCRC is somehow better approach than HHRC.

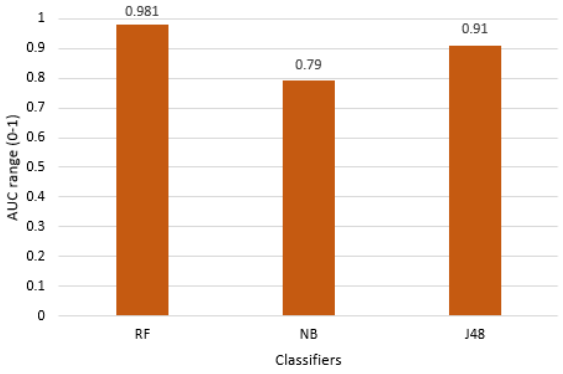


FIGURE 4.6: Performance of the classifiers on HHRC data

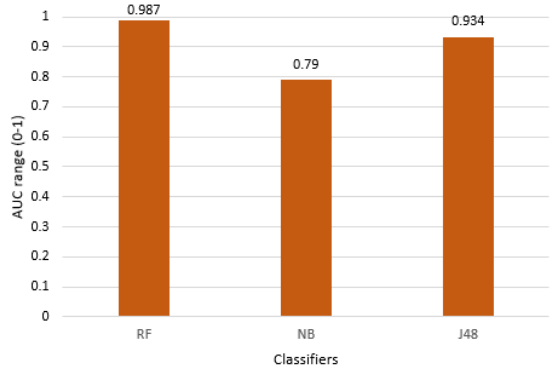


FIGURE 4.7: Performance of the classifiers on HCRC data

4.5 CLASSIFICATION RESULTS

It has been clearly seen from the above mentioned section that Random Forest seemed to be a best classifier among others in case of ransomware classification and detection. Now come the confusion matrices for all the three classifiers that are shown in Table 4.2, Table 4.3 and Table 4.4. These Tables show confusion matrix for static, HHRC and HCRC analyses. Predicted classes of Random Forest shows better results in all tables.

Based on the values of the True Positives (TP) and the Ture Negative (TN) from above-mentioned matrices, we have calculated precision, recall and f-measure for the static, HHRC and HCRC data respectively.

Figure 4.8 shows the effects of the dataset shuffling on the f-measure during the

TABLE 4.2: Confusion Matrix for static analysis

Classifiers		**Predicted Class**	
	Actual Class	**Non-Ransomware**	**Ransomware**
Random Forest	non-ransomware	165	86
	ransomware	51	199
Naive Bayes	non-ransomware	99	152
	ransomware	5	245
Decision Tree (J48)	non-ransomware	134	117
	ransomware	49	201

TABLE 4.3: Confusion Matrix for HHRC analysis

Classifiers		**Predicted Class**	
	Actual Class	**Non-Ransomware**	**Ransomware**
Random Forest	non-ransomware	205	5
	ransomware	12	195
Naive Bayes	non-ransomware	95	115
	ransomware	20	187
Decision Tree (J48)	non-ransomware	187	23
	ransomware	15	192

training and testing. The results of the precision and recall of classification using different classifiers are given below. Results showed that NB generated 7.4% improvement in precision as compared to RF and 15.1% improvement when compared with J48 during static phase. Figure 4.8 shows the recall values of RF is better i.e., 0.727 generating 5.8% and 8.6% improvement in recall as compared to NB and J48. Considering the values of F-Measure that shows 10.3% and 9.3% improvement when RF is compared against J48 and NB. The values of F-Measure for RF, J48, and NB are 0.725, 0.663, and 0.657 respectively.

TABLE 4.4: Confusion Matrix for HCRC analysis

Classifiers		**Predicted Class**	
	Actual Class	**Non-Ransomware**	**Ransomware**
Random Forest	non-ransomware	240	10
	ransomware	10	240
Naive Bayes	non-ransomware	115	135
	ransomware	228	22
Decision Tree (J48)	non-ransomware	222	28
	ransomware	18	232

Considering the data of HHRC technique, the results of which are shown in Figure 4.9. It shows the results of the precision, recall, and F-measure of classification using different classifiers. Results show that RF generated 32.7% and 5.5% improvement in precision as compared to NB and J48. The values of precision for RF, NB, and J48 are 0.96, 0.723, and 0.91 respectively. Figure 4.8 shows the recall values of RF, NB and J48 that are 0.959, 0.676, and 0.909 respectively. Recall of RF is better among generating 41.86% and 5.5% improvement as compared to NB and J48. Considering the values of F-Measure that shows 45.5% and 5.5%

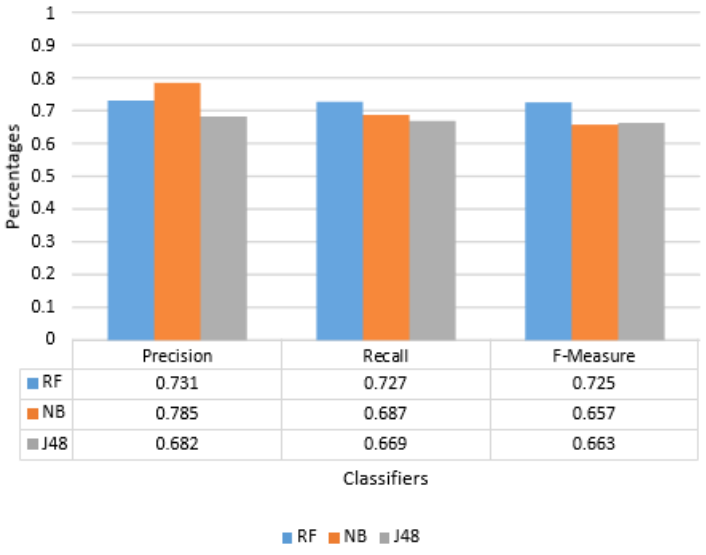


FIGURE 4.8: Precision, recall and F-measure of static data

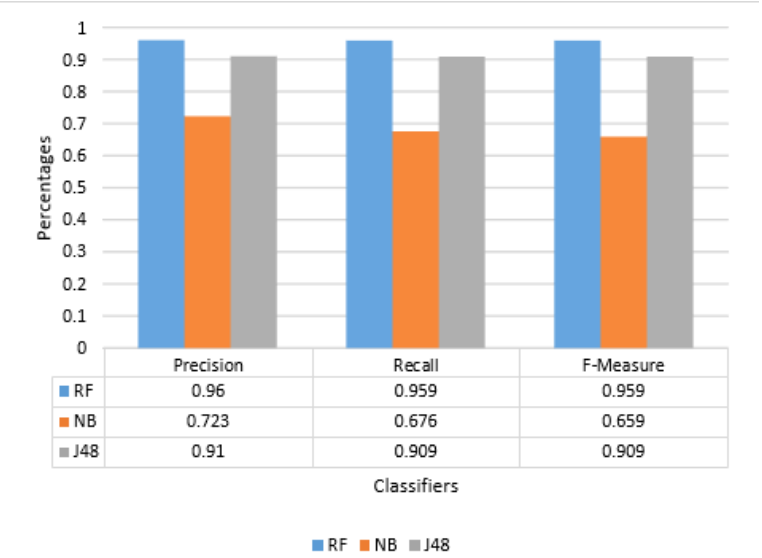


FIGURE 4.9: Precision, recall and F-measure of HHRC data

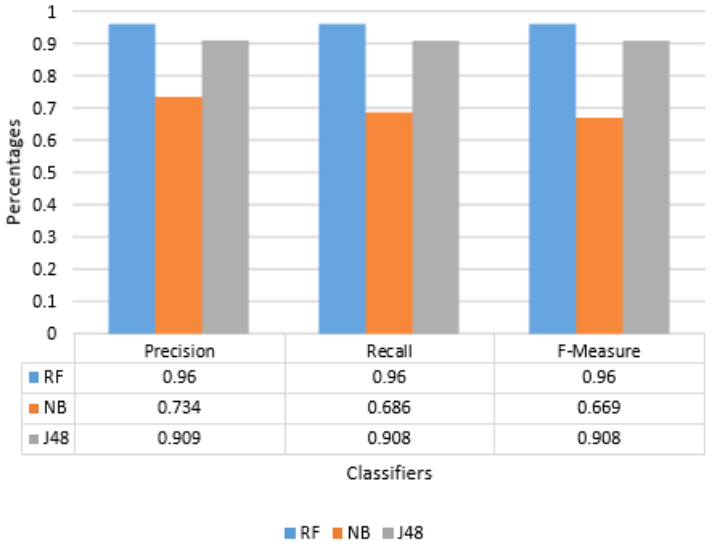


FIGURE 4.10: Precision, recall and F-measure of HCRC data

improvement when RF is compared against J48 and NB. The values of F-Measure for RF, J48, and NB are 0.959, 0.909, and 0.659 respectively.

Figure 4.10 shows the results of the precision, recall and F-measure on HCRC method. Results show the similar behaviour as seen in 4.9 that the values of precision for RF, NB, J48 are 0.96, 0.734 and 0.909 respectively. The recall values of RF, NB and J48 that are 0.96, 0.686 and 0.908 respectively. Considering the values of F-Measure for RF, J48 and NB are 0.96, 0.908 and 0.669 respectively.

Figure 4.11 depicts the accuracy of all the three classifiers on the HHRC approach which consists of two phases. The first phase include static classifier and the second phase consists of dynamic analyzer explained in detail in section 3.1. The results of both machine learning classifiers are shown in Figure 4.11 which indicates that dynamic analyzer comes up with the highest accuracy which is 95.9% whereas NB and J48 produce 67.63% and 90.89% respectively. J48 being the second best. If considering the static analysis alone the result are not that satisfactory producing RF as 72.65% accuracy whereas NB and J48 come up with 68.66% and 66.86%. Therefore it is evident that the static analysis alone is not sufficient for the classification and detection of ransomware whereas the combination of static and dynamic analysis yields good accuracy with the improvement of 32% when compared the dynamic results of RF with static, whereas the overall accuracy yields 82%. NB shows that static phase yields better results and 1.5% reduced when the dynamic analysis is performed whereas overall accuracy reduced to 66%. J48 come up with 34.5% improvement in accuracy when comparing the static analysis phase with dynamic phase with the overall accuracy 79.7% of the HHRC(combination of static and dynamic) data. The reason for reducing in overall accuracy results is that in static phase there might be huge possibility that some of the non-ransomware were predicted as ransomware and therefore, they were eliminated for further analysis. Thus reducing the overall accuracy. However, we have seen that in HCRC the accuracy improves as there is no chance of such error.

Figure 4.12 shows the accuracy of our second approach i.e., HCRC approach explained in section 3.2 among all classifiers. It comprises of a single machine learning analyzer. The results of all the three classifiers are shown in Figure 4.11 which showed RF has achieved the highest accuracy of 96%, J48 being the second competitor by acquiring 90.8% accuracy whereas NB showed the accuracy of 68.6% only.

As shown in Figure 4.11 and Figure 4.12, it is quite evident that the highest achieved accuracy is 0.96 when the classification is done using Random Forest on

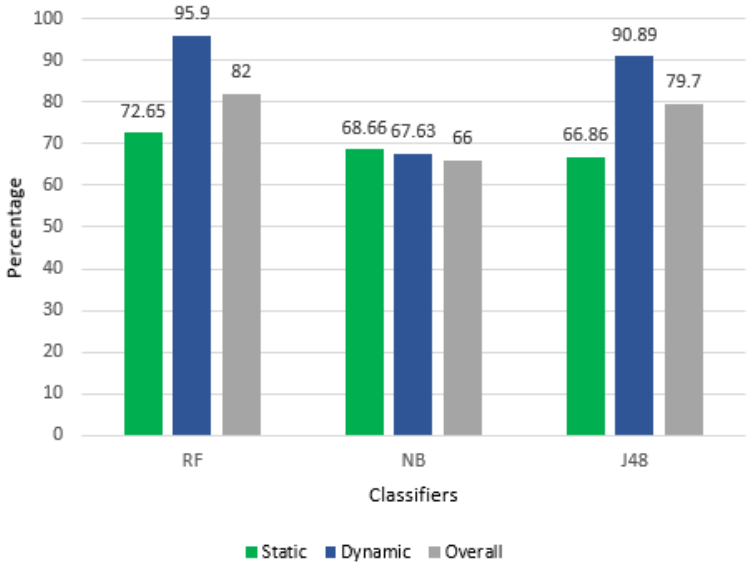


FIGURE 4.11: Accuracy of HHRC analysis

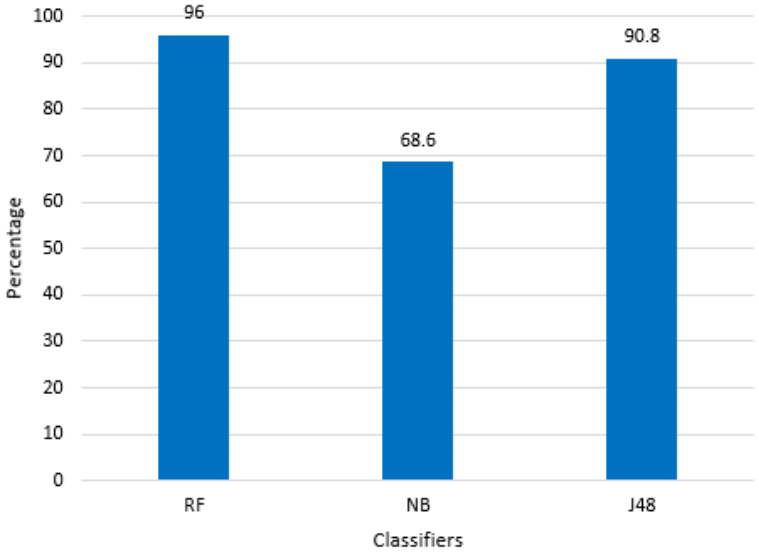


FIGURE 4.12: Accuracy of HCRC data

HCRC data which improved only 0.01% indicating that both techniques HHRC and HCRC produces similar results as far as ransomware classification and detection are concerned. The J48 technique employed on both types of hybrid data performs 0.90 accuracy which seems that the Decision Tree (J48) is the second best classifier identifies ransomware accurately from both types of data sets, while the Naive Bayes showed a low accuracy on both types of datasets.

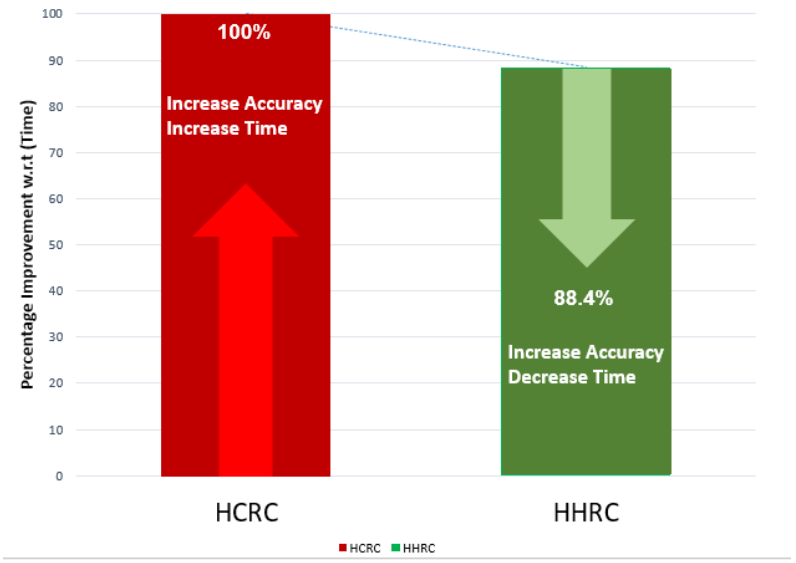


FIGURE 4.13: Percentage Improvement w.r.t time

We have further analyzed that HHRC seems to be a far better approach as compared to HCRC in the context of cost. Although the results indicating the same accuracy of 96% in classifying windows ransomware whereas the interesting factor here is reducing cost as shown in Figure 4.13 . We have not calculated time complexity but as shown in HCRC considers all the 100% applications to be executed and analyzed thus increasing time whereas, in HHRC 11.6% time reduced by eliminating all those applications that were already being recognized as ransomware in static phase 1.

Chapter 5

CONCLUSIONS, LIMITATION, AND FUTURE WORK

5.1 CONCLUSION

With the proliferation of computer devices and internet ransomware threats are seem to be increasing rapidly during the last couple of years. The damage caused by ransomware are rising with the passage of time due to problems faced by antivirus vendors in detecting new ransomware families. Ransomware and other malwares have somehow similar instructions except ransomware contains more threatening issues that are it locks the user system and demands ransom in return in order to have their data to be accessible. Therefore, the classification of ransomware from other malwares that are actually not ransomware become mandatory. In this study, we came up with the machine learning approach to classify and detect ransomware from other non-ransomware considering a variety of features. In this study, we consider hardware features that have not been previously considered for the ransomware classification with the combination of other features such as API calls, Registry Keys and File Operations.

This research work presents the comparative performance of different machine learning classifiers in Windows ransomware classification and detection as well as finding the features that play a role in ransomware classification. It has been seen that Hardware features and the Registry Keys are the two most relevant features for the classification and detection of ransomware from non-ransomware. We presented two types of methodology for ransomware classification and detection i.e., HHRC and HCRC in order to find out which hybrid technique is most suitable for analysis. HHRC approach as the name predicts follows the static and dynamic analysis in a hierarchy way. It comprises of two machine learning analyzers (static and dynamic). We start our analysis with some basic static analysis and after extracting features, these features are then given as input to machine learning analyzer which then predicts the malware to be a ransomware or non-ransomware on the basis of three classifiers (Random Forest, Naive Bayes and Decision Tree (J48)). Only those malwares that are labeled as non-ransomware are further analyzed to get some more dynamic features (such as hardware features, registry keys, API calls) by machine learning dynamic analyzer. Which then predict the executable to be ransomware or non-ransomware.

HCRC approach follows simply the combination of static and dynamic features collectively analyzed and given as input to machine learning analyzer. Results depict that both approaches yield similar results with the difference of 0.01%. Our dataset consists of 500 malwares including 250 ransomware executables and 250 non-ransomware executables. Therefore, the difference could be increased as the dataset increases.

We have used three classifiers for ransomware classification. The reason of using these are for numerical data, choices exists are too many such as starting from basic decision trees, Naive Bayes, SVM, logistic regression, ensemble methods (boosting), Random forest, multi-layer perceptron etc. Whereas for categorical data Naive Bayes, decision trees and their ensembles including Random forest, seems to be good techniques. Our dataset was 'mixed data' including both numerical & categorical data. Therefore, one option is to go with decision trees, other

possibilities are Naive Bayes where you model numeric attributes by a Gaussian distribution or else Random Forest that combines bagging and random subspace. With mixed data, choices are limited and you need to be cautious and creative with your choices.

The comparison among results with other existing techniques was not performed as it is not possible. The reason is our main focus was to maintain or improve accuracy and for accuracy, the comparison is only possible if both techniques must use an exactly same dataset with the same malware apps in the same environment and configurations. So, basically, this is not possible as far as malware analysis comparison is concerned. However, for future researchers, we will try to make our dataset publicly available so that other researchers could use our data for comparison.

Our results endorse our conclusions by achieving incredible results regarding classifiers training for ransomware classification and detection. Our trained classifier (RF) shows an AUC value of 0.98 that indicates the accurate performance of this classifier in ransomware classification and detection. TPR for this classifier is high as 0.96 and FPR is low as 0.04 for HCRC approach. We observe two more classifiers. J48 also proves to be among those who learn the ransomware patterns successfully and shows good accuracy as compared to NB. The RF AUC values for HHRC is 0.981 and for HCRC RF rises to 0.987. RF shows highest precision and recall value that is 0.96 and the f-measure 0.96 respectively. Considering static analysis alone the result are not produced satisfactory such as RF come up with 72.65% accuracy whereas NB and J48 come up with 68.66% and 66.86%. Therefore it is evident that the static analysis alone is not sufficient for the classification and detection of ransomware whereas the combination of static and dynamic analysis yields good accuracy with the improvement of 24.2% when compared the results of RF with the static analyzer. J48 shows 26.4% improvement in accuracy when comparing the static analysis data alone with HHRC (combination of static and dynamic) data.

According to the results, both techniques are similar to 99%. We have concluded that HHRC seems to be most the suitable technique for ransomware analysis and classification. Maintaining accuracy to be 96% whereas decreasing cost to 11.6% as compared to HCRC technique.

5.2 LIMITATION

It has been observed that Hardware Performance Counters is an emerging feature as far as ransomware classification is concerned. Moreover, none of the existing approaches considered Hardware Performance Counter features for the analysis for ransomware. Therefore, we have experimented and successfully grasp the top features to be from Hardware Performance Counters that is *cache-misses*. There exists a limitation for it as the attackers tried to alter in code in such a way that affects its performance while the behavior of the program remains the same will reduce accuracy.

5.3 FUTURE WORK

Application of machine learning algorithms has shown very promising results in order to make advancement in this field. As a future work, we plan to validate our framework on a large dataset. We have also faced the problem during sandbox execution as some applications fail to run or missing some shared libraries. Some of the ransomware even bypasses the sandbox and show attacking behavior which even damages our system too. Some ransomware shows silent behavior after detecting the controlled environment. Therefore, more research is needed on code coverage of executed applications during dynamic analysis. We can even train a classifier in future which can detect the ransomware application and classifies them into families. We can dig more into the network statistics, TCP packets

etc, while tracking down the dynamic properties of more hardware features for the applications.

Appendix A

DATASET USED

Table A.1: List of all ransomware hash value

No	MD5	No	MD5
1	12be6e7241d2503f31fae01046e88d68	126	6d39e2fabd738f6e5242b31d4743a152
2	aea8ab12edf294ddb2804d6618fdd247	127	0a164f674221a6a23b86a73bea05c913
3	d5d010f8b2f145399a9638f457ff3990	128	ed74b74d02b91b3fbfddc94484f031e0
4	d1510b299e8570afd352d20d516f6f48	129	f36443853cc774237e1f673ec8e3df0e
5	ff189061d35cff903af5d25858d6c484	130	fa1d3d1ba53491ed9431a1f4fe50d5bc
6	886b02878836e8bc1e06ccfe73cf1d5b	131	18294ea0655a05b6f1cdd93f3dfad0b9
7	f6fa4051156b35d3a8c9261cb0128d70	132	deec2a79f1cfbdc8dced0f68ec908a28
8	4d36c89ec1915d018b47fc1ddd685234	133	3769d77331279bd1dd5baa8e08495dea
9	3f82790d2d8ad5ddd17b11c911e4352d	134	53df46eca1aa6ce88f08bf660e597494
10	f78046c221d06596a47bbeb4288defb0	135	1abff57e05629475c1a4fa86141f75cf
11	4b68739e3e5607de02f8ef72f3cb26bc	136	93c3d81f9c7f0c69989c8b41e04ce471
12	a864077a8b7d702f2db8dc868049672d	137	c307986eda3b0319e8bdecb3010f841f
13	74b23514e6e5199c3ddd22361128f9bc	138	7edc49ef66f45353e8dc163409dbbad6
14	9d9de70e5d58094bd34c53ff52b18290	139	bcc312a0a0c7dc7ec27f25d4dbf57bac
15	9de699ef09f54e3fdd84cf7c2750bfee	140	f1e1e85fbb64962b57d11a2a76c05410
16	f08577c753c73a14eeb958451635a1c4	141	12666b5054cc0cb62cf758736340c1bc
17	d3fc3d0ad612ee6f43df58f01d7323ce	142	c63dd21d13100728828b2937482e017c

18	f8da21ba71ebf3727971bce9d9724c3e	143	f542094e530fb3bc66efcdb53461499a
19	858adddf40ed251174a1ccfa4b880090	144	6b40feb191cb174090f6bb36724f83e2
20	fdd4fbfc35e05da4b77deb9cdae89390	145	f2efea2a7783f1912d83d6dc71f08c56
21	e28ea9135eb096977e734a506a486aa0	146	29160abb461b01e29098ed3cd8d474e6
22	f73ea1b038efff72397c749d12fbcda0	147	3da6a0168ec0a7a2c55bd627cc76b830
23	920256744075b2d2cffcfc5f62c7f2a9	148	f28546e6e56bb5e14d8585c10c449d72
24	70975d8bdeeb40cb55044f569ddb585d	149	d3552bc70d3345c88f462e270459c953
25	386b47777d8366e59479528376841e28	150	de8225722cbfec609b923bd6b59492e8
26	fe778ea756169a05d3e7ae72100b1978	151	34801b671c5f2a28338570c4c8677e8f
27	2a2b0c8f4a6e765b810a81ef7d2449fd	152	f6b6aca1654ef18fcbe6855217971f50
28	f3d7df9f1882e9f753b19b89aaad6215	153	d56bf4a6522eb58a245810abab8419b3
29	96e5a9de91aa21b8d59e5680bc8e98a0	154	2a59d4ed30a0353f69cce57ad83831ea
30	f18c4cc41fa26858c7ff4296d0049f80	155	c7c3a47ece2caab5e30c351a9fc0b885
31	5abef46a05048e9be91f60b1e8763fc0	156	b594b2e759271166e40383ec952dcce1
32	f8330a82b7b7f6f7f8f249b650d95a97	157	a97e0f078d1bd42e0c004598e01d942e
33	2bafc99b1f149a88044963b577385f3b	158	9b1da810487cdcb458d46f394f561fdb
34	08401f7f7393048d0ac7716abc3a0648	159	2e922ffc0c31a82da57bf3db10b682d6
35	ef466d9b0cebfcbae016649d34a161b6	160	eac1ea433b69fc5b7479f9ebf2804c15
36	3478e753233c4163dc1b0b95133d4771	161	f58e3e28a70e188ae707d4e6020318f0
37	cf16ca22fd93735ac9661cb7f9c2fb80	162	69d0a16ad7d955771129bff188b79b83
38	c237cdb1106ed2bbe63fa122273e1d80	163	f6777c1796c840fba3df48e36c3a41b4
39	af7c40ca4ec4665ed819a9eb6409b587	164	f77b5209cb1ae00fc1f5c598df243030
40	537f3a22bb83b8643c0f676887d49f57	165	7ffb1799f23dc59c536ee492f97c4c92
41	fff891cc1fded5c7e8625b21606bdbac	166	5f0997ee267dec418efdb61112a09f65
42	fe26327c3e175af16967ee690a9671d0	167	fea1608830f74a2fb887115b9cf38e80
43	f4f1b178053078c8ed00272a655dc950	168	08cfa6421607a8405cbb20ac4e2864eb
44	b38995b27c6bcc117fc74e452b65b2c1	169	bd5eed8869652cb5bc0f3e363b987b70
45	b552c103224ce78bd1f5e5962445dd80	170	27cfd891abe1e029466e73bfee2cfc5b
46	9e9db0e6aeeb1ef625d0b92739fb2a93	171	4256b43110d9bd870360824242288aab
47	7cd68627b836ac1c5e2bbffc178b05e2	172	a1b111bd003779ccc4e6c6fe4713e92b

48	7d09214052f036749a287e6dd8ad8cb0	173	bf70eeddab5404a33cfbd39ced8c5f41
49	5c38c88d1de4970579bb155996cfb550	174	65e5113f77b35621d12675447bcd52ef
50	f3b620bdf93a054b8b60c30740999577	175	f67504657cce5ba27f12e026d3a73559
51	371d47e47c1ff7170f60aba5af05c516	176	af5f92145423675d01f09396426d6847
52	fdb15408a22b5f4396f549d68ad126e0	177	451d524f2b5353246c7eeea503e752b4
53	c2438a77e701fab954f516df8be85e6e	178	b38bad8e43b4deb6817a1cb4570b4bf0
54	18da21337a68b2edf0abedc4a6cb6b0c	179	efd36aad32b3a16271c1f13e326a1bf9
55	b268ebf7a6891026d9445ef4fd9296de	180	f570ca1031fc7d2b518271b43e6753de
56	e8b948f5bbcce8e8545c45174c7fd500	181	1eceeb4901686cad6979f3db0574b4ea
57	74ee23f11d7964ab2328341eda4b44b1	182	d4dfc17c79bf9e45ea169a7dfb8731a0
58	13e1538d403a3618db92ad0700f08b5f	183	be611e896251a55fb88a8b38c987bb60
59	1d1ba87f51e85b599f709409265419a8	184	7ef376e81b51abddc7b1598681e9cdde
60	c2a42a0dd7785ce0cc783f17f4fbe355	185	1dbf99bb878f7cad04ba363045556071
61	f59839791993b0db269cb15d57a42dc0	186	72ce1b0c54c45f1424dbd9954c7431f2
62	f9f1d58b27b7032f2c0006d104232560	187	fe8befdddcc9a3ce820a761d55158235
63	7ddef77c68d6a0acc12531a58d3f3743	188	a5fff257b81ea19b3509576e552d0b27
64	e91b9cf9e93f58d9d7d8516a3468093b	189	67ab8dd615923a5e91e33f8a284882b4
65	f671addab4fdf223053d241d59fac7a0	190	d160e20c6caead34681a07cf19299e5a
66	f006e2c76a4dfe750c08130826d0eb34	191	f609cdd24ef60e491e2cd3970fc58dc0
67	a4c1452050618d1cd5c86c2bb0482062	192	2f7efd6ba29680659d2aa0a1c246a56b
68	f7c997f81d95663bc8f57c94f83946b0	193	f2ed2531d6d8e423d268b5a051a75d40
69	6d0d13f4fb5ccc57a7bee6195ff41e99	194	b3d7fdd5a521e434e2637b57209babb3
70	f5bd93c14ce6d72bc1adae10827ed220	195	56f0c40f508f8c1321bd984c3b598e53
71	80b8411b1b2b94af31f9abebc45e12fd	196	ff8fd0f6c996b58c9a3bfeee89b09c10
72	c1a2eef9cf84ebdbf9e3580796e11c8c	197	cefcdd32223840c39e98b82ea1e6a05f
73	f55a4cfc8125cf927f9b632f026cf3e0	198	d8a33a1bd41cfa5408be35cc32a2d79d
74	ef978c66dbd14bc6af14edfbaab08780	199	c4d9811ebe4a6af8037c7eef8aa585aa
75	af9175cb56e41868c13c1ec836b09022	200	bd6316d3e1e0e20ea312ce34c0083265
76	8b0efd1759f66420392559c2c506cd6f	201	6e91bad26a5b5a0f30650a7c515066b3
77	fb6539238a7cd432ef3c01f3976dc101	202	7744b8991723e10df3796e983a49dc1e

78	54c10c2f741ce18c6596e73c4fd083db	203	1d8e332ad2ae1877c94bb2e98e580c90
79	f7539ee27967b19a234342a13d8541a0	204	055eb2a4df678c114d42706d3ce97af7
80	7574bcabb68d64a74f6948938a67cae1	205	bf28a351214372f5ffd4e4011601a8e4
81	c198fadeff984c2465f2e1ebedf4c601	206	6a60a270addbfad002914e5a5bc5ef0e
82	8ab4abdd27bb1e83740ebd29b701a96e	207	e26da8bf616eacf2f9bc6f20d5fc7710
83	fca9c2649ed7e18535bf4c745e16be50	208	cc854611017247dc151b339c2094e0e3
84	c7ef1ff53e4250e2cecda37f1cde3a4a	209	d9f39c323183524f3772072d5e3fa971
85	c962955075b09c3ca181067c248141b0	210	78b3c1ffbb671fbb0c03e1de152bc2b4
86	b5ea1d90bbe8304a6234633478e9fd0d	211	dfb7d9770e1440d26e0e62fe968fda3b
87	4daabf35a0b606a5c807564e1c5293f6	212	ed3aa0dae49ef9bea25539a10c5ae770
88	15498158598632df42dd416de292d24e	213	11ff8a8e9a643deff1dcf58e7e2fdf20
89	fcc93a5a4d010b2de2e8cebca6599f20	214	bab69fc3ad499f871e74a4fad5238830
90	2ba4903ebbf34d43f65c88d00514dadb	215	fce913371f90e5fe1464f04622903720
91	2104c98cf906bb7d3a88b7e471e8e316	216	414146ca9ee9a3b18eff07a16e34a9fb
92	0b54a6f62c6b7150d06ad876e81b21cc	217	76874ed83c4942f5a11937f53533a6c7
93	f378010e63a5cfcf9db96b819ba754e7	218	72eeb1e5d2ee146b51e3919d7330ca20
94	b223ef2e609b7b4a8138bd4c914e52d0	219	7658d91525d791f9516db5f0d30225e4
95	088ee14f59604a5fad42edd1d3b9467e	220	07ce95aeaab7fb4b6630ac576a98b883
96	db3341835fadd9f3edadc9cd19bb36cd	221	b296dbc7fecc0c4b65335236644e61b0
97	1ec80153e3f7c7e27ce8d54e8cc5303e	222	f21f2d7910bc222632dc8ea72dcf8950
98	6374833fa3cfdee0c008a01cfeac98c0	223	f7f2dbe66b60e9a576d05e05e7f51ab3
99	b2c47014b08b78bc7e350f07596273e0	224	ebf114d7dafc3c9a8df9b94b0a7d8560
100	aa6529fb7f26c3370c60181b116f773c	225	ef764bbd3dbee288df85b403865f745e
101	1ffc020d1eb24b96ff1b54d025b7335c	226	f8f26e510136e11d0ed1cec115e3eac0
102	f7ac15f0065ebc1c4e7ceb23bfb71975	227	f569adb9bed09b2dd7d51d9e007e45c2
103	d359c319165caa9dfbe6b78c83b58d60	228	f4b93dae71a92326864eb72a2a27e951
104	c1b839fde0e900bc9d6b0e1b1b2351a6	229	087d86235891edbcc02eab6880143aa5
105	0514927422bcc62e5d22e1e8c6da5f43	230	e51a1233a965d777bc9ba115db87ca10
106	4706adc3668da0e911f888d5d7acbe7d	231	749ed7635492a5ca669798564d8b0350
107	fe5c6b35be61fe6b26c66c1f77416f05	232	43ce7a42d275d1351de591b3b5d2661d

108	db8b0d2fe138e4f32440cacebb30c2ee	233	5749f981b88d2ab6644defcde96652cc
109	fc1d377b1213b008c8e6b4cbb2092551	234	fbf9833409e39b8ac4301eeaaae9c960
110	adfc311d80a2cedb63ef76c1c80e39a3	235	b689050b89ce00dd84229aae23e28d10
111	fd36eee0e9b687219cfefd4a2f1a5a0c	236	f492b061612f1fc55fc87984ec04f800
112	e7c688ecc01980b2824e308c2f80c0c0	237	2653b0e170899c2b5eab42d5c2f618c3
113	614d0780b8e631a6bf419a8209ab5c93	238	bf98ed9ba7dd102fc9dd7cdc57810203
114	40d54310207ce89061bad88550242264	239	d908af88f6cdfc9e7ff0776f08c980fc
115	018cb8a80fb37b5bc93577be1a4537f2	240	daf61f850494910d5dc608ebfee016fc
116	dee8749f50aa8a4bc06154add4a71d70	241	f516554acccb66c653384b8915ba6240
117	3398462231e58f8855c289cdc319f771	242	f05558ca4e5ee463a6a1e0cd375a9d27
118	241c6ac41922a71605e02840dd80e2f0	243	142f4b80695d9a7550c14ab3585b91e5
119	771c27201a8bdb2e3adb45f8abbdf118	244	e1dc0f416e6f924ebfc446e8580a1b30
120	f95eb0adbf1b5ff68b34de2170b16bc0	245	57619746d0e7ac3c17668c5385c6444e
121	35d60d43cc9e8261e92b77ce5f3e93cc	246	ae5b5fd7f9920248471de0b3a4df8ed9
122	e852c78346f4453a6e26ead69be8b0f0	247	cf6a507d19eb57e75f1f94f363978ac5
123	9f2bba8eff09a5a629aeefa3846a080d	248	235bf9e1da0e3db10d63c00d6a405182
124	a3b240186ef2c7c470aaec5b25ebfffb	249	ab201e3e84433991dfa2536271aff49c
125	2e2d9a9e4f1ba67bb7d6acccd2cc7cdb	250	0e3afd506ecb079ee78ef2d754e6faa0

Appendix B

FEATURES DESCRIPTION

TABLE B.1: Feature Description of Top ranked Static Feature

Feature Name	**Description**
SizeofStackReverse	The amount of virtual memory to reserve for the initial thread's stack.
MajorOperatingSytem Version	The minimum version of the operating system required to use this executable. This field is somewhat ambiguous since the subsystem fields (a few fields later) appear to serve a similar purpose.
Subsystem	The type of subsystem that this executable uses for its user interface e.g., 2 for windows GUI
Checksum	A CRC checksum of the file.
AddressOfEntryPoint	The address where the loader will begin execution.
NoOfSections	The Section table contains information about each section present e.g., code section, data section
ExecShell: warning: error ("%s": file:"%s" params:"%s")=%d	String
mscoree.dll	If compile code as an .exe assembly, the runtime is started automatically by mscoree.dll.
DecryptFileA	Function requires exclusive access to the file being decrypted, and will fail if another process is using the file.
CryptDeriveKey	Function generates cryptographic session keysderived from a base data value. This function guarantees that when the same cryptographic service provider (CSP) and algorithms are used, the keys generated from the same base data are identical. The base data can be a password or any other user data.

TABLE B.2: Feature Description of Top ranked Dynamic Features

Feature Name	**Description**
cachemisses	Cache misses is a state of not getting data which is being processed by a component or application that is not found in the cache.
branches	A branch is an instruction in a computer program that can cause a computer to begin executing a different instruction sequence and thus deviate from its default behaviour of executing instructions in order wise
instructionsPerCycle	The average number of instructions executed for each clock cycle
pageFaults	pageFaults occurs when a programs virtual content has to be copied to the physical memory.
REG.READ.HKEY _CURRENT_USER. Software.Microsoft.Windows.Current Version.Policies. Explorer.	Registry Key
REG.OPENED.HKEY _LOCAL_MACHINE.SOFTWARE. Microsoft.Windows.CurrentVersion. ShellCompatibility.Objects.	Registry Key
CreateDirectoryW	API Calls
GetVolumePathNamesForVolumeNameW	API Calls
REG.OPENED.HKEY_CURRENT _USER.Software.Microsoft.Windows. CurrentVersion.Policies.	Registry Key
REG.OPENED.HKEY_CLASSES _ROOT .CLSID..20D04FE0.3AEA.1069.A2D8.08002B30309D	Registry Key

Bibliography

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] Ö. Aslan and R. Samet, "Investigation of possibilities to detect malware using existing tools," in *14th ACS/IEEE International Conference on Computer Systems and Applications AICCSA*, 2017.

[3] A. Ferrante, M. Malek, F. Martinelli, F. Mercaldo, and J. Milosevic, "Extinguishing ransomware-a hybrid approach to android ransomware detection," in *The 10th International Symposium on Foundations Practice of Security*, 2017.

[4] C. T. Alliance, "Lucrative ransomware attacks: Analysis of the cryptowall version 3 threat," *Retrieved November*, vol. 3, 2015.

[5] D. Sgandurra, L. Muñoz-González, R. Mohsen, and E. C. Lupu, "Automated dynamic analysis of ransomware: Benefits, limitations and use for detection," *arXiv preprint arXiv:1609.03020*, 2016.

[6] B. Krebs, "Fbi: North korea to blame for sony hack," *Retrieved from KrebsOnSecurity: http://krebsonsecurity. com/2014/12/fbi-north-korea-to-blamefor-sony-hack*, 2014.

[7] A. Kharraz, S. Arshad, C. Mulliner, W. K. Robertson, and E. Kirda, "Unveil: A large-scale, automated approach to detecting ransomware." in *USENIX Security Symposium*, 2016, pp. 757–772.

[8] D. Formby, S. Durbha, and R. Beyah, "Out of control: Ransomware for industrial control systems," 2017.

[9] C. Johnson, "Kenzero virus blackmails those who illegally download anime porn/porn virus publishes web history of victims on the net. bbc." [Online]. Available: http://news.bbc.co.uk/2/hi/technology/8622665.stm

[10] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, "A survey of mobile malware in the wild," in *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*. ACM, 2011, pp. 3–14.

[11] G. Clucley, "Hacked iphones held hostage for 5 euros," *Naked Security*, 2009.

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

[13] F. Martinelli, F. Mercaldo, and A. Saracino, "Bridemaid: An hybrid tool for accurate detection of android malware," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. ACM, 2017, pp. 899–901.

[14] B. A. S. Al-rimy, M. A. Maarof, and S. Z. M. Shaid, "A 0-day aware crypto-ransomware early behavioral detection framework," in *International Conference of Reliable Information and Communication Technology*. Springer, 2017, pp. 758–766.

[15] N. Andronio, S. Zanero, and F. Maggi, "Heldroid: Dissecting and detecting mobile ransomware," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2015, pp. 382–404.

[16] G. Kaur, R. Dhir, and M. Singh, "Anatomy of ransomware malware: detection, analysis and reporting," *International Journal of Security and Networks*, vol. 12, no. 3, pp. 188–197, 2017.

[17] J. Demme, M. Maycock, J. Schmitz, A. Tang, A. Waksman, S. Sethumadhavan, and S. Stolfo, "On the feasibility of online malware detection with performance counters," in *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3. ACM, 2013, pp. 559–570.

[18] Q. Chen and R. A. Bridges, "Automated behavioral analysis of malware a case study of wannacry ransomware," *arXiv preprint arXiv:1709.08753*, 2017.

[19] D. Maiorca, F. Mercaldo, G. Giacinto, C. A. Visaggio, and F. Martinelli, "R-packdroid: Api package-based characterization and detection of mobile ransomware," in *Proceedings of the Symposium on Applied Computing*. ACM, 2017, pp. 1718–1723.

[20] S. Aurangzeb, M. Aleem, M. A. Iqbal, and M. A. Islam, "Ransomware: A survey and trends." *Journal of Information Assurance & Security*, vol. 6, no. 2, 2017.

[21] K.-s. Choi, T. Scott, and D. P. LeClair, "Ransomware against police: diagnosis of risk factors via application of cyber-routine activities theory," *International Journal of Forensic Science & Pathology*, 2016.

[22] "Cawley, C a history of ransomware: Where it started and where its going," http://www.makeuseof.com/tag/history-ransomware-russia-reveton/, accessed: 2016-08-30.

[23] T. Micro, "Ransomware," *Trend Micro,[Online]. Available: http://www. trendmicro. com/vinfo/us/security/definition/Ransomware.[Accessed 06 January 2016]*, 2016.

[24] T. S. Rajput, "Evolving threat agents: Ransomware and their variants," *International Journal of Computer Applications*, vol. 164, no. 7, 2017.

[25] M. Labs, "Threat predictions ransomware infographic," 2017.

[26] K. Savage, P. Coogan, and H. Lau, "The evolution of ransomware," *Symantec, Mountain View*, 2015.

[27] N. Hampton, Z. Baig, and S. Zeadally, "Ransomware behavioural analysis on windows platforms," *Journal of Information Security and Applications*, vol. 40, pp. 44–51, 2018.

[28] F. Mercaldo, V. Nardone, A. Santone, and C. A. Visaggio, "Ransomware steals your phone. formal methods rescue it," in *36th IFIP WG 6.1 International Conference on Formal Techniques for Distributed Objects, Components, and Systems - Volume 9688*. Berlin, Heidelberg: Springer-Verlag, 2016, pp. 212–221. [Online]. Available: https://doi.org/10.1007/978-3-319-39570-8_14

[29] Q. Jibz, "snaker, and xineohp. peid," 2005.

[30] U. Sternfeld, "Operation kofer: Mutating ransomware enters the fray," 2015.

[31] D. Nieuwenhuizen, "A behavioural-based approach to ransomware detection," *Whitepaper. MWR Labs Whitepaper*, 2017.

[32] P. Zavarsky, D. Lindskog *et al.*, "Experimental analysis of ransomware on windows and android platforms: Evolution and characterization," *Procedia Computer Science*, vol. 94, pp. 465–472, 2016.

[33] H. V. Nath and B. M. Mehtre, "Static malware analysis using machine learning methods," in *International Conference on Security in Computer Networks and Distributed Systems*. Springer, 2014, pp. 440–450.

[34] S. Song, B. Kim, and S. Lee, "The effective ransomware prevention technique using process monitoring on android platform," *Mobile Information Systems*, vol. 2016, 2016.

[35] Z.-G. Chen, H.-S. Kang, S.-N. Yin, and S.-R. Kim, "Automatic ransomware detection and analysis based on dynamic api calls flow graph," in *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*. ACM, 2017, pp. 196–201.

[36] F. A. Narudin, A. Feizollah, N. B. Anuar, and A. Gani, "Evaluation of machine learning classifiers for mobile malware detection," *Soft Computing*, vol. 20, no. 1, pp. 343–357, 2016.

[37] F. Beneventi, A. Bartolini, C. Cavazzoni, and L. Benini, "Continuous learning of hpc infrastructure models using big data analytics and in-memory processing tools," in *Proceedings of the Conference on Design, Automation & Test in Europe*. European Design and Automation Association, 2017, pp. 1038–1043.

[38] A. Kharraz, W. Robertson, D. Balzarotti, L. Bilge, and E. Kirda, "Cutting the gordian knot: A look under the hood of ransomware attacks," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2015, pp. 3–24.

[39] L. Perf, "Linux profiling with performance counters," *Accessed: October*, vol. 17, 2016.

[40] "Virusshare virushare," http://www.virusshare.com/, accessed: 2018-03-30.

[41] C. Guarnieri, A. Tanasi, J. Bremer, and M. Schloesser, "The cuckoo sandbox," 2012.

[42] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *Icml*, vol. 97, 1997, pp. 412–420.

[43] C. E. Shannon, "A mathematical theory of communication, part ii," *Bell Syst. Tech. J.*, vol. 27, pp. 623–656, 1948.

[44] K. S. Jones, *Readings in information retrieval*. Morgan Kaufmann, 1997.

[45] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[46] J. R. Quinlan, "C4. 5: Programming for machine learning," *Morgan Kauffmann*, vol. 38, p. 48, 1993.

[47] S. Khan, "Which algorithm fits best for categorical and continuous independent variables with categorical response in machine learning?" [Online]. Available: https://www.quora.com/Which-algorithm-fits-best-for-categorical-and-continuous-independent-variables-with-categ answer/Shehroz-Khan-2

[48] T. Arens, F. Hettlich, C. Karpfinger, U. Kockelkorn, K. Lichtenegger, and H. Stachel, *Mathematik*. Springer-Verlag, 2015.

[49] N.-A. Le-Khac and A. Linke, "Control flow change in assembly as a classifier in malware analysis," in *Digital Forensic and Security (ISDFS), 2016 4th International Symposium on*. IEEE, 2016, pp. 38–43.

[50] W. Chen, D. Aspinall, A. D. Gordon, C. Sutton, and I. Muttik, "More semantics more robust: Improving android malware classifiers," in *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 2016, pp. 147–158.

[51] R. Tian, L. Batten, R. Islam, and S. Versteeg, "An automated classification system based on the strings of trojan and virus families," in *Malicious and Unwanted Software (MALWARE), 2009 4th International Conference on*. IEEE, 2009, pp. 23–30.

[52] M. Z. Shafiq, S. M. Tabish, and M. Farooq, "Are evolutionary rule learning algorithms appropriate for malware detection?" in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. ACM, 2009, pp. 1915–1916.

[53] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss, "andromaly: a behavioral malware detection framework for android devices," *Journal of Intelligent Information Systems*, vol. 38, no. 1, pp. 161–190, 2012.

[54] G. Kaur and A. Chhabra, "Improved j48 classification algorithm for the prediction of diabetes," *International Journal of Computer Applications*, vol. 98, no. 22, 2014.

[55] M. Ivan, "How-to simulate support vector machine (svm) in r," 2018. [Online]. Available: https://en.proft.me/2014/04/22/how-simulate-support-vector-machine-svm-r/

[56] C. E. Metz, "Basic principles of roc analysis," in *Seminars in nuclear medicine*, vol. 8, no. 4. Elsevier, 1978, pp. 283–298.

[57] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[58] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-validation," *Encyclopedia of database systems*, pp. 1–7, 2016.