T.C.
SELÇUK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

AUTOMATED ANALYSIS APPROACH FOR THE DETECTION OF HIGH SURVIVABLE RANSOMWARES

Yahye Abukar AHMED

Ph.D. THESIS

COMPUTER ENGINEERING DEPARTMENT

APRIL-2020
KONYA

TEZ KABUL VE ONAYI

Yahye Abukar AHMED tarafından hazırlanan "Sinsi Fidye Yazılımlarının Tespiti için Otomatik Analiz Yaklaşımı" tez adlı çalışması 21/04/2020 tarihinde aşağıdaki jüri üyeleri tarafından oy birliği ile Selçuk Üniversitesi, Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda DOKTORA TEZİ olarak kabul edilmiştir.

Jüri Üyeleri	**İmza**
Başkan Prof. Dr. Sabri KOÇER	[illegible]
Danışman Doç. Dr. Barış KOÇER	[illegible]
Üye Doç. Dr. Mustafa Servet KIRAN	[illegible]
Üye Doç. Dr. Mehmet HACIBEYOĞLU	[illegible]
Üye Dr. Öğr. Üyesi Hasan Ali AKYÜREK	[illegible]

Yukarıdaki sonucu onaylarım.

Prof. Dr. Saadettin Erhan KESEN
LEE Müdür V.

DECLARATION

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

TEZ BİLDİRİMİ

Bu tezdeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve tez yazım kurallarına uygun olarak hazırlanan bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

[illegible]

Yahye Abukar AHMED
Date: 21/04/2020

ÖZET

DOKTORA TEZİ

SİNSİ FİDYE YAZILIMLARININ TESPİTİ İÇİN OTOMATİK ANALİZ YAKLAŞIMI

Yahye Abukar AHMED

Selçuk Üniversitesi Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Doç. Dr. Barış KOÇER

Üniversite Dişi Danışman: Dr. Shamsul HUDA

2020, 216 Sayfa

Ransomware, kullanıcı ile ilgili dosyaları ve verileri şifreleyen ve onları fidye olarak tutan kötü amaçlı bir yazılımdır. Bu tür saldırılar hem bireyler hem de iş organizasyonları için ciddi tehdit oluşturan en yaygın kötü amaçlı yazılımlardan biri haline gelmiştir. Bu yıkıcı zararlı program, son yıllarda siber suçlulara çok daha büyük fidye talepleri ödeyerek birçok kuruluşun büyük gelir kaybetmesine neden olmuştur. Fidye yazılımının hızlıca büyümesini sağlayan araçlar olarak; sosyal mühendislik, e-posta eki, zip dosyası indirmesi, kötü amaçlı siteye göz atma, virüslü arama motoru gibi büyük enfeksiyon yayılma yolları olarak gösterilebilir. Ayrıca kolayca kullanılabilen şifreleme araçları, Ransomware As a Service (RaaS), bulut depolama ve kendi kendine fidye yazılımı araç kitleri bu tür kötü amaçlı yazılımların geliştirilmesini kolaylaştırmıştır. Virüs yayılmasını kolaylaştıran enfeksiyon kitleri ve mevcut geliştirme araçları fidye yazılımlarını son derece büyütmekle kalmamış, aynı zamanda yeni varyantları da daha gizlenmiş, şifrelenmiş ve değişen desenler haline getirmiştir. Bu yıkıcı zararlı programa karşı, dinamik analiz yaklaşımı böyle bir saldırıyı tespit etmek için en popüler yaklaşımdır. Dinamik analizlerin çoğu, sistem çağrılarına dayanmaktadır, çünkü bunlar işletim sisteminden hizmet talep eden programlar için bir arabirim sağlar. Bununla birlikte, virüs yazarının çalıştırılabilir dosyaya enjekte ettiği fazlalık ve ilgisiz sistem çağrıları, fidye yazılımı tespit edilmesini olumsuz yönde etkileyen yüksek gürültülü bir davranış dizisi oluşturmaktadır. Bu yüzden de algılama motorları ransomware'in yeni varyantlarını tespit edememektedir. Bu araştırma hem denetimli hem de yarı denetimli makine öğrenme tekniklerini kullanarak etkili Windows API çağrı dizileri üzerinden imzasız bir algılama yaklaşımı önermiştir. Bu hedefe ulaşmak için, gürültülü özellikleri kaldırmak, fidye yazılımının gerçek davranışını karakterize etmek ve en alakalı özellik alt kümesini seçmek için Gelişmiş Maksimum Alaka Düzeyi ve Minimum Yedeklilik (EmRmR) filtre yöntemi önerilmiştir. Orijinal mRmR'den farklı olarak, EmRmR az sayıda değerlendirmeyle orijinal mRmR algoritmalarına özgü gereksiz hesaplamaları önler. Buna ek olarak, bu çalışmada, fidye yazılımının kritik davranışını açıklamak için anlamlı olmayan Windows API çağrılarını kaldırarak programın çağrı izlerinin boyutunu azaltmak için bir arıtma işlemi geliştirilmiştir. Rafine edilmiş sistem çağrılarını kullanarak birkaç sınıflandırıcı algoritması geliştirilmiş ve saldırının erken aşamalarında fidye yazılımını tespit etmek için daha düşük yanlış pozitif oranla yüksek doğruluk elde edilmiştir. Buna ek olarak, bu araştırma geleneksel denetimli algılama motorunun sınırlamalarına değinmekte ve ayrıca derin öğrenme yaklaşımlarını kullanarak yeni varyantlardaki değişken örüntülerin doğal gizli kaynaklarını denetimsiz bir şekilde hesaplamak için yarı denetimli bir çerçeve önermektedir. Önerilen çerçeve, yaklaşan kötü amaçlı çalıştırılabilir dosyaları barındırmak için ölçeklenebilir olan vahşi ortamdan elde edilen etiketlenmemiş fidye yazılımlarından farklı desenlerdeki doğal özellikleri ayıklar. Kapsamlı deneysel sonuçlarımız ve tartışmamız, önerilen uyarlanabilir çerçevenin, fidye yazılımının farklı varyantlarının davranışlarını başarıyla ayırt edebildiğini ve mevcut denetimli yaklaşımlardan daha yüksek performans elde edebildiğini göstermektedir.

Anahtar Kelimeler: Fidye yazılımı, Sistem çağrısı, Terim Frekans-Ters Belge Frekansı, Maksimum Alaka Düzeyi ve Minimum Artıklık, N-Gram, Dijital gasp, derin öğrenme, uyarlanabilir yaklaşımlar.

ABSTRACT

Ph.D. THESIS

AUTOMATED ANALYSIS APPROACH FOR THE DETECTION OF HIGH SURVIVABLE RANSOMWARES

Yahye Abukar AHMED

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCE OF SELÇUK UNIVERSITY

THE DEGREE OF DOCTOR OF PHILOSOPHY IN COMPUTER ENGINEERING

Advisor: Assoc. Prof. Dr. Barış KOCER

External Advisor: Dr. Shamsul HUDA

2020, 216 Pages

Jury
Prof. Dr. Sabri KOÇER
Assoc. Prof. Dr. Barış KOCER
Assoc. Prof. Dr. Mustafa Servet KIRAN
Assoc. Prof. Dr. Mehmet HACIBEYOĞLU
Dr. Hasan Ali AKYÜREK

Ransomware is malicious software that encrypts the user-related files and data and holds them to ransom. Such attacks have become one of the most widespread malwares that poses serious threat to both individuals and business organizations. This destructive malicious program has caused many organizations to lose huge revenue by paying much bigger ransom demands to the cyber criminals in recent years. Explosive growth of ransomware is due to the existing large infection vector such as social engineering, email attachment, zip file download, browsing malicious site, infected search engine which are boosted dramatically by easily available cryptographic tools, Ransomware As a Service (RaaS), increased cloud storage and off-the-self ransomware toolkits. The large infection vector and available toolkits not only grew ransomware extremely, but also made them more obfuscated, encrypted and varying patterns in the new variants. Against this destructive malicious program, the dynamic analysis approach is the most popular approach for detecting such an attack. The majority of dynamic analysis relies on the system calls as these provide an interface for programs to request service from the operating system. However, the redundancy and the irrelevant system calls that the ransomware authors inject in the actual execution flow of suspicious binaries generate a high noisy behavioral sequence that adversely impacts in the induction of the supervised classifiers. This, in turn, caused the conventional supervised analysis and detection engine to fail to detect the new variants of ransomware. This research proposed a non-signature-based detection approach on the effective windows API call sequences using both supervised and semi-supervised machine learning techniques. To achieve this objective, we proposed an Enhanced Maximum-Relevance and Minimum-Redundancy (EmRmR) filter method to remove the noisy features and select the most relevant subset of features to characterize the real behavior of the ransomware. Unlike the original mRmR, the EmRmR avoids unnecessary computations intrinsic in the original mRmR algorithms with small number of evaluations. In addition, this research has introduced a refinement process to reduce the size of the program's call traces by removing those windows API calls that do not have strong indication for describing the critical behavior of the ransomware. We developed several classifiers algorithms using refined system calls and achieves high accuracy with a lower false-positive rate for detecting ransomware in the early phases of the attack. In addition, this research addresses the limitations of conventional supervised detection engine and also proposed a semi-supervised framework to compute the inherent latent sources of the varying patterns in the new variants in an unsupervised way using deep learning approaches. The Proposed framework extracts the inherent characteristics in the varying patterns from the unlabeled ransomware obtained from the wild which is scalable to accommodate upcoming malicious executables. After accomplishing Our extensive experimental results and discussion demonstrate that the proposed adaptive framework can successfully discriminate the behavior of different variants of ransomware and achieve higher performance than existing supervised approaches.

Key Words: Ransomware, System call, Term Frequency-Inverse Document Frequency, Maximum Relevance and Minimum Redundancy, N-Grams, Digital extortion, deep learning, adaptive approaches.

PREFACE

"Ransomware is unique among cybercrime because in order for the attack to be successful, it requires the victim to become a willing accomplice after the fact"

-James Scott, Institute for Critical Infrastructure Technology, 2018

The most prevalent and potentially devastating form of malware, ransomware encrypts the user's related files and hard drive, and demands payment of a ransom before a deadline. A famous global ransomware attack of this variety occurred in 2017, when the Wannacry ransomware targeted thousands of computers around the world and spread itself within corporate networks. The frequency of ransomware attacks increased by three times in 2017 over 2016: an attack occurred every 40 seconds. For example, WannaCry cyber-attack has been reported in 99 countries and over 75,000 attacks have been carried out on machines running the Windows operating system. The losses due to ransom was calculated as 200 million USD per year extorted by the criminal gangs. Due to this significant economic loss, severity of disruption in sensitive business organizations, and the explosive growth of ransomware, the detection of ransomware has been an important research field which gives us the motivation of this thesis. Against this destructive malicious program, the dynamic analysis approach is the most popular and reliable approach for detecting such an attack. The majority of dynamic analysis relies on the system calls as these provide an interface for programs to request service from the operating system. However, the redundancy and the irrelevant system calls that the ransomware authors inject in the actual execution flow of suspicious binaries generate a high noisy behavioral sequence that adversely impacts in the induction of the supervised classifiers. The propose of this thesis is to describe and monitor the valuable features of ransomware dynamically by conducting a behavioral-based analysis of ransomware within sandbox in an isolated environment, and to developed detection models for ransomware utilizing supervised machine learning algorithms, and adaptive detection engine using deep learning based semi-supervised model.

Yahye Abukar AHMED
KOYNA- 2020

ACKNOWLEDGMENT

First and foremost, I would like to express heartfelt gratitude and my sincere appreciation to both my supervisor **Assoc. Prof. Dr. Barış KOÇER** and **Dr. Shamsul Huda** for their constant support, encouragement, guidance and friendship. They inspired me greatly to work in this thesis. Their willingness to motivate me contributed tremendously to our thesis. I have learned a lot from them. This thesis would not have been possible without their exceptional guidance, creative suggestions, patience, motivation and support. I am fortunate to have they're as my mentor and supervisor. Besides my supervisors, I am deeply grateful to the rest of my thesis jury: **Prof. Dr. Sabir KOÇER, Assoc. Prof. Dr. Mustafa Servet KIRAN**, **Assoc.Prof. Dr. Mehmet HACIBEYOĞLU**, **Dr. Hasan Ali AKYÜREK** for their wholehearted cooperation, identifying the weaknesses, helpful suggestions and insightful comments for my thesis.

My sincere thanks go to the authority of **Selçuk University** for providing me with a good environment and facilities such as Computer laboratory to complete this thesis and software that I needed during the process. I also extend my sincere gratitude to the Turkish Scholarships (**Türkiye Bursları**) for granting me the opportunity to study in Turkey with full scholarship. Moreover, I extremely acknowledge the **Turkish people** for their warm and generous hospitality throughout my graduate school journey in Turkey. I also like to thank my sponsor **Simad University** for the financial support and the endless effort they provide me during my leave study.

My heartful thanks go to my beloved dear wife **Layla Mohamud Nageye** and my lovely daughters **Yusra**, **Yasmin**, **Yildiz** and **Yara** for their support, love, encouragement, advice, and brightening my world. I would like to show deep gratitude to my family- mother, brothers and sisters, for supporting me spiritually and financially throughout writing this thesis and my life in general. Finally, my worm thanks to my friends for always being there whenever I need them in matters to do with my research.

Yahye Abukar Ahmed
April 21, 2020
Konya, Turkey

LIST OF TABLES

LIST OF FIGURES

LIST OF BOXES

LIST OFABBREVIATIONS

ACC: Accuracy Rate

AES: Advanced Encryption Standard

AIDS: Aids Information Disk

ANN: Artificial Neural Network

API: Application Program Interface

AUC: Area Under Curve

BSOD: Blue Screen of Death

C&C: Command-and-Control

CIA: Confidentiality Integrity and Availability

DT: Decision Tree

HTTP: Hypertext Transfer Protocol

IoT: Internet of Things

IRP: I/O Request Packets

JSON: JavaScript Object Notation

kNN: K-Nearest Neighbor

LR: Logistic Regression

M2M: Machine-to-Machine

MBR: Master Boot Record

MD5: Message Digest-algorithm 5

MFT: Master File Table

MID: Mutual Information Difference

ML: Machine Learning	
MLP: multilayer perceptron	
MRI: Magnetic Resonance Imaging	
MRMR: Minimum-Redundancy Maximum-Relevance	
OS: Operating System	
PCA: Principal Component Analysis	
PE: Portable Executable	
PEOHF: Portable Executable Optional Header Fields	
QEMU: Quick Emulator	
RaaS: Ransomware-as-a-Service	
RBF: Radial basis function	
RF: Random Forest	
ROC: Receiver Operating Characteristic	
RSA: Rivest–Shamir–Adleman	
SHA-256: Secure Hashing Algorithm, 256-Bits	
SMB: Server Message Block	
SME: Small Medium Enterprises	
SVM: Support Vector Machine	
TF-IDF: Term Frequency-Inverse document frequency	
URL: Uniform Resource Locator	
VBR: Volume Boot Record	
VMM: Virtual Machine Monitor	

TABLE OF CONTENTS

1. INTRODUCTION

The most devastated and fast-spreading computer world attack is ransomware that can encrypt the assets in the victim's machine, make it unavailable to the users and pose a serious threat to achieving the CIA Triad (Al-rimy, Maarof, & Shaid, 2019) security goals such as availability. The term ransomware is originally derived from two combined words ransom and malware. After encrypting the victim's assets ransomware author demands a ransom for the restoration of the assets (user's data) into their original states (ur Rehman, Yafi, Nazir, & Mustafa, 2018). If the victim paid the ransom to the attacker through the anonymous currency mechanisms like Bitcoin (Kalaimannan, John, DuBose, & Pinto, 2017) , the access to the encrypted assets is made available again. The malware encrypts the most important user's files on the hard drives, removable drives and mapped network shares for extortion. Once ransomware reaches to the victim's machine through the infection vehicle, it starts the reconnaissance phase in which it searches for OS version, installed applications, user's files and folders, accessibility functions, backup files and folders, credential information in the victim's machine, and thereby identifies the most important resources and files (Scaife, Carter, Traynor, & Butler, 2016). After the encryption, the ransomware displays a message that requires payment to restore the captured user's data. The next step is to register the decryption key with a particular user and make available when the ransom is paid; therefore, ransomware uses the command-and-control (C&C) server to establish communication with its creator (Ahmadian, Shahriari, & Ghaffarian, 2015).

Although the revolution of ransomware appeared at the end of the 1980s (Shukla, Mondal, & Lodha, 2016)when the PC CYBORG also known as Aids Info Disk (AIDS) Trojan starts to calculate the number of times the machine has booted until a criterion number (90) reached. After that, the Trojan AIDS locks the critical user's files, hides all directory and encrypts the labels of the files on the drive C: (Shukla et al., 2016). This ransomware targeted the healthcare industry, after 28 years, the healthcare industry remains a top target for ransomware attacks. However, the sequence of successful attacks of ransomware has resulted in increasing many new ransomware variants in the last few years; for instance, the WannaCry cyber threat has been reported in 99 countries, and over 75,000 attacks have been carried out on machines running the Windows operating system (Al-rimy, Maarof, & Shaid, 2018).

The motivation is the significant revenue of the extortion, for example, effective ransomware like CryptoWall version 3.0 earned an estimated $325 Million as extortion in the USA alone (Moore, 2016). A report released by FBI just in 2016 estimated that the losses of $1 billion caused by ransomware. The victims of ransomware are not only limited to home users or individuals but also targets government networks, businesses and health services. It causes damage to financial losses or sensitive information that can lead to the disruption of daily operations (Da-Yu, HSIAO, & Raylin, 2019).

Availability of cryptographic tools and easy anonymous financial transaction methods such as cryptocurrencies, off-the-shelf ransomware development kit such as eda2, angler exploit kit, Neutrino exploit kit, Ransomware-as-a-Service (RaaS), increased usage of cloud-based file sharing are the primary reasons for explosive growth of ransomware which encourages ransomware attacker to develop new ransomware variants in the last few years (Mansfield-Devine, 2017) .

On other hand, machine learning (ML), a broad branch of artificial intelligence, is computational methods using regularities, induced patterns and previous experience to improve accurate predictions and performance. Machine learning is the science of getting computers to act without being explicitly programmed. It is designed to develop the efficiency of computer algorithms to solve with large-scale of data. In machine learning classifier is used to recognize contents inside executable code files to classify new files from normal files (Menahem, Shabtai, Rokach, & Elovici, 2009). The classifier is a set of rules that is applied to a specified training of malicious executables and normal files.

Generally, classifiers are trained to recognize unseen malicious executables as maliciousness, and complex patterns recognition that lead to intelligent decisions based on the training data. In machine learning algorithms track the sequences generated by the system calls and addressed as the characteristics of the program. The programs interact with the operating system through system calls. Therefore, the input of machine learning algorithm depends on the feature extraction which generates new features that are extracted from the original one, while the selection methods keep the subset of the original features. The extracted features including API calls (Takeuchi, Sakai, & Fukumoto, 2018).

The remaining of this section is organized as follows; the first subsection will discuss problem background. The second subsection problem statement is described in detail. The scope of the thesis will also be highlighted in subsection three. The rest are the objectivities of the research is also discussed in the fourth subsection. Finally, the significance of the study will be illustrated.

2. Problem Background

Recently, ransomware has become one of the most widespread malware threats that internet users experience (Ahmadian et al., 2015). Normally, victims of ransomware are not limited to home users or individuals, but also targets government networks, businesses, national health service hospital and causes permanent or temporary loss of proprietary or sensitive information, disruption to regular operations and financial losses. This threat has become a major cyber risk for many organizations; small-medium enterprises (SME) to large enterprises business and individual entrepreneurs (Al-rimy et al., 2018).

For example, the courier companies FedEx and TNT, Maerx, WPP (the world's largest advertising agency), pharmaceutical company Reckitt Benckiser and Kingdom's National Health Service (Mansfield-Devine, 2017). These attacks caused severe financial losses, e.g. an estimated damage by the WannaCry alone was 5 billion dollars approximately. The leading courier company of the world, FedEx, acquired $300million financial loss resulting from disrupted operations, legal and reputational cost caused by ransomware attacks (Al-rimy et al., 2018). The world leader of shipping and logistic business, Maresk lost $200 million to $300 million due to catastrophic ransomware attacks which caused it to shut down its 76 terminal ports (Yaqoob et al., 2017). In 2016, the Hollywood Presbyterian Medical Center (HPMC) computer network was down for more than a week as the Southern California hospital worked to recover from a ransomware attack, after a ransom of 40 Bitcoins — approximately $17,000 — was paid, the hospital's computer systems were released. A report released by FBI just in 2016 estimated that losses of $1 billion caused by ransomware. A report released by McAfee, demonstrates ransomwares have grown since 2014 (Al-rimy et al., 2018).

In January 2017, a hotel in Austria named Seehotel Jagerwirt was affected by a ransomware attack that took over the systems of the hotel and tampered the room key cards and guest check-ins (Mansfield-Devine, 2017). As part of the ransom to release the hotel's computers, hackers required that the hotel pays 2 Bitcoin or roughly 1,500 euro or $1,600. The hotel agreed to this payment because it was faster and cheaper than trying to fight it. May 15, 2017, new ransomware, called WannaCry emerged, which is a kind of ransomware that targets all kinds of files including PDF files, word documents, excel sheets, etc, and encrypts them in the form of .wcry extension. WannaCry causes crises across the world and infecting vulnerable systems globally. WannaCry cyber-attack has been reported in 99 countries and over 75,000 attacks have been carried out on machines running the Windows operating system (Da-Yu et al., 2019).

The explosive growth of ransomware happened due to enormous availability of easy cryptographic tools for applying encryption techniques such as single key (symmetric key mechanism), dual key (public-private key) or hybrid to produce ransomware (Yaqoob et al., 2017), easily available financial transaction methods with anonymity such as P2P cryptocurrencies which influence ransomware authors to feel safe (not being caught by law enforcement agencies), availability of off-the self-ransomware development kit such as eda2, angler exploit kit, Neutrino exploit kit, Ransomware- as-a-Service (RaaS) based on the cloud platform which enable a novice to create ransomware and spread. Increased usage of cloud-based file sharing such as OneDrive, Google drive has also accelerated ransomware distribution for large business organizations. Often ransomware authors not only demand the ransom, but the installed ransomware also create mass disruption in the system, for example, WannaCry locked out the health professionals from the electronic medical recording system (EMRS), computerized tomography (CT), magnetic resonance imaging (MRI) scanners, blood test service systems of UK's national health services (Zhao et al., 2018).

Detection of the ransomware is commonly performed by tools such as anti-virus programs based on the analysis of the signature recognition. Ransomware analysis approaches are widely classified into static and dynamic analyses. In the static analysis approach, there is no need to execute the ransomware samples. When a new malicious sample is explored, the static detection needs to catch its binary signature through analyzing the executable instructions (Sgandurra, Muñoz-González, Mohsen, & Lupu, 2016). Ransomware detection program searches the virus signature database to find if there are matched signatures. If a match is found, the file under test will be identified as a malicious executable. This approach has proved to be effective when the malware is known beforehand in the database, and the accuracy is totally dependent on the signature database of the system. However, this signature-based detection method is hampered by the avoidance techniques that ransomware employs such as obfuscation and/or packing. Such an approach is unreliable for detecting to the zero-day ransomware, as it suffers several shortcomings such as frequently updated signature repository, and the need for expert intervention to analyse and extract attack signatures (Fukushima, Sakai, Hori, & Sakurai, 2010).

In the dynamic analysis, on the other hand, ransomware samples are executed in a controlled environment such as sandbox to reveal the runtime behaviour of the samples. Certain dynamic behavioural features are extracted from the malicious file and used for classification and detection purposes. The most promising approaches to detect and characterize the malware behaviour are system calls as they provide a valuable information and attack patterns that help in the detection of such attacks. To execute the suspicious payload, ransomware needs to request services from the operating system through Windows API calls. These system calls can represent the essential characteristics of the ransomware. However, significant growth of ransomware through a huge infection vector, changes the patterns of infection very rapidly. This requires a sophisticated detection engine which is based on the runtime feature of ransomware and requires as less supervised knowledge as possible (Vinod & Viswalakshmi, 2018).

2.2 Problem Statement

Ransomware is malicious software that encrypts the user-related files and data and holds them to ransom. Such attacks have become one of the serious threats to cyberspace. The avoidance techniques that ransomware employs such as obfuscation and/or packing makes it difficult to analyse such programs statically. Although many ransomware detections studies have been conducted, they are limited to a small portion of the attack's characteristics. In the dynamic analysis, several current studies rely on system calls as they are effective for distinguishing between the behaviour of malicious and benign programs. A system call is a way for programs to interact with the operating system. A computer program makes a system call when it makes a request to the operating system’s kernel. System call provides the services of the operating system to the user programs via Application Program Interface (API). It provides an interface between a process and operating system to allow user-level processes to request services of the operating system. System calls are the only entry points into the kernel system. (Al-rimy, Maarof, & Shaid, 2017).

Authors Hampton et al. employed windows API call features for identifying the salient feature of the ransomware. For the detection purpose, the frequency of the system calls for the ransomware and baseline applications were compared to measure the similarity between them (Hampton, Baig, & Zeadally, 2018). Amamra et al. introduced a filtering and abstraction process to eliminate the irrelevant and redundancy system calls for anomaly-based malware detection. This process has also combined the same system calls to reduce the size of the traces (Amamra, Robert, & Talhi, 2015). However, the redundant and irrelevant system calls that are injected by the malware authors in the actual execution flow of suspicious binaries can easily defeat these detection approaches.

Moreover, the size of the system call traces is commonly very large that generates a high noisy behavioural sequence (Chou, Yen, & Luo, 2008). This has adversely impact on the induction of machine learning classifiers such as the increase in training time, more storage requirement and the difficult analysis of real malicious behaviour that can lead overhead and poor prediction ability (Xiao, Xia, Yang, Huang, & Wang, 2015).

To address this issue, dimensionality reduction approaches such as filters and wrappers have been proposed to handle the noisy problem and select the optimal features to improve the performance of the classifiers. Wrappers select features based on predetermined learning algorithms, but this method tends to be computationally expensive and has overfitting problems (Acid, De Campos, & Fernández, 2011). Unlike wrapper methods, filters select the subsets of the features by finding the correlation to the target class without involving any learning algorithm.

Filters are less computational than wrapper approaches. Among the widely-used filter methods, the Minimum-Redundancy Maximum-Relevance (mRmR) method has been successfully employed in the malware detection applications in the past few years.

Several studies have employed such approach as this provides a relevant feature for discriminating the behaviour of the malware and benign files (Sedano et al., 2015). However, the original mRmR method has a limitation of unnecessary computations due to the mutual information calculations among feature sets. In the mRmR, to find the most relevant subset feature, the mutual information between a specific feature and the class target is quantified. The redundancy of the features is penalized based on mutual information within features (Darshan & Jaidhar, 2018). This process continues until the subset features are equal to the selected features, and the algorithm calculates the same mutual information values more than one time that leads to duplications.

Therefore, the original mRMR method is not suitable for the detection of ransomware because it is computationally expensive due to the large number of system call features generated by n-gram. Therefore, we need a lighter version of mRmR to overcome this difficulty.

2.3 The Research Questions

There are important questions which arise:

1. Feature extraction is a key to apply machine learning to successfully detect malicious executables, which feature extraction approach can propose significant features that can represent the real behavior of the ransomware?

2. The most promising approaches to detect and characterize the ransomware behaviour are system calls as they provide valuable information and attack patterns. Windows API calls are suffering a massive amount of irrelevant and redundant system calls invoked by the malicious executables during its execution, how to reduce the size of the system call traces?

3. How supervised machine learning implemented using an integrated number of features?

4. How to develop an adaptive detection engine using deep learning-based semi-supervised model on an integrated number of features?

5. How to evaluate the efficiency of supervised machine and semi-supervised in detecting of the ransomware.

2.4 Objectives of the Research

The following are the objectives of the research:

1. We proposed a framework for describing dynamically monitored valuable features of ransomware by conducting a behavioral-based analysis of ransomware within a sandbox in an isolated environment, through the Term Frequency-Inverse document frequency (TF-IDF), Enhanced Minimum-Redundancy and Maximum-Relevance (EmRmR) and FastICA methods, we have extracted the most relevant features that provide the best performance in detecting new ransomware on windows platforms.

2. We have developed detection models for ransomware utilizing supervised machine learning algorithms, and adaptive detection engine using deep learning based semi-supervised model. The proposed method achieves high accuracy and less false positive rate for detecting ransomware in the early phases of the attack.

3. We have empirically validated the method with an extensive experimental evaluation to show the effectiveness of the proposed models.

2.5 Scope of the Research

The scope of this research will be the following:

1. Focus on ransomware that exists in Microsoft Windows platform, due to a large number of the ransomware attack occurs; there are more Windows-based computers than any other type of OS. Ransomware attackers often use exploit kits software in Microsoft based machines to get access on victims' machines.
2. For analysis purposes, the samples are executed in Cuckoo sandbox installed in Ubuntu 16.04 LTS Desktop fully updated, with WindowsXp_server_Pack3 32bit installed as a guest machine due to its weaker security protections that enable us to observe more ransomware behavior. To perform the analysis in a secure, Virtual box machine was used with controlled access to the Internet.

3. In this research, Supervised Machine learning and semi-supervised techniques were focused as they provide the real characteristic of ransomware during execution, because they perform statistical comparisons on specific datasets to examine the accuracies of the algorithms.

4. Four common performance metric was used, so evaluate the performance of ensemble machine learning technique are True Positive (TP), False Positive (FP), True Negative (TN), and finally False Negative (FN).

2.6 Significance of the Study

Regarding cyber-attacks caused by the malware, the most wide-spread and sophisticated destructive is the one motivated by the ransomware. Ransomware is one of the most discussed cyber security threats and constitutes a hot topic in the cybercriminals in present time. The number of infected ransomware victims has dramatically increased now days from the perspective of small individuals, businesses, enterprise and some hospitals. The losses due to ransom was calculated as 200 million USD per year extorted by the criminal gangs. Due to the significant economic loss and severity of disruption in sensitive business organizations, the detection of ransomware has been an important research problem.

Therefore, an efficient of ransomware detection can save sensitive data, organization integrity and financial loss, it provides computer home user and organizations confidence in the security field. The expected outcome of this study is to detect malicious executable files with better accuracy in comparison to other detection methods. This proposed scheme can be used in real-life situations such as business and organization network. The supervised machine learning and semi-supervised techniques is expected to have higher performance while maintaining low false positives. This study will be beneficial to the antivirus researches through effective machine learning algorithms, and provide recommendations on how to evaluate the performance of a certain ML algorithms in accordance to ransomware detection.

2.7 Organization of the Research

This study consists of six sections. Section 1 is about introduction of the study, Problem background, objectives, scope and significance of the project. Section 2 provides the literature reviews on ransomware, the categorization and the types of the ransomware. Analysis of the ransomware based on the static and dynamic approach. In this section the detection methods including machine learning algorithms will be illustrated. The framework of methodology and data set used to detect new executable malicious files will be discussed in the Section 3. Section 4 analysis and data pre-processing steps, feature extraction and feature selection, developing models, parameter settings are also discussed. Finally, result and discussions of the proposed methods and their extensive experiments, and comparing the proposed method based on the accuracy of the algorithms are discussed in Section 5.

2. LITERATURE REVIEW

This section reviews the literature of ransomware and its detection based on the different aspects. It begins the overview of the ransomware in terms of the its revolution and the phases of the ransomware attack. The category of the ransomware based on the threat type of view, whether the fake ransomware that scares the user to extort it or the real ransomware is focused in this section. In this section, two types of real ransomware, those lock the victim's screen while other variant encrypts the user's related files called crypto-ransomware are also classified. In addition, in this section, the digital extortion that becomes a major cyber risk for many organizations; small-medium enterprises (SME) to large enterprises business and individual entrepreneurs, followed by the various popular type of the ransomware families are discussed, these families are based on the behaviours of the ransomware including the type of the algorithm used, the encryption approach, the amount of the extortion, and the threatening messages are classified. Analysing of ransomware, both static and dynamic approaches are also briefly explained. This section defines ransomware analysis as the action taking malware apart to study it in order to determine the impact and sophisticated level of ransomware. It also concluded the detection of ransomware including signature, anomaly and emulation-based detection. The remaining subsections are discussed the avoidance techniques used by the ransomware writers to evade the detection such as encryption, compression data and obfuscation techniques. This subsection addresses detection mechanisms of unseen nasty code through data mining techniques based on the extraction of static malicious features from binary files. Finally, the machine learning classifier algorithms to identify new files as benign or malicious is focused.

2.1 Overview of Ransomware

The expansion of the Internet and its importance is increasing at an amazing rate in recent years, not only the size but also the services offered; along with this particular importance and benefits, the number of complex attacks has also grown especially, in the wide use of the Internet. In recent years, the malicious code has posed a serious security threat to business and commercial companies, computer network system and governments. Therefore, the level of the security in malicious code has reached a peak, in (Reddy & Pujari, 2006), ranked the impact of viruses and worms as top serious security threats. Moreover, as the number of the unknown virus rises, the rate of detection complexity also increases. Due to the significant increase of the available tools and the extortions encouraged to increase the attack of malicious programs like the newly emerged malicious executable called ransomware.

In recent years, ransomware has been making headlines around the world, but this kind of software is not new. The first type of ransomware appeared in 1989 (Shukla et al., 2016). It was the Trojan AIDS also known as PC Cyborg. At that time, AIDS was one of the newspapers in the whole world. After that, Doctor Joseph Popp took advantage of the situation and distributed around 20,000 floppy disks to patients, individuals and also medical institutions. This diskette contains an AIDS information program (Da-Yu et al., 2019). But it also contained ransomware, which after a few days encrypted computer files and then demanded a ransom of $ 189 to recover the encrypted files.

The first attack was the PC cyborg in 1989; the ransomware attacks remained unnoticed until the mid-2000s. One reason is that hackers wrote their own encryption code, which was quite simple to decrypt and, therefore, easy to counter. But everything changed when they started to rely on encryption libraries that are almost impossible to decrypt without the decryption key (Hampton et al., 2018). The first ransomware to use encryption techniques arrived in 2005 (for example Gpcod used RSA1024 bit encryption). GPCoder infected Windows systems and targeted files with a variety of extensions**.** There are two types of ransomware: Encrypting ransomware and blocker as we will discuss in the following sections, but in simple way, the encrypting ransomware encrypts files and folders on the computer while blocker ransomware locks the devices. Both ask for a ransom to allow the victim to regain control of their data or device(Gazet, 2010).

Ransomware has taken on a whole new dimension and it all started with the popularization of Bitcoin, which allows hackers to be very difficult to trace. In addition, encryption algorithms have become more and more complex, which makes them almost impossible to decipher without knowing the key.

Some even decrypt a file to show the victim that the key actually works. This pushes victims to pay the ransom since they are confident that hackers can unlock the files. For businesses, paying the ransom is often the cheapest option. So, if companies are sure they will find their files for a fee, they will not hesitate. Because of all these elements that make ransomware viable and very attractive in financial terms, their number is simply exploded, as the graph indicates (Kalaimannan et al., 2017).

2.2 Ransomware Attack Phases

To encrypt the user’s related files, ransomware requires to carry out attack phases, this will lead the ransomware successfully spread and infect the machine. The following are the most prominent phases of ransomware:

2.2.1 Infection phase

Ransomware can attack a computer, a smartphone or a Tablet, using different techniques: phishing, adware or malicious applications as we will discuss in the following infection vector section. Once the malicious payload is hosted in the machine, the "Ransomware" can be triggered either remotely by the hacker, or at a date and time previously defined or when the user performs a specific action (Dada, Bassi, Chiroma, Adetunmbi, & Ajibuwa, 2019).

2.2.2 Spoliation of the backup phase

Once the malicious file is executed, the Ransomware can locate and remove the backup files to prevent the user from performing a restore (Dada et al., 2019).

2.2.3 Encryption phase

At the heart of crypto ransomware, main objective is its ability to transform mass amounts of data from a usable state to an unusable state. Typically, ransomware data transformation function is employed through encryption by opening the original file and directly overwrites its content with its encrypted data. Depending on its category, ransomware can encrypt files, display a permanent threat message (overlay) and even change the password of a terminal. In any case, the user can no longer use his device (Dada et al., 2019).

2.2.4 Notification phase

The user is informed that his files are being held and that he must pay a ransom to recover them. Often, victims have a few days to pay, otherwise the ransom amount increases. The files are eventually permanently deleted once the authorized time is reached (Palisse, Le Bouder, Lanet, Le Guernic, & Legay, 2016).

2.3 Ransomware Categorization

The categorization of the ransomware is based on the several factors that determine the layout of ransomware such as:

1. The type of threat Classification,
2. The targeted approach that infects the victims
3. The nature of infecting the systems.

2.3.1 Ransomware threat type classification

We classified the ransomware according to the type of threat to the infected machine. This threat varies based on the different factors, the purpose of the attack and the type of victim. So, from this threat type point of view, ransomware is classified as scareware and Real ransomware.

2.3.1.1 Fake ransomware

This type of ransomware does not compromise the user's files, but they only scare the users that they have encrypted the files. Fake ransomware criminals tackle the fear of ransomware threats instead of creating the real ransomware, they only use a simple encryption tool. The purpose of the fake ransomware is extortion by persuading the victim to pay, this kind of ransomware employ social engineering as an attack vector by showing an encrypted page so that the victim can think his/her data can be recovered(Pathak & Nanded, 2016). Another purpose of fake ransomware is to divert the attention of the users from the real attack which is another ransomware.

To infect the user's machine, fake ransomware uses a social engineering technique to convince the users that their computer systems are compromised and they are offering free antivirus downloads to scan for the ransomware (Rajab, Ballard, Marvrommatis, Provos, & Zhao, 2010). The fake antivirus plays on the security fears and calls for the user to take actions in self-preservation. For instance, Personal Shield Pro is a rogue antivirus program that infects the system and takes over the control of the compromised computer. This program pretends to be the updates of some programs such as Shockwave, Flash, or codecs. When the Windows boots, the Personal Shield Pro performs a fake scan to infect the machine. Personal Shield Pro is capable of infecting Windows 9x, 2000, XP, Vista, and Windows 7.

2.3.1.2 Real ransomware

In contrast to the fake ransomware, the real ransomware is a harmful program that uses various system utilities to escalate the extortion. We can divide this type of ransomware into two main categories, locker ransomware and crypto-ransomware (Cabaj & Mazurczyk, 2016).

A. Screen-locker ransomware

Screen- Locker ransomware is a malicious program that locks the screen of the victim when the computer is compromised. and emails victims into thinking their computer is locked. After the affection, the ransomware blocks the victim's desktop, computer input devices for end users or mobile devices or input interface devices such as the keyboard and mouse by denying access to the device owner (Pathak & Nanded, 2016). The ransomware displays a message on the screen and allows limited access to some functions such as moving the mouse or keeping the keys on the numeric keypad activated so that the victim can enter the ransom and pay a ransom before the normal access is restored(Aurangzeb, Aleem, Iqbal, & Islam, 2017).

The Screen-Locker ransomware accuses the victims accessing un illegitimate websites or doing a prohibited activity. The Screen-Locker ransomware imitates a police officer that is going to punish the computer users for employing pirated software. The Screen-Locker Ransomware displays a message for ransom but does not include any detailed instructions about how to make the payments (Aurangzeb et al., 2017).

This Locker-ransomware keeps the system and the files intact and can be removed through various system restoration techniques such as restoring the system to its safe Mode in order to find the original data that ransomware locked (Bhardwaj, Avasthi, Sastry, & Subrahmanyam, 2016). Un updated anti-malware software can also be removed from the malicious payload associated with the screen-locker ransomware. The following are some Screen-Locker ransomwares:

- Kovter
- Winlock
- Reveton
- LockScreen
- BlueScreen

B. Crypto-ransomware

Crypto-Ransomware is malicious software that encrypts the user-related files and data and holds them to ransom. User's data access is permitted again if the victim paid the requested ransom using the anonymous currency mechanisms like Bitcoin. Ransomware that employs the encryption algorithms is known as crypto ransomware. The revolution of the crypto-ransomware begun in 2013 when Crypto-Locker appeared (McIntosh, Jang-Jaccard, & Watters, 2018). The aim of the crypto-ransomware is to breach the availability of the data by encrypting a victim's files, or rendering them inaccessible as shown in Figure 2.1.

The ransomware encrypts the most important user's files on the hard drives, removable drives and mapped network shares for extortion (Kharraz & Kirda, 2017). After the encryption occurs, the ransomware shows a message that requires payment to restore the captured user's data (Ahmadian et al., 2015). The next step is to register the decryption key with a particular user and make available when the ransom is paid; therefore, ransomware uses the command- and- control(C&C) server to establish communication with its creator (Brewer, 2016). The Crypto-ransomware contacts C&C server through multiple proxy servers which are typically legitimate but hacked machines to request a public encryption key. The amount of ransom is vary depending on the specific ransomware variant, and the payment is often only in Bitcoins, or a similar digital cryptocurrency. Specific instructions are also provided

Unlike the locker-ransomware, the effect of a crypto-ransomware attack is irreversible; to encrypt the victim's files, the crypto-ransomware employs cryptography functions. In the first quarter of 2016, the increase of the crypto-ransomware becomes high as reported in (Gostev, Unuchek, Garnaeva, Makrushin, & Ivanov, 2016), due to its ability to exhibit massive damage and tangible extortion against victims.

The Crypto-ransomware spread is changing dramatically. In 2018, Sophos discovered that half (54%) of the organizations that they investigated had been the victim of ransomware in the past year. The main target was government networks, businesses, and national health service hospitals. The impact crypto-ransomware showed that India had the highest level of infection, followed by Mexico, the United States and Canada. A Report released by the FBI just in 2016 estimated that losses of $1 billion caused by ransomware (Moore, 2016).

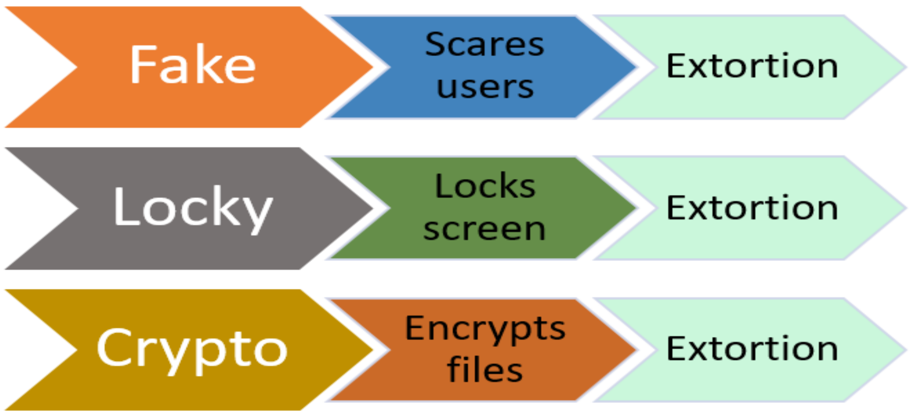


Figure 2.1: Ransomware Categorizations (Kok, Abdullah, Jhanjhi, & Supramaniam, 2019)

2.3.2 Ransomware platform classification

Ransomware variants can be classified based on the targeted environment. The target platform includes Personal computers (PC), internet of things (IoT) or mobile environment. The detailed description about the environmental-based ransomwares classification are provided below:

2.3.2.1 Personal computer ransomware

The personal Computer ransomware as the name implies, this type ransomware is a malicious program that infects only the personal data or user's multiple files on the individual computers. The PC ransomware spread to other computers when the attachment is sent via email or carried by users on physical media such as USB drives, an external hard disk, or floppy disks. According to the McAfee and Symantec reported that the number of ransomwares that attack PCs is growing dramatically. Attacks of this type are not only limited to the windows-based computer, but also other PC-based systems such as Mac OS and Linux.

The PC ransomware prevents the victim from accessing their data, the attack phases of the Windows based ransomware is shown in Figure 2.2. There are several ways to do this, such as encrypting data or blocking computer access as we mentioned in the previous subsections. These methods are intended to obtain the payment of a ransom. Once paid, the victim will be able to access their data (Al-rimy et al., 2018).

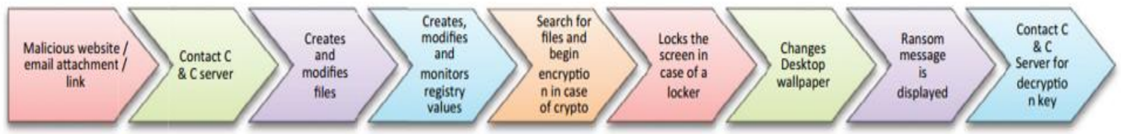

Malicious website / email attachment / link
Contact C & C server
Creates and modifies files
Creates, modifies and monitors registry values
Search for files and begin encryption in case of crypto
Locks the screen in case of a locker
Changes Desktop wallpaper
Ransom message is displayed
Contact C & C Server for decryption key

Figure 2.2: The Phases of the windows-based ransomware (Zavarsky & Lindskog, 2016).

In window-based ransomware, after delivering and installing the malicious payload on the system, some significant changes are observed. These changes can be described as File system activities, registry activities, and network communications (Zavarsky & Lindskog, 2016).

- **File System Activities:** during the attack of the windows-based ransomware, several files are modified, opened, deleted and created. The constantly muddied files include a.txt files that the ransomware employed for threatening the victim after the encryption carried out, for contacting PIPE\lsarpc is used with the Local Security Authority subsystem. For resistance purpose, the Cryptowall ransomware changed the system.pif available under the Start Menu. Not to recover the encrypted files, window-based ransomware employed *vssadmin* tool to deletes the shadow using the command Delete *Shadows/All/Quietcommand (Zavarsky & Lindskog, 2016).*
- **Registry Activities:** after the execution of the samples, most of the windows-based ransomware modified the registry key values. Here, the most observed changed register keys like Crypto-Wall do as presented in the box 1.

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run, HKU\S-1-5-21-842925246-1425521274-308236825-500\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell

Box 2.1: Registry activities made by the Crypto-Wall during execution

Then, the Crypto-Wal changes the AppData value to C:\Documents and Settings\Administrator\Application Data, cache value to C:\Documents and Settings\Administrator\Local Settings\Temporary Internet Files. Some variants modified the registery key values of the computer name. for instance, the following are: HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Nls\ComputerName\ActiveComputerName and HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\WinLogon. Some keys like HKLM\System\CurrentControlSet\Control\Terminal Server checks if Terminal Server user is enabled or not. It makes sure that Language Hotkey and Layout Hotkey are also enabled(Chen & Bridges, 2017).

- **Network Activity:** The windows-based ransomware contacting the Command and Control server by starting communication with a Client Hello using TLSV1 with the response. The server then sent through a certified message to victim's machine. After establishing the communication between the server and the victim's machine Client key exchange are performed in the victim's system. Also Encrypted Handshake message is completed to obtain the encryption keys and other messages (Zavarsky & Lindskog, 2016).

2.3.2.2 Mobile ransomware

The market for smartphones has boomed considerably in recent years. These phones have exceeded their primary functionality of voice communication and are now real mini-computers, with their own operating system which allows the user to install all kinds of applications. Although the number of phone models are significantly used, two operating systems largely dominate the market: Apple's iOS and Google's Android. The latter allows any user with some programming knowledge to create and publish their own applications on the Google Play site, where other users can download them. On the one hand, malicious applications (ransomware) are regularly found in this market. The newly emerged malicious program include ransomware as it becomes aware of such contamination, Google reacts by removing suspicious applications from the market. However, the time required for this reaction leaves time for many users to become infected (Zavarsky & Lindskog, 2016). These markets are not controlled and are therefore infested with malware: it is crucial for these users to be able to detect them in order to limit the risks. The following are some of the mobile ransomware during the attack:

- **Privilege Escalation**: when the application is delivered to user's mobile, after that, the application needs to open, so that, it has to request for administrator rights. To take the privileges of the application users are required to activate the button by clicking it., and this causes the malicious application to be removed from the device. The newly emerged ransomware variant, the activation window, is covered with a malicious window imitating to be an update patch installation.

So, in some way, the program attempts to find the administrator privileges to lock the victim's device.

- **Information Collection:** during the attack, the ransomware collects the important information about the International Mobile Equipment Identity (IMEI) number, call logs, contacts, profile, history bookmarks, SMS, the list of accounts in account service, phone state, GPS location of the phone, and IP address. Some of the ransomware even check the tasks running on the device.
- **Permissions Used:** All programs that are installed by normal users 'needs authorizations and permission to be given to role properly. Nonetheless, the ransomware requests for permissions that is not intended for the working of the program. All the permission requests which don't appear to be in agreement with application functions can be taken threat and may not be granted (Nicoló Andronio, Zanero, & Maggi, 2015).

2.3.2.3 Internet of Things (IoT) ransomware

Today, the Internet is gradually transforming into a Hyper Network, like a network formed by multitudes of connections between Artefacts (physical, documentary), actors (biological, algorithmic), scripts and concepts (linked data, metadata, ontologies , folksonomies), called "Internet of Things (IoT) Internet of Things (IoT)", connecting billions of human beings, but also billions of objects. It becomes the most powerful tool ever invented by humans to create, modify, and share information. This transformation shows the evolution of the Internet network: from a network of computers to a network of personal computers, and then to a nomad network integrating communications technologies. Developments in Machine-to-Machine (M2M) technologies for remote machine control and also the appearance in the year 2000 of IP (Internet Protocol) on mobile cellular networks have accelerated the evolution of M2M towards 'IoT (Yaqoob et al., 2017).

The Internet of Things must be designed for easy use and secure handling to avoid potential threats and risks, while masking the underlying technological complexity. The Internet of Things become potential target of ransomware attacks. Even though these devices are intended to save the personal or organizational documents and files, locker-ransomware attacks become the most serious threat to the IoT devices by disrupting the normal flow of the work and deactivate the access to the surveillance systems. This kind of ransomware causes a power outage, or imposing a disturbing break of the manufacturing processes. For instance, the Android.Lockdroid.E is considered to be the IoT-based ransomware that locks smart TVs and disables the users to access (Richet, 2016).

2.3.3 Ransomware target classification

In this section, we will discuss the effect of the ransomware based on the targeted victim. The classification of this kind of ransomware can be divided on the three main part as explained in the following section as detail.

2.3.3.1 Individual

The best targets for ransomware are individuals, businesses or even public institutions. Anyone can be the target of an attack. But hackers have learned to target their attacks better. There are many reasons for hackers to attack individuals:

- Individuals almost never back up their data. They lack computer security knowledge which makes them easy to handle (for example they do not pay attention to what they will click).

- They do not keep their software up to date. They use free antivirus which are less efficient; they think that antivirus protection against all threats. Basic protections are not implemented (proxy, firewall, etc.).

All of these elements make individuals vulnerable to a ransomware attack. Of course, It is obvious that these criticisms do not apply to everyone and luckily elsewhere. But users should be aware of that a single mistake can allow the ransomware to infect their machine (Palisse et al., 2016).

2.3.3.2 Business

For businesses, the reasons for attacks are also numerous. First, businesses have money and a ransomware attack causes a lot of trouble for the business. There is also the human factor which is still undervalued, hackers use social engineering techniques to deceive the collaborators of the company. Hackers can use malware to attack computers, servers, and even files on sharing systems. Small businesses are not prepared to deal with ransomware attacks. In addition, companies prefer not to report a ransomware infection for fear of the consequences of the image of the company. Indeed, such an incident can have serious consequences for a company. These can be financial but it can also break the trust that customers, suppliers or partners have with the company (Corrigan, 2017).

As the researcher stated that 70% of companies pay the ransom which is a huge percentage, but as we said above, it is impossible for a company to work without its data. So, if they don't have a backup plan, they are forced to pay in the hope of recovering the data. The biggest risk is the data that employees save in unprotected locations. Ensure that users are aware of and follow best practices for recording data so that it is stored securely. If this is not the case, do not hesitate to organize training or simulation sessions. Saving important data to suitably protected storage will help reduce the risks (Corrigan, 2017).

2.3.3.3 Public institutions

Public institutions also have employees who are rarely aware of the risks of social engineering which is used with skill by hackers. Hackers can see successfully attacking a media-known target as personal achievement. But the biggest problem remains that public institutions generally use outdated software and hardware, so their computer systems have flaws that have not been patched up. As we can see, hackers can attack anyone. It all depends on the resources and motivations of the hacker (Kendzierskyj & Jahankhani, 2019).

2.4 Types of Ransomware

There are different types of ransomware families as shown in Table 2.1, those who block the computer screen, those who encrypt the data, those who block the start-up of the system and those which target the mobiles. The first versions of ransomware were those lock the computer screen until the victim paid the ransom. There was no data encryption. So, the victim has to find a way to remove the ransomware from the computer to recover the data (Hampton et al., 2018). After that, ransomware has evolved and become a more serious threat to the world. One of the evolutions of ransomware is encrypting ransomware. This method works only after installing data from network drives, computers or servers. There are some who even look for connected USB sticks.

Encrypting ransomware is effective because the private data is often important to the users. Once, the file is encrypted, it is impossible for the victim to recover with a simple reset of the operating system(Al-rimy et al., 2018). To encrypt, there are two ways.

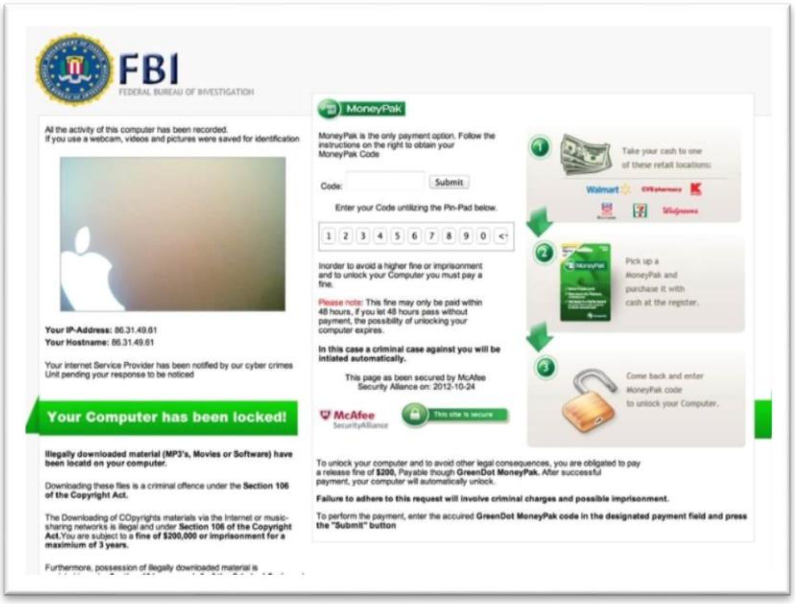
- The symmetrical way that uses the same key to encrypt and decrypt. So, this way is easier to break since it uses the same key. The victim only needs to find the key used to recover their data.
- The asymmetric way, in this method there is a public key which is shared and which allows the data to be encrypted. There is also a private key, which it is not shared and is used to decrypt the data. The private key will only be released once the ransom has been paid.

Once the data has been encrypted, a screen message will appear showing the amount of the ransom that the victim has to pay if they want to recover their data. Generally, the victim is asked to pay in bitcoin (cryptocurrency), since it is very difficult to identify who owns the bitcoin addresses. The following are the famous ransomware types:

2.4.1 Reveton

The Reveton - is also known as Police Ransomware- is a computer infection program that locks the victim's computer unless the demanded ransom is paid to the ransomware developer. Reveton is identified as one of the most famous Screen Lockers, which appeared in August 2012, and it infected many windows operating systems. The distribution of Reveton ransomware is Blackhole Exploit kit, after delivering the exploitation on the victim computer, it downloads Citadel malware.

The Citadel malware also downloads the Reveton payload. Citadel was malware that is very similar to the associate of Zeus malware. The characteristics of the Citadel stealer is to steal the credentials of the users that are stored in password managers such as password safe and KeepPass. Reveton usually displays a lock screen when you log in to Windows that pretends to be from a law enforcement agency based on the geographical (Hampton & Baig, 2015). For example, FBI message pops up if you are in the United States of America. A message from Metropolitan Police if you are in the United Kingdom.


FBI
FEDERAL BUREAU OF INVESTIGATION

All the activity of this computer has been recorded.
If you use a webcam, videos and pictures were saved for identification

Your IP-Address: 86.31.49.61
Your Hostname: 86.31.49.61

Your internet Service Provider has been notified by our cyber crimes Unit pending your response to be noticed

Your Computer has been locked!

Illegally downloaded material (MP3's, Movies or Software) have been locatd on your computer.

Downloading these files is a criminal offence under the **Section 106 of the Copyright Act.**

The Downloading of COpyrights materials via the Internet or music-sharing networks is illegal and under **Section 106 of the Copyright Act.** You are subject to a **fine of $200,000 or imprisonment for a maximium of 3 years.**

Furthermore, possession of illegally downloaded material is

MoneyPak

MoneyPak is the only payment option. Follow the instructions on the right to obtain your MoneyPak Code

Code: Submit

Enter your Code untilizing the Pin-Pad below.

1 2 3 4 5 6 7 8 9 0 <-

Inorder to avoid a higher fine or imprisonment and to unlock your Computer you must pay a fine.

Please note: This fine may only be paid within 48 hours, if you let 48 hours pass without payment, the possibility of unlocking your computer expires.

In this case a criminal case against you will be intiated automatically.

This page as been secured by McAfee Security Alliance on: 2012-10-24

McAfee SecurityAlliance   This site is secure

1 Take your cash to one of these retail locations:
Walmart CVS pharmacy K Walgreens

2 Pick up a MoneyPak and purchase it with cash at the register.

3 Come back and enter MoneyPak code to unlock your Computer.

To unlock your computer and to avoid other legal consequences, you are obligated to pay a release fine of **$200,** Payable though **GreenDot MoneyPak.** After successful payment, your computer will automatically unlock.

Failure to adhere to this request will involve criminal charges and possible imprisonment.

To perform the payment, enter the accuired **GreenDot MoneyPak code in the designated payment field and press the "Submit" button**

Figure 2.3: Fake message from Reveton ransomware

In order to access your computer, you must submit a MoneyPak voucher, or other payment coupons, to the malware developers and they will then unlock your computer so you can access your Windows desktop again. The displayed message is written like the victim’s computer was detected as having broken various laws regarding pornographic material, download copyrighted programs, or the distribution of copyrighted programs as shown in Figure 2.3. For example, messages usually are in the form of long lines announcement states “You have browsed illegal materials and must pay a fine”. To access the computer victim, you need to pay a fine or the government will prosecute you or will be jailed. In order to pay a fine, you will typically need to purchase a MoneyPak voucher and submit the voucher identification number into the lock screen (Bhardwaj et al., 2016).

2.4.2 Cryptlocker

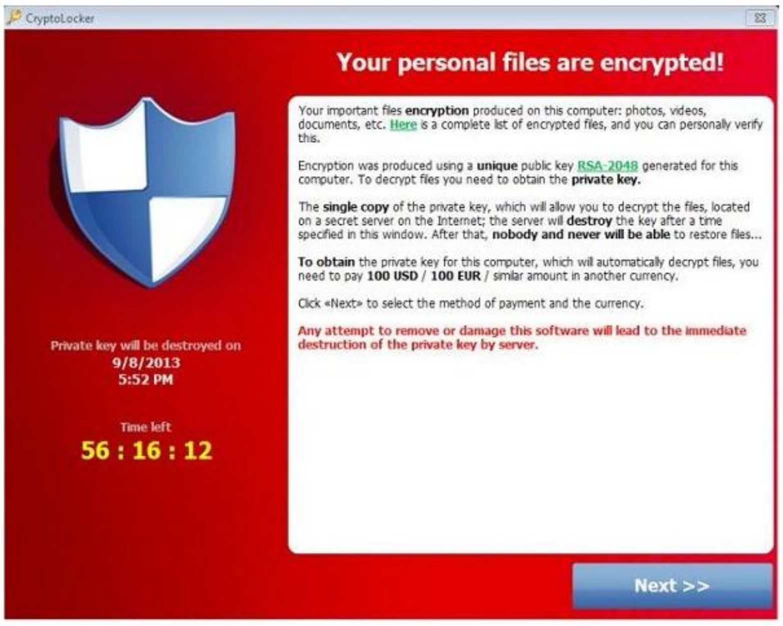
Appeared on September 5, 2013, a variant of ransomware that encrypts the user’s files on a victim's machine and demands a ransom to be paid to the malware authors. The decryption keys will be provided by threat authors via MoneyPak or Bitcoin within 72 hours, after that limited time the decryption keys will be destroyed and the user’s files will be almost impossible to recover (Scaife et al., 2016). The files are encrypted using AES with a random key which is then encrypted to a 2048-bits RSA public key. The Crypto-locker encrypts important widely used user’s related files such as word extension files, PDF files and excel files as shown in Box 2.2. Crypto-Locker spread in two methods. In the first version release, Crypto-Locker attackers focus on the on companies, business professionals through spam emails pretending as a customer complaining the organization's products. The malicious payload files are distributed as ZIP archives, which destructively encrypt all the user's files on a system if it is open (Kyurkchiev, Iliev, Rahnev, & Terzieva, 2019).

Later versions of Crypto-Locker that appeared on October 7, 2013, spread through Game over Zeus, a peer-to-peer botnet who have used the Cutwail spam network to send huge amounts of spam emails that occur as established online retailers and financial institutions. These emails are usually forged invoices, order urgent confirmations or unpaid balances to lure victims to follow the malicious links that have redirected to Crypto-Locker to operate the kits.

*.jpe, *.jpg, *.3fr, *.accdb, *.ai, *.arw, *.bay, *.cdr, *.cer, *.cr2, *.crt, *.crw, *.dbf, *.dcr, *.der, *.dng, *.doc, *.docm, *.docx, *.dwg, *.dxf, *.dxg, *.eps, *.erf, *.indd, *.kdc, *.mdb, *.mdf, *.mef, *.mrw, *.nef, *.nrw, *.odb, *.odc, *.odm, *.odp, *.ods, *.odt, *.orf, *.p7b, *.p7c, *.p12, *.pdd, *.pef, *.pem, *.pfx, *.ppt, *.pptm, *.pptx, *.psd, *.pst, *.ptx, *.r3d, *.raf, *.raw, *.rtf, *.rw2, *.rwl, *.sr2, *.srf, *.srw, *.wb2,

Box 2.2: List of extension files encrypted by the Crypto-locker.

Crypto-locker employs a dual key (public/private) encryption method, which means that decrypting the files is impossible without the private key. This is where the ransom part comes into play: after Crypto-locker executed and infected the machine, a message displays demanding a ransom within a limited time in 72 hours. The average of Crypto-locker ransom is about $300 to $2,000. The payment is required to be in Bitcoin form, which is an untraceable electronic monetary system. If the victims fail to pay the ransom within mentioned above time, the ransomware will not appear from the victim's system leaving your important files still encrypted – and unusable. The ransom notice may appear to come from the government or the police, but this is not the case. Paying the ransom may – or may not – remove the malware and there is no guarantee it will not re-infect your system in the future (Liao, Zhao, Doupé, & Ahn, 2016).


CryptoLocker

Your personal files are encrypted!

Your important files **encryption** produced on this computer: photos, videos, documents, etc. Here is a complete list of encrypted files, and you can personally verify this.

Encryption was produced using a **unique** public key RSA-2048 generated for this computer. To decrypt files you need to obtain the **private key.**

The **single copy** of the private key, which will allow you to decrypt the files, located on a secret server on the Internet; the server will **destroy** the key after a time specified in this window. After that, **nobody and never will be able** to restore files...

To obtain the private key for this computer, which will automatically decrypt files, you need to pay **100 USD** / **100 EUR** / similar amount in another currency.

Click «Next» to select the method of payment and the currency.

Any attempt to remove or damage this software will lead to the immediate destruction of the private key by server.

Private key will be destroyed on
9/8/2013
5:52 PM

Time left
56 : 16 : 12

Figure 2.4: Crypto-locker ransom note.

According to the above Figure 2.4, the ransom message states that the personal files, photos and videos of the victim has been encrypted with strong encryption algorithm that nobody will never be able to restore the files. This kind of threat is displayed to the victim and only a certain amount of time is given to the normal users to pay a ransom and recover the access to their files. Crypto-locker leaves the so-called ransom note like the one shown above.

2.4.3 Teslacrypt

TeslaCrypt was discovered in February 2015, which encrypts the files and spread through websites that convey victims to an exploit kit. Unlike other ransomware, during the attack, TeslaCrypt saves the encryption keys on the user's hard disk. This kind of ransomware is considered to be the first ransomware that attacks specifically on files used by video games. Similar to the CryptoLocker, the encryption is performed with the stream cipher. The infection vector of TeslaCrypt was through compromised websites that redirect the victims to an exploit kit (drive-by-download).

Angler is considered to be the carrier who uses Adobe Flash. The encryption process takes place independently of communication with the C&C, but their concept of cryptosystems differs somewhat (Wyke & Ajjan, 2015). TeslaCrypt includes a public ECC key in its binary code, shared among many examples, which is used to calculate a shared secret involved in generating an AES session key. The system has an ECC master private key.

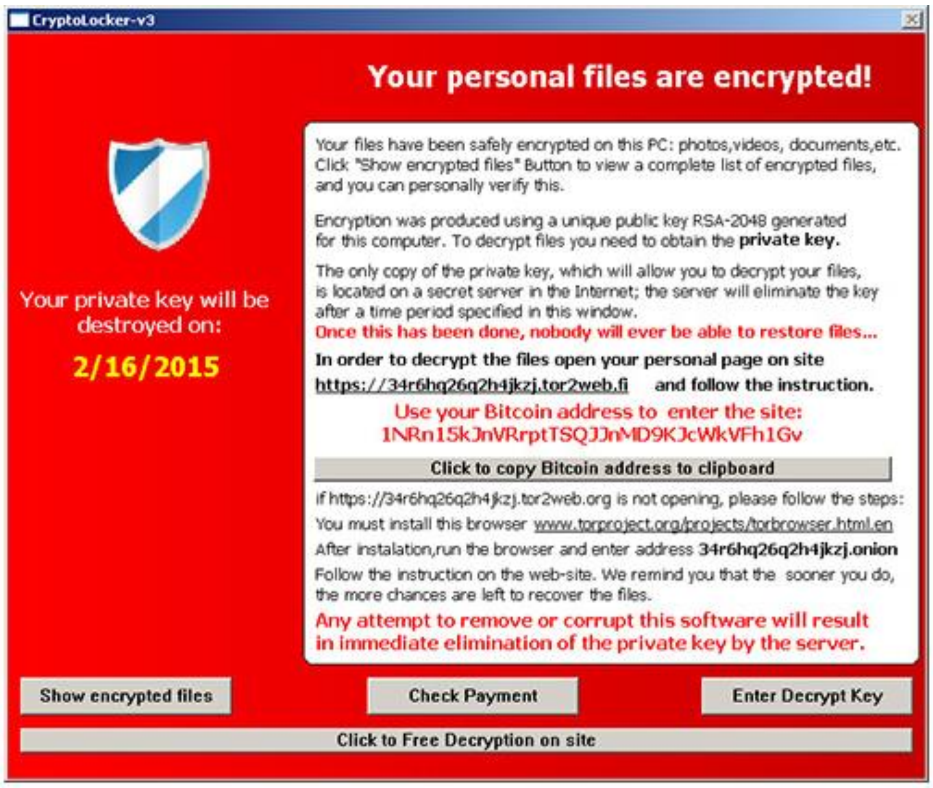
TeslaCrypt focuses many of the file extensions that commonly used on the system for general-purpose. It does not encrypt the music and video file formats such as MP3s and MP4s, as well as many file extensions associated with common business-class applications. It encrypts the file formats from efficiency suites such as Open Office and Microsoft Office, as well as formats associated with video games as shown in Box 2.3 (Pascariu & Barbu, 2015). To encrypt the user's files, TeslaCrypt employs the Advanced Encryption Standard (AES) algorithm, but the ransomware perverts its file encryption in two main methods:

1. The encrypted files have been renamed with an "ecc" extension files, which suggests the use of an Elliptic Curve Cryptographic (ECC) algorithm. The ransomware employs the algorithm when generating Bitcoin addresses, but not to encrypt files.
2. Splash screen messages and files left on compromised systems claim to use the RSA-2048 encryption algorithm.

.7z, .map, .m2, .rb, .jpg, .cdr, .png, .mcmeta, .wmo, .rar,.indd, .jpeg, .vfs0, .itm, .m4a, .ai, .txt, .mpqge, .sb, .wma, .eps, .p7c, .kdb, .fos, .avi, .pdf, .p7b, .db0, .mcgame, .wmv, .pdd, .p12, .DayZProfile, .vdf, .csv, .d3dbsp, .psd, .pfx, .rofl, .ztmp, .sc2save, .sis, .hkx, .sie, .sid, .bar, .pem, .crt, .dbfv, .mdf, .wb2, .cer, .sum, .ncf, .upk, .rtf, .ibank, .t13, .t12, .qdf, .gdb.tax, .pkpass, .bc6, .bc7, .bkp, .qic, .bkf, .sidn, .sidd, .mddata, .itl, .itdb, .icxs, .fsh, .w3x, .unity3d, .big, .menu, .das, .der, .layout, .dmp, .blob, .esm, .001, .vtf, .dazip, .fpk, .mlx, .kf, .iwd, .vpk, .tor, .psk, .rim, .ntl, .hvpl, .hplg, .hkdb, .mdbackup, .syncdb, .gho, .svg, .cas, .lrf, .css, .jpe, .odt, .ods, .dng, .js, .vpp_pc, .ff, .odp, .pak, .3fr, .flv, .odm, .m3u, .cfr, .arw, .odc, .srf, .py, .snx, .lvl, .odb, .desc, .sr2, .doc, .bay, .crw, .cr2, .dcr, .xxx, .arch00, .wotreplay, .docx, .docm, .wps, .xls, .kdc, .xlsx, .xlsm, .mef, .epk, .erf, .rgss3a, .bik, .xlsb, .nrw, .mrwref, .slm, .orf, .xlk, .lbf, .ppt, .pptx, .pptm, .mdb, .accdb, .pst, .sav, .raf, .raw, .re4, .apk, .rwl,

Box 2.3: TeslaCrypt encrypts this list of extension files.

The encryption process begins with the malware using the GetLogicalDriveStrings() API function to enumerate storage on the system's lettered drives (e.g., C:\). The GetDriveType() API call then selectively targets DRIVE_FIXED drives (e.g., hard disks, solid-state drives (SSDs)) and DRIVE_REMOTE drives (e.g., mapped network shares). TeslaCrypt does not attack removable (e.g., USB) storage or scan connected networks for open shares. The ransomware recursively scans the drives for files with targeted extensions, and then opens, reads, and encrypts each file. The encrypted data is written into the original file, which reduces the likelihood that forensic tools can recover the original data (O'Kane, Sezer, & Carlin, 2018).


CryptoLocker-v3

Your personal files are encrypted!

Your private key will be destroyed on:

2/16/2015

Your files have been safely encrypted on this PC: photos,videos, documents,etc. Click "Show encrypted files" Button to view a complete list of encrypted files, and you can personally verify this.

Encryption was produced using a unique public key RSA-2048 generated for this computer. To decrypt files you need to obtain the **private key.**

The only copy of the private key, which will allow you to decrypt your files, is located on a secret server in the Internet; the server will eliminate the key after a time period specified in this window.
Once this has been done, nobody will ever be able to restore files...

In order to decrypt the files open your personal page on site https://34r6hq26q2h4jkzj.tor2web.fi and follow the instruction.

Use your Bitcoin address to enter the site:
1NRn15kJnVRrptTSQJJnMD9KJcWkVFh1Gv

If https://34r6hq26q2h4jkzj.tor2web.org is not opening, please follow the steps:
You must install this browser www.torproject.org/projects/torbrowser.html.en
After instalation,run the browser and enter address **34r6hq26q2h4jkzj.onion**
Follow the instruction on the web-site. We remind you that the sooner you do, the more chances are left to recover the files.
Any attempt to remove or corrupt this software will result in immediate elimination of the private key by the server.

Click to copy Bitcoin address to clipboard
Show encrypted files
Check Payment
Enter Decrypt Key
Click to Free Decryption on site

Figure 2.5: T*eslaCrypt ransom splash screen.*

TeslaCrypt creates a "HELP_RESTORE_FILES.txt" file that contains ransom payment details. Many hundreds or thousands of these files can be found on compromised systems. A ransom message (see Figure 2.5) is saved as "HeLP_ReSTORe_FILeS.bmp" on the desktop and becomes the desktop's background image. Some older versions used the filenames "HELP_TO_SAVE_YOUR_FILES" and "HELP_TO_DECRYPT_YOUR_FILES".

The ransom splash screen is displayed states that the victim's files have been encrypted, show encrypted files button can be seen the list of the encrypted files (Pathak & Nanded, 2016). If the victim attempts to remove or corrupt the software their files will be lost.

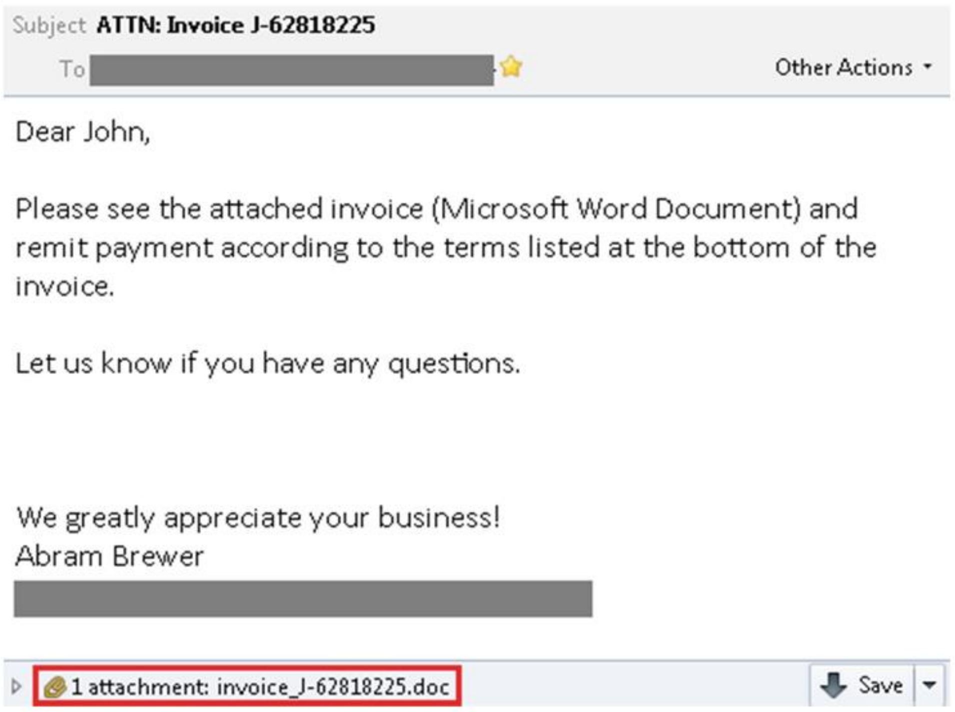
2.4.4 Locky

Locky ransomware is a malicious program that encrypts the user's files on Windows and holds them hostage for ransom. This ransomware is identified as an instance of crypto-locker ransomware. Locky ransomware was discovered at the beginning of 2016 and immediately became one of the most serious ransomware threats. The Locky ransomware spreads more than 100 countries worldwide. The United States and France are the most infected countries, authors concentrate the best of the previous ransomware to achieve a highly skilled threat (Almashhadani, Kaiiali, Sezer, & O'Kane, 2019). Locky was spread through spam email campaigns and exploit kits. There are some famous Locky ransomware variants such as:

- ***Locky Decryptor*** is a variant of locky ransomware that pretends to be a decrypt tool for Locky.
- ***AutoLocy*** the impact of this variant, is not more as the original. This variant spreads through spam emails.
- ***Zepto*** appreaed in June 2016 with malicious email campaigns distribution. This variant encrypts the user's files using AES-128 and RSA-2048 cipher that make it very hard to crack it.
- ***Hucky*** is an abbreviation of Hungarian Locky that encrypts the with desktop picture in Hungarian.

The infection vector starts with a social engineered (SE) email. Ransomware developers send spam email imitating as invoice payment with purchase ordered references as shown in Figure 2.6. The red highlighted section shows the structure of the Locky URL that is needed to be clicked. All of the affected emails consist of an archive (7zip, rar, zip) which has an embedded in VBscript file. After the victim opens the attached document, it'll prompt the victim to enable the Word macros so that its contents can be displayed properly. A macro is somewhat like a shortcut that performs some sort of automated function. When executed, the script connects to command and control (C&C) servers to download the Locky Ransomware, which then encrypts the users' data locally as well as the files on network shares.

The Necurs botnet is identified as another main distributor that spreads the Locky payload infections, usually as a result of a specially-crafted Microsoft Office Word or Excel file with malicious macros, and then enabled. To inspire the victim to enable the macros, a distorted message is displayed with "the data encoding is not correct, please permit macro to be updated" (Prakash, Nafis, & Biswas, 2017). The macro is then downloaded by the Word document and the Locky code starts to encrypts the files on the user's directories and simultaneously renames the all file names and changes the file extension to .locky.


Subject ATTN: Invoice J-62818225

To [redacted]

Dear John,

Please see the attached invoice (Microsoft Word Document) and remit payment according to the terms listed at the bottom of the invoice.

Let us know if you have any questions.

We greatly appreciate your business!
Abram Brewer
[redacted]

1 attachment: invoice_J-62818225.doc

Figure 2.6: Spam email with invoice attackmen

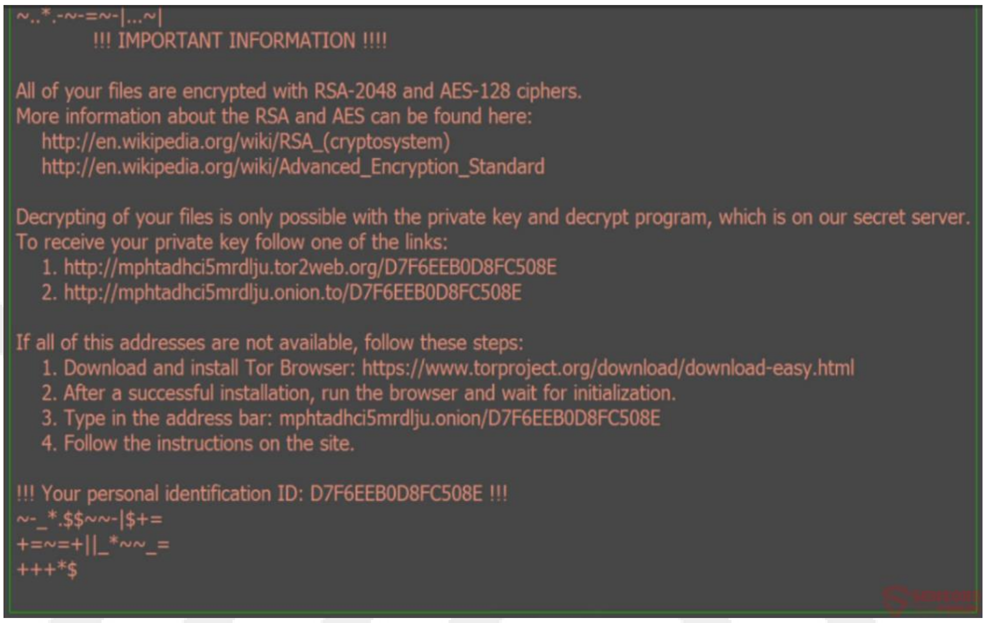
For distribution of the malware, the first versions employed hiding techniques by injecting itself into the windows explorer process, but it was easily identified by some registry keys. Here are some registry keys created:

- ***Id:*** to keep the victim's identity id key is employed
- ***Pubkey:*** for encryption, the Locky needs to store the key in the RSA publicly.
- ***Paytext:*** in the ransom note usually is stored in the registry
- ***Completed:*** if the encryption process is successfully carried out this registery key is used.

Once the malicious payload is delivered into the system, some steps are followed:

- Locky pretends like normal windows executable, it renames itself to svchost.exe to avoid the detection.
- The renamed Locky file starts to delete the backup files and prevent a system from restoring.
- Locky begins to communicate the command and control servers to find the he RSA public key
- Locky generates a unique ID of the victim and saved on the command and control server.

When Locky has finished encrypting the victim’s files, it will remove the downloaded executable and then display a ransom note through instructions in Bitcoin as shown in Figure 2.7, it requires ~0.5 Bitcoin, that is equal to ~$400-500, where the value of Bitcoin was around $900-1000, but in 2017 some variants demanded $900, ransom also went up to $1000.



```
~..*.-~-=~-|...~|
          !!! IMPORTANT INFORMATION !!!!

All of your files are encrypted with RSA-2048 and AES-128 ciphers.
More information about the RSA and AES can be found here:
   http://en.wikipedia.org/wiki/RSA_(cryptosystem)
   http://en.wikipedia.org/wiki/Advanced_Encryption_Standard

Decrypting of your files is only possible with the private key and decrypt program, which is on our secret server.
To receive your private key follow one of the links:
   1. http://mphtadhci5mrdlju.tor2web.org/D7F6EEB0D8FC508E
   2. http://mphtadhci5mrdlju.onion.to/D7F6EEB0D8FC508E

If all of this addresses are not available, follow these steps:
   1. Download and install Tor Browser: https://www.torproject.org/download/download-easy.html
   2. After a successful installation, run the browser and wait for initialization.
   3. Type in the address bar: mphtadhci5mrdlju.onion/D7F6EEB0D8FC508E
   4. Follow the instructions on the site.

!!! Your personal identification ID: D7F6EEB0D8FC508E !!!
~-_*.$$~~-|$+=
+=~=+||_*~~_=
+++*$
```

Figure 2.7: Locky Ransomware Note

2.4.5 Cryptowall

CryptoWall -is also known as Cropto- was discovered in early 2014 and encrypted the user’s files using RSA and employed Onion Router Tor to obfuscate communications with the command and control server. The CyptoWall user interface was quite similar to CryptoLocker. Unlike other ransomware families, the CryptoWall followed a proper software development life cycle. Due to this development, their versions were given version numbers. The infection vector of CryptoWall employed email spams that sent through email and are contained within spam attachments. The email attachment contains a JavaScript code that will download the executable JPG files to harm the victim. It will also generate a new instance of EXPLORER.EXE AND SVCHOST.EXE to make communication with their server. In addition to, to encrypt the victim’s files, first, it will delete the volume shadow copies and installs spyware that steals passwords and BITCOIN wallets. This kind of ransomware has equipped with different types of spyware involved in it (Cabaj, Gawkowski, Grochowski, & Osojca, 2015). To execute and infect the machine, the Cryptowall hide itself by injecting into a legitimate Windows process such as svchost. Several versions of Cryptowall have the capability of identifying the sandbox environment by employing self-protection mechanisms. There is some new version of CryptoWall as describe below:

- **Cryptowall 2.0.** This variant is almost similar to the original Cryptowal. It encrypts the victim's files with RSA-2048 encryption algorithm and seeks to make it victim pay either $500 USD, 500 EUR or 1.22 Bitcoin. Unlike Cryptowall 1.0 version which cannot delete the files, this variant had the ability to securely delete files. To collect and track the victim's payment, Cryptowall 2.0 generated a unique bitcoin address for each of its victims, which was not available in CryptoLocker1.0 (Aurangzeb et al., 2017).
- **Cryptowall 3.0.** to infect the system, this version distributes through Magnitude and Fiesta exploit kits, which is easier than its previous examples. This Version 3.0 employed an RSA 2048-bit public key, that downloaded from the Common and Control domain. For encryption purposes, Cryptowall 3.0 upgraded to AES symmetric encryption with Cipher Block Chaining (CBC) mode which is applied via the Invisible Internet Project (I2P) network.
- **Cryptowall 4.0** -is known as Help Your Files ransomware- was seen first in November 2015. It has a unique characteristic that has not been seen in the previous version of this ransomware. The infection vector of this ransomware was employed via email attachments. Some exploit kits, such as Angler, also distributed it. The subject of these emails was mainly related to job vacancies. This Cryptowall version demands to pay $700 in exchange for the decryption key (Cabaj & Mazurczyk, 2016).

2.4.6 Pgpcoder

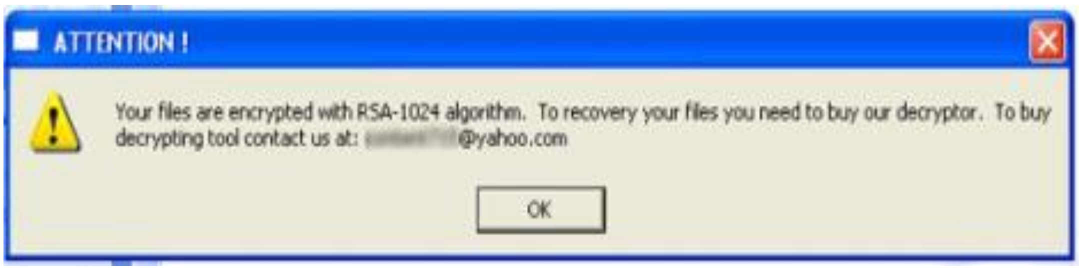
PGPCODER was first appeared in Russia, and it is the oldest ransomware that discovered in 2005. PGPCODER is considered to be the first ransomware that was seen by enforcement. PGPCODER encrypts the files with extension of Vnimanie_.txt in which means Attention in Russian. The early versions of Gpcoder employed a symmetric key that was simply breakable. Because of this, many antivirus engines recovered the victim's files by decrypting the hostage files. This ransomware was distributed through infected websites by the drive-by-download technique. Later in 2010, another updated version of this ransomware was discovered (Jones Jr & Muhammad).

The Initial versions of PGpcoder employed a symmetric key and were easily breakable. Many antivirus vendors could decrypt the encrypted files. But later, the encryption algorithms got stronger and very tough to crack. Some of the encryption algorithms used were RSA1024, AES256, and so on. Ransomware would change the file extension of the original file to something else. This version searches the files with the following extensions, as shown in the following Box 2.4. Gpcoder obtains these extensions to encrypt the file. It starts by reading the contents of the file into memory. Then the ransomware encrypts the contents and writes it into a new file. The new file has a different extension from the original file. The original file is deleted (Tailor & Patel, 2017).

.asc,.db,.db1,.db2,.dbf,.doc,.htm,.html,.jpg, .pgp,.rar,.rtf,.txt,.xls,.zip, .omg, .encoded

Box 2.4: List of file extensions for Pgpcoder ransomware

Victims received an alert to send an email for instructions on how to decrypt the files after payment. This was pretty much the same situation as the 1989 version. It would take files, archive them, and then put a password over the files, but a security researcher cracked the code, and then gave that code to anyone who had encrypted files. The failure in this version was that the ransomware could be removed from the computer's safe mode where one could uninstall, or delete it. "It didn’t tamper with files on the disc, but criminals became more aggressive with trying to get you to pay (Zavarsky & Lindskog, 2016). The average of the payment ranging from $100 -$200, the ransom pop up message states that the victim’s files are encrypted with the RSA-1024 algorithm as shown in Figure 2.8. To access the files the victim needs to by the decryptor tool through normal email like yahoo.


ATTENTION !

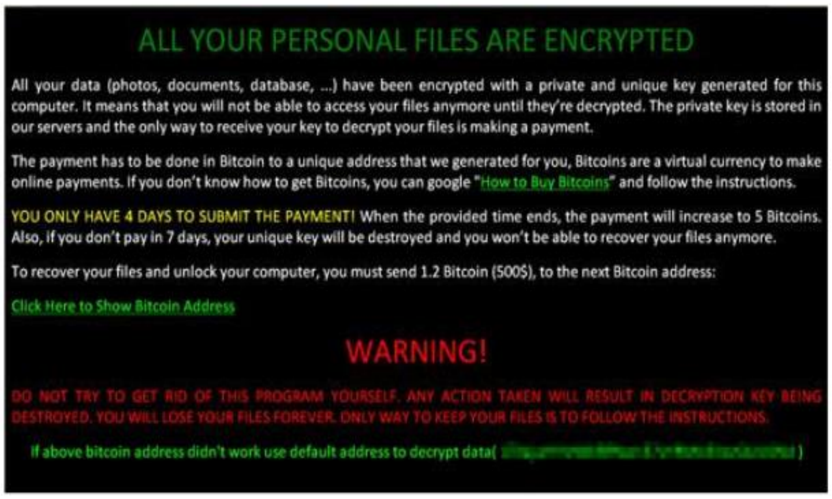
Your files are encrypted with RSA-1024 algorithm. To recovery your files you need to buy our decryptor. To buy decrypting tool contact us at: [redacted]@yahoo.com

Figure 2.8: PGpcoder ransom splash screen.

2.4.7 Zcryptor

Zcryptor is a classic ransomware that encrypts the user's file like the other ransomware. However, it has a special feature that should not be overlooked. This ransomware is also spread via USB keys and hard drives. Once it has infected a computer via traditional means (social engineering, email, spam, ...), it starts to execute the payload by using the masquerading as an installer of the program. It detects whether there are any computer media (USB keys, hard drives) that are connected to the computer. If it finds one, it will copy the files and make them invisible to the user. Then, as soon as the device is connected to a new computer(O'Kane et al., 2018). The ransomware will infect it.

Once the payload is delivered on the system and executed it. The ransomware creates a registry key that make sure that it could run at start-up, then it drops the autorun.inf on removable drives, along with a *zycrypt.lnk* in the start-up folder. After that, the ransomware generates a hidden copy of itself under the system folder of the registry (Roberts, 2018). as *Drive:\system.exe* and *%appdata%\zcrypt.exe*.



ALL YOUR PERSONAL FILES ARE ENCRYPTED

All your data (photos, documents, database, ...) have been encrypted with a private and unique key generated for this computer. It means that you will not be able to access your files anymore until they're decrypted. The private key is stored in our servers and the only way to receive your key to decrypt your files is making a payment.

The payment has to be done in Bitcoin to a unique address that we generated for you, Bitcoins are a virtual currency to make online payments. If you don't know how to get Bitcoins, you can google "How to Buy Bitcoins" and follow the instructions.

YOU ONLY HAVE 4 DAYS TO SUBMIT THE PAYMENT! When the provided time ends, the payment will increase to 5 Bitcoins. Also, if you don't pay in 7 days, your unique key will be destroyed and you won't be able to recover your files anymore.

To recover your files and unlock your computer, you must send 1.2 Bitcoin (500$), to the next Bitcoin address:

Click Here to Show Bitcoin Address

WARNING!

DO NOT TRY TO GET RID OF THIS PROGRAM YOURSELF. ANY ACTION TAKEN WILL RESULT IN DECRYPTION KEY BEING DESTROYED. YOU WILL LOSE YOUR FILES FOREVER. ONLY WAY TO KEEP YOUR FILES IS TO FOLLOW THE INSTRUCTIONS.

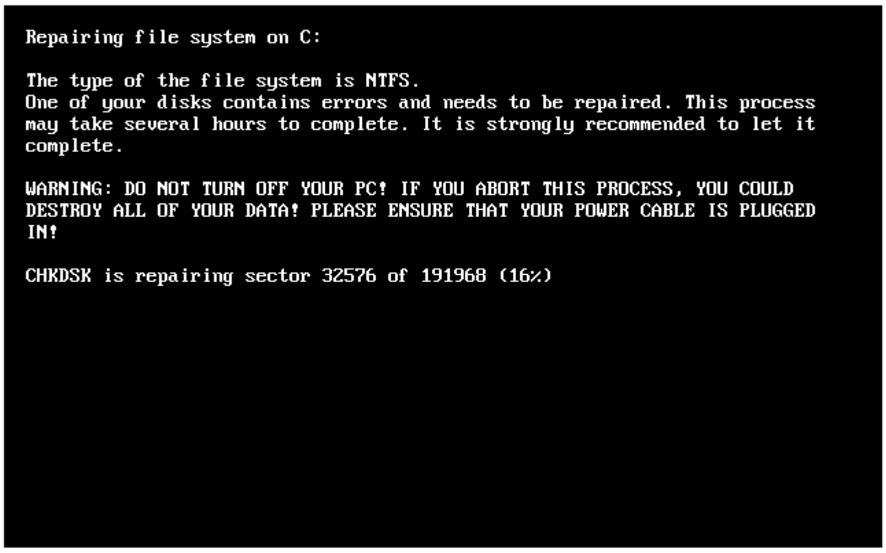
If above bitcoin address didn't work use default address to decrypt data([redacted])

Figure 2.9: Zcryptor ransom not

After the encryption, Zcryptor displays a message that looks like a normal HTML page stating that their files are encrypted and demanded a ransom ranging 1.2 bitcoins (average of $650) as shown in Figure 2.9. If the victim did not pay the required money in four days, the ransom increases to 5 bitcoins (more than $2,500).

2.4.8 Petya

The Petya was first discovered in May 2016 that encrypts the Master Boot Record (MBR) of the windows. After execution, Petya encrypts the user's files by overwriting the Master Boot Record with a malicious payload and then boot the windows system with extensions as shown in Box 2.5. then the victim has redirected the boot screen that demands a ransom. It also encrypts the Master File Table (MFT) system drive that causes the system to reboot and displays the blue screen of death (BSOD). After restarting the machine, Petya generates a fake CHDISK screen as shown in Figure 2.10. This is made by the boot-loader that has replaced the original MBR (Aidan, Verma, & Awasthi, 2017). This boot-loader further encrypts MFT in the background while the CHKDISK screen is shown to the victim.



```
Repairing file system on C:

The type of the file system is NTFS.
One of your disks contains errors and needs to be repaired. This process
may take several hours to complete. It is strongly recommended to let it
complete.

WARNING: DO NOT TURN OFF YOUR PC! IF YOU ABORT THIS PROCESS, YOU COULD
DESTROY ALL OF YOUR DATA! PLEASE ENSURE THAT YOUR POWER CABLE IS PLUGGED
IN!

CHKDSK is repairing sector 32576 of 191968 (16%)
```

Figure 2.10: fake CHDISK screen

When the computer is on, the first program to run the is the Basic Input Output System (BIOS). This BIOS performs the Power on Self (POST) test and reads the Master Boot Record (MBR). POST checks whether all hardware devices are connected to the system for the proper functioning of the system. The BIOS then reads the MBR. MBR refers to the first sector of a partition known as the volume boot record (VBR). VBR contains a lot of information, such as partition size and partition type (Aurangzeb et al., 2017). If the partition type is NTFS (New Technology File System is the file system used by Windows), VBR contains information about the Master File Table (MFT). The space of the MFT is the kept by the NTFS file system, where all information about a file, including its size, time and date, permissions, and data content, is stored in MFT entries, or in a space outside the MFT described. through MFT inputs. Because of the MFT is encrypted, Windows cannot identify the file system and therefore it cannot load the rest of the operating system components.

.3ds, .asp, .bak, .cpp, .disk, .7z, .aspx, .accdb, .avhd, .c, .cfg, .cs, .ctl, .djvu, .zip, .work, .vmsd, .xls, .vmx, .xlsx, .vsdx, .vfd, .vbox, .rtf, .pvi, .pmf, .vdi, .tar, .rar, .pst, .php, .ost, .mdb, .ova, .msg, .vmc, .vbs, .sln, .py, .ppt, .ovf, .hdd, .nrg, .kdbx, .fdb, .dwg, .h, .eml, .doc, .ai, .back, .conf, .dbf, .docx, .gz, .mail, .ora, .pdf, .pptx

Box 2.5: Petya Encryption file extensions

Petya uses an EternalBlue and SMB as propagation methods to spread within an infected network. Another infection method was spread via spam emails that show as a resume with the malicious attachment (Richardson & North, 2017). There are some updated versions of Petya ransomware such as Petya-Mischa and goldeneye Petya.

- ***RED-PETYA*** is a new version of the original Petya that has all functions mentioned above. The RED-Petya is identified to be the first version that employs Salsa20, asymmetrical key algorithm, to encrypt the MFT and make the drives unavailable. However, the execution of the algorithm is not implemented correctly, resulting in an unintended error that enables to decrypt of the infected files, although the bugs of the later version of the Red-Petya has been fixed.
- ***PETYA-MISCHA*** also is known as GREEN-PETYA, was first discovered in September 2016. This Petya-Mischa consists of two components, the Petya and the MISCHA. To execute the payload, the Petya needs administrator rights to run; otherwise, it fails. If the Petya-Mischa failed to encrypts MF, MISCHA encrypts the victim's personal files, and displays a splash green screen; this version was therefore called GREEN-PETYA.
- ***PETYA-GOLDENEYE*** is a later version of Petya that spread in Germany in December 2016. GoldenEye is discovered to be the next version of Petya-Mischa. Unlike the Petya-Mischa, Goldeneye encrypts the files in the file on the hard disk first, then encodes the MFT. After encoding the files, GoldenEye adds a string that consists of eight random characters to the end of every file name. The distribution of the Petya- GoldenEye through the spam emails that seemed to be legitimate emails. Unlike the Petya-Mischa that requires the administrator rights for encryption of the MFT, the GoldenEye acquired the administrator rights (Zakaria, Abdollah, Mohd, & Ariffin, 2017).

2.4.9 Cerber

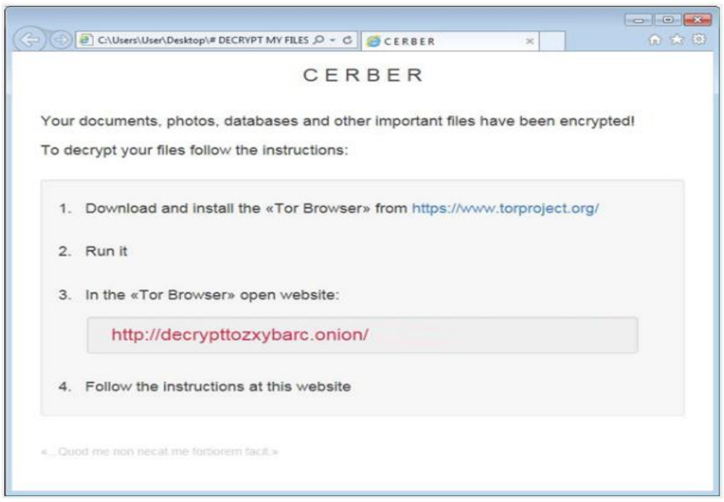
Cerber is a ransomware infection that is used to encrypt the victims' files. This ransomware was considered to be allegedly Locky's twin and creates the extension of CERBER to every file that the ransomware encrypts. Cerber appeared in 2016 that employed the AES encryption algorithm. This kind of ransomware earned for a mature profit of $ 1 million to $ 2.5 million, and the ransomware author has reported receiving 40% of the ransom. This attack encouraged by the underground Russian forums that allow everyone to buy Cerber, which makes ransomware attacks possible from anyone with enough money to pay (Kara & Aydos, 2018).

Cerber was distributed through Botnet, exploit kits, and email spam that contains the word document attachment that looks like invoices. If the victim opens the attachment, he/she will receive a message states that the document has the wrong coding. The victim is interested in activating the Word macros that encoded VBScript and then execute it. The VBScript downloads the payload of the Cerbe and enforces the computer talk to the victim by reading a load message warning. The malicious code is delivered to the victim's machine, and the attack begins (Wyke & Ajjan, 2015).

There are some other new versions of Cerber that have avoided security solutions by dividing the code into minor pieces of code. Without dropping the components onto a physical disk, these small pieces were extracted and read in their own process. This gives the advantage of not scanning the reader with an antivirus engine. This version has a configuration file as JSON format representation. The configuration file offers the ability to update itself easily, and improves the functionality of the Cerber. The encryption type and the target files are dictated by this configuration file while executing its activities (Kurniawan & Riadi, 2018). Cerber uses the following configuration files:

- ***Blacklist:*** states that Cerber needs to encrypt the identified folder and decides which countries should be infected.
- ***Close_process:*** indicates which processes must be eliminated.
- ***Encrypt - keys:*** Cerber employed this file for encryption purposes.
- ***Help_files:*** identifies the type of a ransom note that will be shown to the victim.
- ***Self_deleting:*** decides whether the ransomware should delete itself after execution.
- ***Whitelist:*** the number of the files extensions that are targeted by the ransomware.

Unlike Pgpcoder, once the files are encrypted by the Cerber's is not easy to break the encryption key. There is no guarantee that victim's files can be obtained back even if the victim paid the required ransom to the attackers' criminal for the intended limited time.



C:\Users\User\Desktop\# DECRYPT MY FILES

CERBER

CERBER

Your documents, photos, databases and other important files have been encrypted!

To decrypt your files follow the instructions:

1. Download and install the «Tor Browser» from https://www.torproject.org/
2. Run it
3. In the «Tor Browser» open website:

   http://decrypttozxybarc.onion/

4. Follow the instructions at this website

«...Quod me non necat me fortiorem facit.»

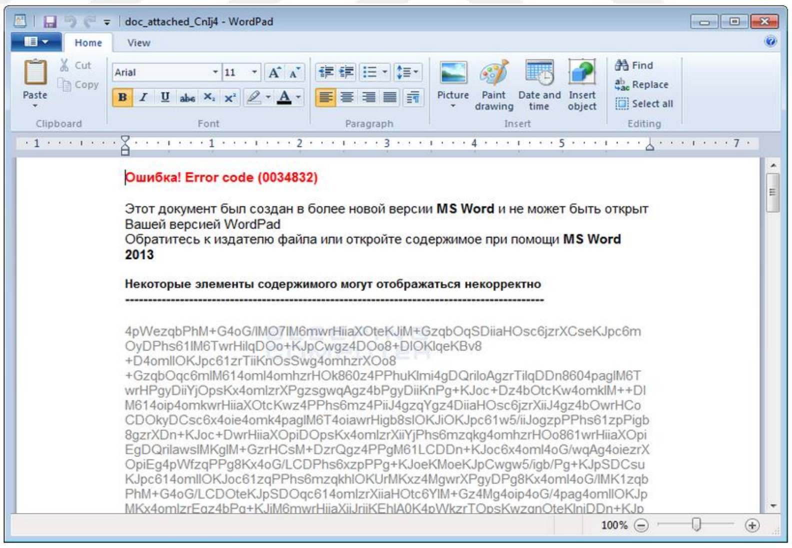
Figure 2.11: Text file displayed by the Cerber ransom not.

The ransomware note is generated into several ways: as an audio message, as a text file dropped into a folder containing the compromised data, the same file is used to generate a screensaver and desktop wallpaper. The message is written that the victim's documents, photos and database files have been encrypted. To decrypt these files users, need to download the Tor Browser and run it, then following the instruction provided by the website as shown in Figure 2.11. Unlike other ransomware families, this variant doubles the ransom bitcoins from 1.24 to 2.48 after seven days of non-payment (Adamov & Carlsson, 2017).

2.4.10 RAA (JS / RANSOM-DLL)

RAA (JS / RANSOM-DLL) is a ransomware variant that is distributed via email attachment. RAA is written entirely in JavaScript (a programming language). One of the advantages of using JavaScript is that Windows does not display these extensions by default. This kind of ransomware pretends like normal attachment that should display "invoice.js", but the user can see only "invoice". RAA victims do not care about the extension since it is not visible. RAA (JS / RANSOM-DLL) does not need to download ransomware to the server. When the malicious payload is executed in the victim's system, RAA creates a fake word document under the %MyDocuments% folder. The name of the malicious word document will have similar to doc_attached_CnIj4 and will automatically pretend that the opened word attachment was corrupted. While the victim's intention is drifted, the attachment is corrupted, RAA Ransomware is executed in the background, then begins to scan all the computer drives and decides if the victim has read and writes access to them (Misini, 2018). As soon as the ransomware has infected the victim it is ready to encrypt the data and demand a ransom. RAA does some steps:

- RAA (JS / RANSOM-DLL) launches a decoy file that contains a message which is mainly used to distract attention. The ransomware makes a call to the server to request an encryption key.
- The server provides an AES random encryption key as well as an identifier (public key).
- As soon as the data is encrypted, the victim will have a message demanding to pay the ransom so he can recover the corresponding AES key for decryption.



Ошибка! Error code (0034832)

Этот документ был создан в более новой версии **MS Word** и не может быть открыт Вашей версией WordPad
Обратитесь к издателю файла или откройте содержимое при помощи **MS Word 2013**

Некоторые элементы содержимого могут отображаться некорректно
--

```
4pWezqbPhM+G4oG/lMO7lM6mwrHiiaXOteKJiM+GzqbOqSDiiaHOsc6jzrXCseKJpc6m
OyDPhs61lM6TwrHilqDOo+KJpCwgz4DOo8+DlOKlqeKBv8
+D4omllOKJpc61zrTiiKnOsSwg4omhzrXOo8
+GzqbOqc6mlM614oml4omhzrHOk860z4PPhuKlmi4gDQriloAgzrTilqDDn8604paglM6T
wrHPgyDiiYjOpsKx4omlzrXPgzsgwqAgz4bPgyDiiKnPg+KJoc+Dz4bOtcKw4omklM++DI
M614oip4omkwrHiiaXOtcKwz4PPhs6mz4PiiJ4gzqYgz4DiiaHOsc6jzrXiiJ4gz4bOwrHCo
CDOkyDCsc6x4oie4omk4paglM6T4oiawrHigb8slOKJiOKJpc61w5/iiJogzpPPhs61zpPigb
8gzrXDn+KJoc+DwrHiiaXOpiDOpsKx4omlzrXiiYjPhs6mzqkg4omhzrHOo861wrHiiaXOpi
EgDQrilawslMKglM+GzrHCsM+DzrQgz4PPgM61LCDDn+KJoc6x4oml4oG/wqAg4oiezrX
OpiEg4pWfzqPPg8Kx4oG/LCDPhs6xzpPPg+KJoeKMoeKJpCwgw5/igb/Pg+KJpSDCsu
KJpc614omllOKJoc61zqPPhs6mzqkhlOKUrMKxz4MgwrXPgyDPg8Kx4oml4oG/lMK1zqb
PhM+G4oG/LCDOteKJpSDOqc614omlzrXiiaHOtc6YlM+Gz4Mg4oip4oG/4pag4omllOKJp
MKx4omlzrEqz4bPq+KJiM6mwrHiiaXiiJriiKEhlA0K4pWkzrTOpsKwzqnOteKlniDDn+KJp
```

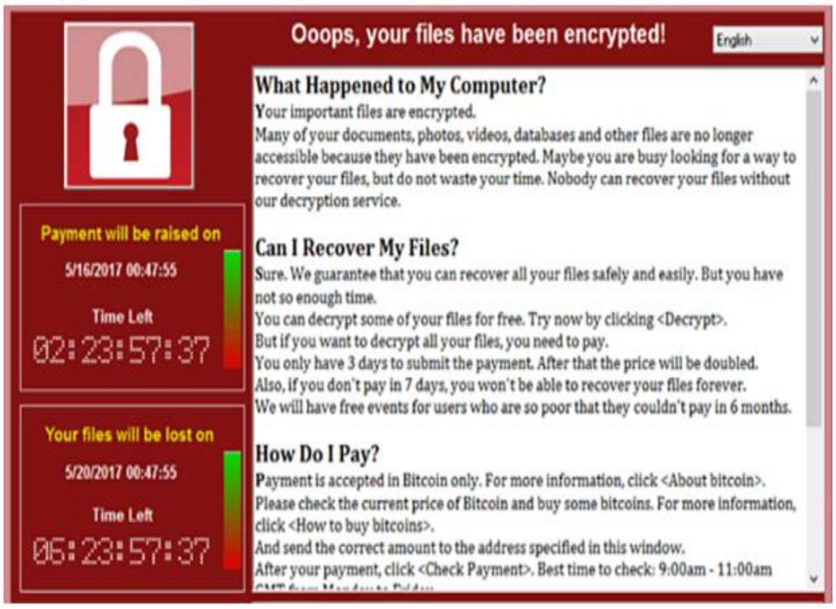
Figure 2.12: The Fake attachment of the RAA ransomware

The AES encryption key is not kept in memory by RAA (JS / RANSOM-DLL), as soon as the encryption is finished, it is deleted, so only the server has a copy of the key for decryption. Once the encryption is complete, there will be, as with other ransomware, a procedure to explain how to recover the data. RAA (JS / RANSOM-DLL) also installs a password thief (Misini, 2018). The codename for this virus is Troj / Fareit-AWR. It is stored in the My Documents directory with the following name: st.exe.

2.4.11 WANNACRY

WannaCry is a crypto-ransomware worm that encrypts the user's related files and the hard drive to extort money. Then demands a ransom payment of \$300-\$600 in bitcoin in order to decrypt them. WannaCry spread rapidly and become a huge global outbreak. It infected a number of computer networks especially in a Windows computer. The WannaCry is also known as Wannacrypt and Wcry as refers to its extension. May 15, 2017, this variant of ransomware has emerged, which is a kind of ransomware that targets all kinds of files including PDF files, word documents, excel sheets, etc, and encrypts them in the form of .wcry extension. Program files with the extension .cpp and its source code is also encrypted by the Wannacry. The WannaCry causes crises across the world and infecting vulnerable systems globally (Al-rimy et al., 2018).

WannaCry cyber-attack has been reported in 99 countries and over 75,000 attacks have been carried out on machines running the Windows operating system (Chen & Bridges, 2017). The most affected organizations are Telefonica, a Spanish broadband and telecommunications provider, UK's National Health Service (NHS), FedEx, LATAM Airlines and Deutsche Bahn.



Ooops, your files have been encrypted!

Payment will be raised on
5/16/2017 00:47:55
Time Left
02:23:57:37

Your files will be lost on
5/20/2017 00:47:55
Time Left
06:23:57:37

What Happened to My Computer?
Your important files are encrypted.
Many of your documents, photos, videos, databases and other files are no longer accessible because they have been encrypted. Maybe you are busy looking for a way to recover your files, but do not waste your time. Nobody can recover your files without our decryption service.

Can I Recover My Files?
Sure. We guarantee that you can recover all your files safely and easily. But you have not so enough time.
You can decrypt some of your files for free. Try now by clicking <Decrypt>.
But if you want to decrypt all your files, you need to pay.
You only have 3 days to submit the payment. After that the price will be doubled.
Also, if you don't pay in 7 days, you won't be able to recover your files forever.
We will have free events for users who are so poor that they couldn't pay in 6 months.

How Do I Pay?
Payment is accepted in Bitcoin only. For more information, click <About bitcoin>.
Please check the current price of Bitcoin and buy some bitcoins. For more information, click <How to buy bitcoins>.
And send the correct amount to the address specified in this window.
After your payment, click <Check Payment>. Best time to check: 9:00am - 11:00am

Figure 2.13: Wannacry splash screen

Table 2.1: Ransomware Families

Family	Year	Encryption	CC	Infection Vector
REVETON	2012	No	No	Blackhole Exploit kit
CRYPTLOCKER	2013	Symmetric AES	Static Domain	Spam Emails
TESLACRYPT	2015	Symmetric AES-256 and CBC	Static and DGA domain	Phishing and Spam emails
LOCKY	2016	Symmetric AES	DGA domain	Spam
CRYPTOWALL	2014	Asymmetric RSA-2048	Static Domain	Spam and Malvertising
CTBLOCKER	2015	Symmetric AES	No contact	Spam Emails
TORRENTLOCKER	2015	Symmetric AES	Static Domain	Spam Attachment
RAA	2016	Symmetric AES	Static Domain	Spam Emails
CERBER	2016	Symmetric AES	Static Domain and no contact	Spam and Fake software
BART	2016	Symmetric AES	No contact	HTML attachments
PETYA	2016	Symmetric AES	Static Domain	Worm Fake software
NOTPETYA	2017	Symmetric AES	No contact to	EternalBlue
WANNACRY	2017	Symmetric AES	Static Domain	EternalBlue
LOCKER	2017	Symmetric AES	Static Domain	Spam and compromised websites
BADRABBIT	2017	Symmetric AES	No contact	Compromised websites
RYUK	2018	Symmetric AES	No contact	Directed attack to business
KATYUSHA	2018	Asymmetric RSA-2048	Static Domain	EternalBlue
GANDCRAB	2018	Symmetric AES	Static Domain	Fake software crack
LOCKERGOGA	2019	Asymmetric RSA-4096	No contact	worms

The attack vector of the WannaCry compromise through TCP /445 (SMB), since the ransomware uses a worm that lies in the Windows implementation of vulnerabilities of the Server Message Block (SMB) protocol, it then attempts to exploit those systems employing the EternalBlue exploit. If a machine is successfully exploited, WannaCry gains access to the machine. To deploy the WannaCry into the exploited machine, it uses a tool called DoublePulsar. This tool represents a backdoor for delivering the infection. TOR client is implemented by the WannaCry to communicate with its CnC (Da-Yu et al., 2019). To encrypt the user's files, WannaCry follows the phases:

- WannaCry created a public/private RSA-2048 key pair representation.
- A public key is fixed wannacry payload; however, the private RSA key is not intended to send the victim.

- A random AES key is created for each key on the victim system that needs to be encrypted.
- Wannacry thus does not require to contact its Common & Control to encrypt files.
- The extension .wcry or wannacry for every encrypted files.

Microsoft has released a security patch that protected user's machine against these Eternalblue patches. A patch, distributed on time, could have saved the wannacry epidemic. Unfortunately, several organizations and individuals do not often update their operating systems that cause to be exposed to the attack. WannaCry attempts to escape from the sandboxes by reaching a non-existent domain. When the ransomware tries to contact to its CnC, sandboxes provide a false positive alarm and stop the malware (Tailor & Patel, 2017).

2.5 Ransomware Payment Using Bitcoin

Ransom payment is vary depending on the type of the ransomware variant and the worth digital currencies rates. Ransomware authors normally determine the ransom payments in bitcoins (Morato, Berrueta, Magaña, & Izal, 2018). The amount of all ransoms are of equal value. When the attack is not targeted and it targets a large number of companies, the ransoms can be low, for amounts between 200 and 500 dollars, for example. Conversely, if the hackers attempted to target a particular company, they would not hesitate to demand a larger ransom amount, up to several tens of thousands of dollars. The newly emerged ransomware families provide a liste of payment options such as iTunes and Amazon gift card. As the researchers recommend that paying the ransom does not guarantee that the victim will get back the decryption key to access to the infected system or the encrypted files (Palisse et al., 2016).

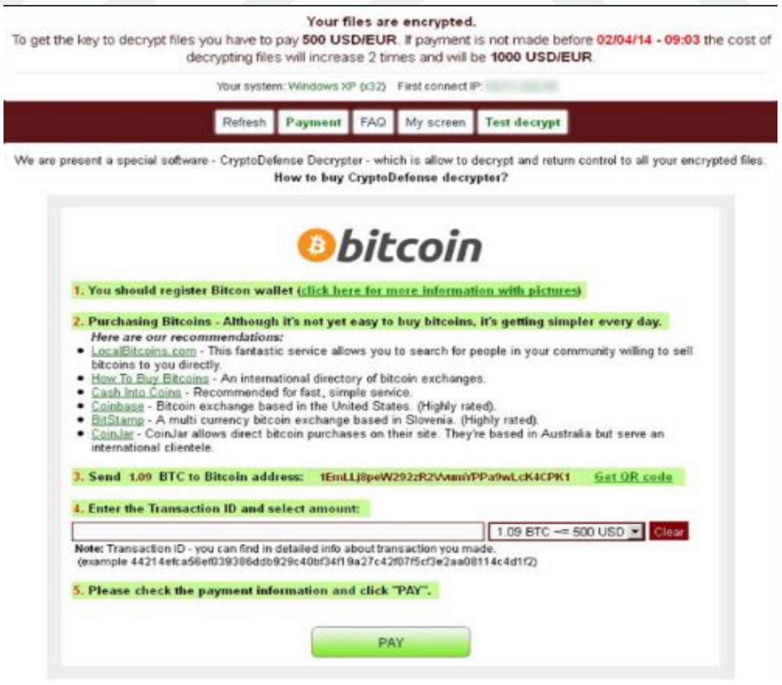
The most popular cryptocurrencies are bitcoin. It is an electronic currency (there are no notes or coins) invented by Satoshi Nakamoto in 2008. It has the particularity of not depending on any central bank or central authority . Bitcoin is therefore decentralized; its value is assessed by the supply and demand of websites that serve as trading places for bitcoin. This means that each transfer is made directly between users without going through a third-party authority.

The price of bitcoin is very volatile, for example if you have to pay 1 bitcoin (1 btc = 4,900 euros), the price of Bitcoin is continually fluctuating, in 2016 the value of the one 1 BTC was $400. This rate was dramatically changed to $7,000 in February 2018.

Tomorrow a bitcoin could be worth more than 7,500 or 8,500 euros. It all depends on supply and demand. Bitcoin transactions use asymmetric encryption.

There is a private key and a public key. The private key serves as a password and is used to sign transaction messages. The public key is the account number of the user, i.e. the wallet. Then an address is created from the public key, which itself is created from the private key. Users can use a new address for each new payment. This allows the transactions to be separated so that it is not possible to associate them. So, if someone sends you bitcoins, they can't see your other addresses. Addresses cannot be associated. For all these reasons bitcoin has become ubiquitous in the cybercrime world.

The financial transactions that are executed and completed are saved in a common and transparent record, called the blockchain, that is widely accessible. The input becomes transaction and a list of outputs, each number of Bitcoins transferred to an explicit receiver's address. A Bitcoin address is an alphanumeric string derived from the public key of an asymmetric key pair generated by a Bitcoin user. Every user has many addresses and key pairs representation in the wallet, this allow to rent a new address for each transaction to add the level of anonymity (Corrigan, 2017).



Your files are encrypted.

To get the key to decrypt files you have to pay **500 USD/EUR**. If payment is not made before 02/04/14 - 09:03 the cost of decrypting files will increase 2 times and will be **1000 USD/EUR**.

Your system: Windows XP (x32) First connect IP: [redacted]

We are present a special software - CryptoDefense Decrypter - which is allow to decrypt and return control to all your encrypted files.

How to buy CryptoDefense decrypter?

1. You should register Bitcon wallet (click here for more information with pictures)

2. Purchasing Bitcoins - Although it's not yet easy to buy bitcoins, it's getting simpler every day.

Here are our recommendations:

- LocalBitcoins.com - This fantastic service allows you to search for people in your community willing to sell bitcoins to you directly.
- How To Buy Bitcoins - An international directory of bitcoin exchanges.
- Cash Into Coins - Recommended for fast, simple service.
- Coinbase - Bitcoin exchange based in the United States. (Highly rated).
- BitStamp - A multi currency bitcoin exchange based in Slovenia. (Highly rated).
- CoinJar - CoinJar allows direct bitcoin purchases on their site. They're based in Australia but serve an international clientele.

3. Send 1.09 BTC to Bitcoin address: 1EmLLj8peW292zR2VvamYPPa9wLcK4CPK1 Get QR code

4. Enter the Transaction ID and select amount:

1.09 BTC ~= 500 USD Clear

Note: Transaction ID - you can find in detailed info about transaction you made.
(example 44214efca56ef039386ddb929c40bf34f19a27c42f07f5cf3e2aa08114c4d1f2)

5. Please check the payment information and click "PAY".

Figure 2.14: Ransom not through Bitcoins

Bitcoin performs the same functions as traditional currency. It represents a unit of account, and measures the usefulness of a good or service. It also facilitates trade; it can be used, just like normal money, to buy goods and services. It can also serve as a store of value, and therefore can be used to buy goods and services in the future. In this sense, Bitcoin, which is the first currency without a bank, can be considered a currency. However, there are differences between Bitcoin and a normal currency like a dollar. The dollar is legal tender that is recognized by the governments, and persons or institutions must be paid as hard letters of a dollar,when no one is forced to accept payment in Bitcoin. Another disadvantage is that there is no guarantee that it will be accepted in the future.

In addition, unlike the dollar for example, whose excessive fluctuations as shown in the below Figure 2.15, upwards or downwards, can be regulated by the government, Bitcoin is based on a decentralized system, that does not under the control of no authority, it does not belong to a person, a government, or a company, and therefore its value depends only on supply and demand, which may explain its high volatility. In addition, in a centralized system, the use of normal money is framed by rules, which allows bank customers to be reimbursed when someone fraudulently uses a bank card. In a decentralized system, there is no legal recourse possible, even if users can take out insurance in order to obtain compensation from specialized organizations, in the event of problems (Paquet-Clouston, Haslhofer, & Dupont, 2019).

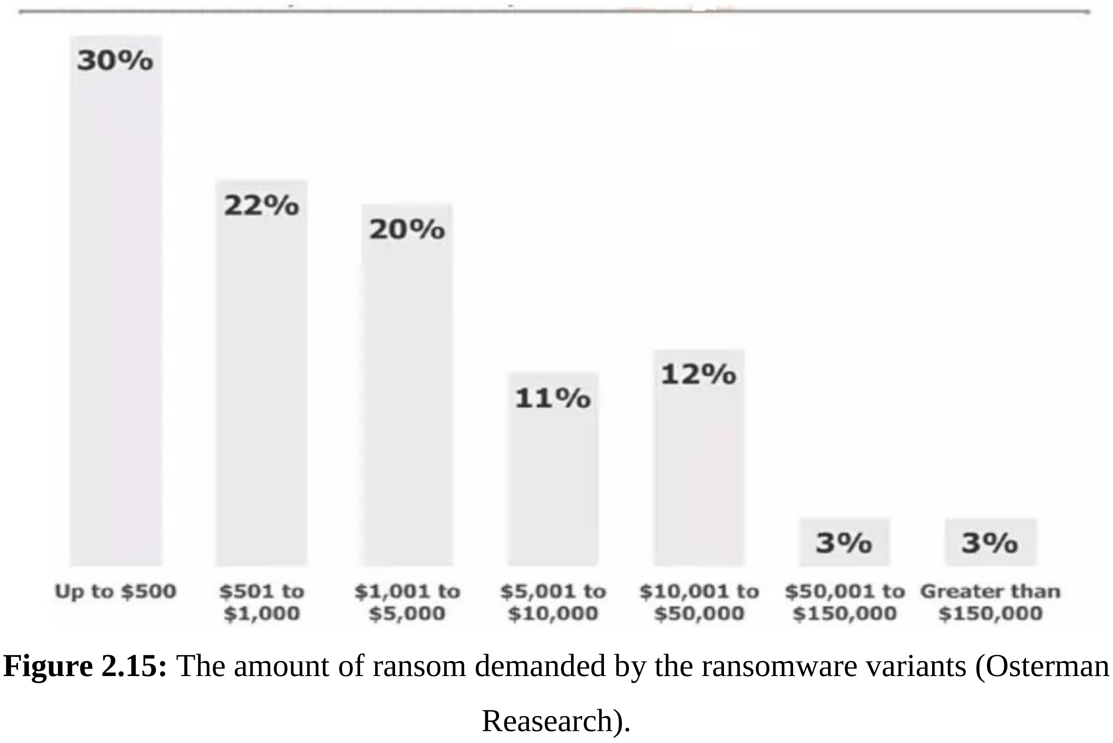


Figure 2.15: The amount of ransom demanded by the ransomware variants (Osterman Reasearch).

Current ransomware uses cryptocurrency more precisely Bitcoin to pay ransoms. But before using the cryptocurrency the ransomware requested to settle the ransom with other payment methods. At the beginning the victims of ransomware could pay the ransoms by sending an SMS via a code like for example those which one can use to pay a TPG bus ticket. Victims could also pay the ransom to an electronic wallet such as PayPal. Police authorities and security experts found a solution when changing the regulations for electronic payments. Using electronic wallets has become less profitable and much riskier for ransomware which displays their number decrease at that time. A few years ago, the cryptocurrency began to become popular among individuals but also among cybercriminals (Conti, Gangwal, & Ruj, 2018).

2.5.1 The technical aspects of bitcoin

One of the fundamental innovations of Bitcoin is that it relies on the network "Peer-to-peer", a computer network where users are connected to each other via nodes and on which they can exchange electronic money directly and free of charge, without going through a third-party entity. Users can contribute to the power of the network by putting the computing power of their computers to the benefit of the bitcoin network. Another major invention specific to Bitcoin is that transactions are encrypted using asymmetric cryptography, that is, by a public key / private key system (Conti et al., 2018).

The signatures of the bitcoin transactions all derive from the ECDSA public key, one of the safest algorithms at present. The address of the public key is used to send bitcoins while the private key is used to receive or pay for it. The third revolutionary singularity alone, is the Blockchain. The "blockchain" is the underlying innovation bitcoin, "a technology for storing and transmitting information at minimal cost, secure, transparent and functioning without a central control body". More intuitively, a "blockchain" is a distributed and secure database allowing any user of the protocol to observe all of the transactions almost instantly and to be able to check the validity. By analogy, a Blockchain can therefore be compared to a public ledger, anonymous and falsifiable. This innovation solves the problem of double-spending, which is to say, that a bitcoin can be used twice by the same person. In the traditional banking system that we know, each bank keeps the database of all its clients individually and privately and other economic players, such as regulatory institutions such as FINMA, do not have access to this information (Paquet-Clouston et al., 2019).

Whereas with normal money it is possible to make counterfeit money, with bitcoin, this problem is solved thanks to Blockchain. Since the first transaction, at 6:15 p.m. on January 3, 2009, there have never been any fake bitcoins in circulation. When a transaction is sent over the Bitcoin network, it is randomly routed to one node, then redirected to all other nodes on the network. The transactions are grouped in what are called blocks. So that the protocol can verify transactions, "secure the network but also allowing all users of the system to stay synchronized ", it takes computing power, and this is called mining. Miners are certain members of the network who have requested to use the computing power of their machines from the network. In addition to validating data relating to the transactions which will form a block, minors must solve very complex mathematical problems. The complexity of these problems varies constantly depending on the number of miners connected. This is why the SHA-256 hash algorithm is an integral part of the bitcoin protocol, in order to constantly adapt the difficulty of the problem, which allows maintaining a validation time by block close to 10 minutes(Conti et al., 2018).

In addition to the complexity of the problems, their resolution depends on a random variable, which makes it possible to ensure that it is not always the same minor which validates all the blocks. This is one of the essential points that secure the Blockchain. When the problem is resolved by one of the users, it must provide proof that it is the one who has the solution. This is called proof of work or proof of calculation.

The block validated by this minor is then temporarily inserted into the Blockchain, so that other users can authenticate the proof of work. If the distributed consensus differs according to the users, we can end up with two different block chains, and parallel. In a case, the rule is that, all nodes on the network must keep the two chains and, however, that one of the two chains are working. Very quickly, one of the two channels will take precedence over the other. This is why it is said that it takes an hour on average before a block is irreversibly confirmed, which is equivalent to waiting for five to six other blocks to be integrated into the Blockchain (Conti et al., 2018).

Minors are paid in bitcoins specially created for block validation and according to the computing power they bring to the network: this is a strong incentive to contribute to the computing power of the protocol. To make the process fair, a random variable is introduced in the protocol to validate a block. This is the only way to create new bitcoins and therefore the only way to increase the money supply. As an anecdote, the reason they are called "miners" refers to the gold diggers who increased the money supply as they discovered it. Initially, in 2009, the remuneration was 50 bitcoins per block inserted in the Blockchain. The rule is that the remuneration is halved every 210,000 blocks created, and the process that takes approximately four years.

Today, miners touch 25 bitcoins per block and this Figure 2.15 will soon be halved again because we soon reached 203,000 blocks. The mechanism is thus made that the reward for mining bitcoins will decrease over time, and therefore, in the end, the bitcoin money supply is predictable. We know that it will converge to 21 million bitcoins. Each bitcoin is divisible up to 100 millionth. This is particularly interesting, because this mechanism is known in advance and completely transparent (Pletinckx, Trap, & Doerr, 2018).

2.5.2 The advantages of using a digital currency

Cryptocurrencies are innovative, first of all for individuals, because they offer a new form of money and payment. Take the case of bitcoin: bitcoin allows simplified payments via a mobile phone. A credit card is no longer necessary, just scan the recipient's QR code or put the two phones against each other to perform a value exchange, using contactless technology (NFC). Second, security and control over money are greatly enhanced. The cryptography used in the bitcoin protocol is made up of the most secure algorithms currently available: ECDSA and SHA-256. As a result, no trusted third party can puncture a user's account or impersonate their identity, or make payments on their behalf: some possible fraud with the Current payment system are therefore avoided thanks to bitcoin (Doguet, 2012).

Other innovation: it is a universal currency, usable worldwide and at all times. There is no longer a need to have the same service provider, since it is interoperable. In addition, since it has no physical existence, Bitcoin allows international transfers, regardless of the amount, immediately: bitcoins can be transferred from one end of the world to the other in less than one ten minutes and at almost zero cost. If service providers like Western Union currently take a 10% commission on money transfers, bitcoin just helps. Ultimately, bitcoin allows individuals to pay anonymously, in digital cash. As for entrepreneurs, in addition to benefiting from all the advantages cited for individuals, they no longer need to submit to PCI regulations, controlling and securing online payments, which is now at their expensein the current system(Conti et al., 2018).

Bitcoin is also a new market, with customers looking to use their new currency. Using bitcoins is also a way for entrepreneurs to stand out from the competition and gain visibility. In addition, Bitcoin allows more transparency within a company, since, in its protocol, transactions are only carried out after all the persons authorized to sign have actually done so, thanks to the multi-signature function. And, finally, as said before, thanks to Blockchain, the company's accounting is completely transparent and tamper-proof, which makes it possible to gain the confidence of investors (Rogojanu & Badea, 2014).

2.6 Ransomware Infection Vectors

Ransomware creators use a range of different sophisticated techniques to spread their malicious intents; in this section, we highlight the most prevalent ransomware propagation methods depicted below:

2.6.1 Spam emails

Electronic mail (e-mail) is an important communication for millions of people including governments, health care groups, institutions, and organizations. E-mail have been widely used by many people in different purpose. At the same time, e-mail is one of the growing and expensive problems associated with the internet today, in which case it is spam. Spam is a term that usually refers to the unwanted, unsolicited messages that are sent to a user's inbox. By obtaining such messages, the recipients are encountered to the security threat that is exposed to illegal content. It has an attractive link to famous websites, but it leads to websites that are disturbing (Dada et al., 2019). Spam Email is also considered to be the carrier of malware to infect the victim’s machine. Spam mails are widely classified into:

- ***Spam of adult content:*** this is common for the young group of ages to fall into this flash trap for the content of products and services aimed at improving the sexual life of adults.
- ***Health and medical spam:*** this kind of spam email is huge just for beauty purpose; this is intended to promote the products for weight loss and skincare.
- ***Computer and internet spam:*** this spam is generally more dangerous than the previous one, because it relates to the field of work that offers hardware and software services with a familiar image to those who are active in a company with offers for very under an IT department that makes it easy for users and companies that are aware of the threat (Lee, Lee, & Hong, 2017).

- ***Financial spam:*** for extortion and financial purpose, this spam focuses on the banking sector, insurance policies and low-interest loans (Grimes, Hough, & Signorella, 2007).

The primary infection vectors for ransomware is through malicious spam emails, where the victim is tricked (Da-Yu et al., 2019). Opening a phishing email is an insufficient method to execute ransomware, but attackers still need users to download or open malicious attachments that directly install the ransomware; another way of the phishing email to deliver ransomware is to click on malicious links within phishing emails that appear to be a legitimate email message, the 93% of phishing attacks is ransomware purpose (Goel & Jain, 2018).

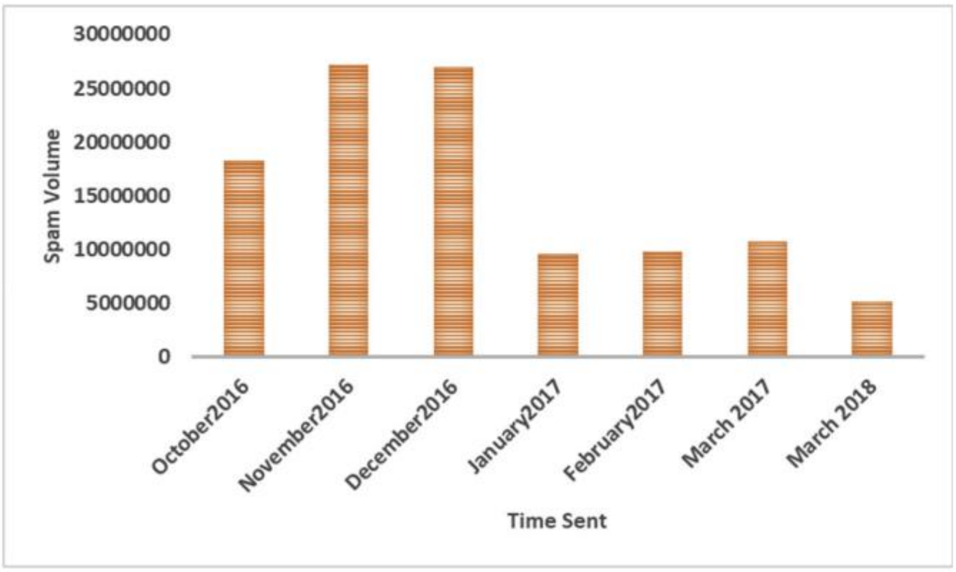


Figure 2.16: The distribution of spam emails in 2016 to 2018 (Dada et al., 2019).

2.6.2 Social engineering

Social engineering is the art of manipulating, persuading, suggestion, and deceiving people to gain access to a user's computer. It is an easier method that plays into human nature's inclination to trust or to carry out actions that grant the ransomware creators to access the victim's machine (Gallegos-Segovia et al., 2017). Social engineering attacks take place in one or more phases, as shown in Figure 2.17. First, the attacker investigates the targeted victim to collect the necessary basic information, such as potential access points and weak security protocols, that are needed to continue the attack. The attacker then moves to obtain the trust of the victim and provide an incentive for further actions that violate security practices, such as revealing sensitive information or providing access to critical resources. The most commonly known model generated by the Kevin Mitnick’s social engineering attack cycle described as (Mouton, Malan, Leenen, & Venter, 2014):

I. ***Collecting information**:* is a process that involves the collection of victim information to have a scoop of the attack and identification of the possible attack vectors.

II. ***Trust:*** strives for creating a relationship between the attacker and the victim. Usually, people tend to discover the information when they trust someone, which will obtain critical information.

III. ***Exploitation:*** as soon as a victim trusts an attacker, the link is operated through requests for information or develop specific actions. Moreover, the victim can be manipulated to seek the help of an attacker.

IV. ***Reach /ending:*** results that obtained from the earlier phases are used to achieve the goals of the attacker.

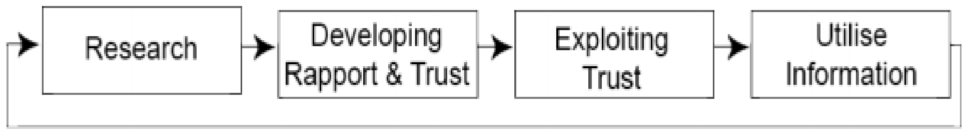


Figure 2.17: The Social Engineering Cycle Attack (Mouton et al., 2014)

For instance, the ransomware attacker needs to connect to the network of an organization. The attacker discovers that a help desk employee knows the password for the organization's wireless network. Moreover, the attacker obtained personal information about the employee who has been recognized to be his target. The attacker starts a conversation with the target using the information obtained to create a trust; in this case, the ransomware attacker presents himself as an old-fashioned knowledge of the target. The attacker then exploits the trust by requesting permission to use the corporate wireless network to send an e-mail. The help desk is willing to provide the password to the attacker, then the ransomware author has access to the organizational network and achieves its goal (Gallegos-Segovia et al., 2017).

2.6.3 Exploit kits

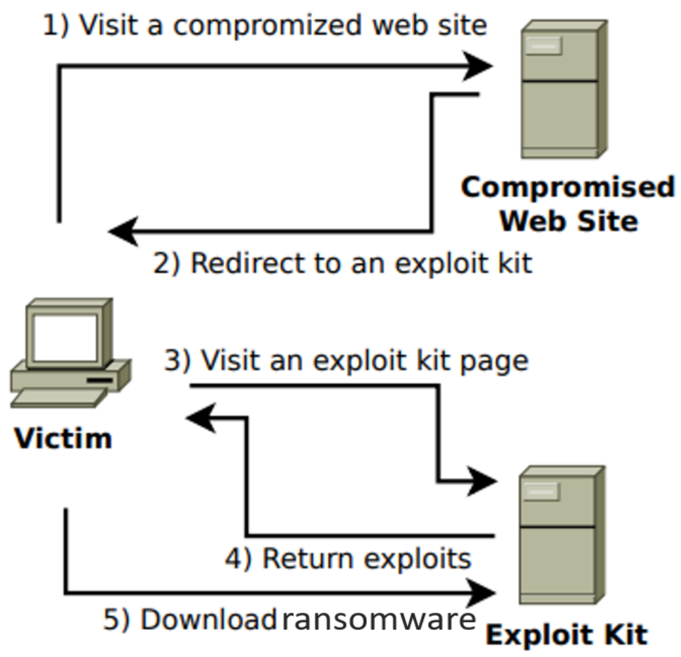
Another common method for spreading ransomware is a toolkit that automates the exploitation of software vulnerabilities for distributing malware(Corrigan, 2017). An exploitation kit is a malicious software tool that is created, sold and rented, and available on the software industries and used by ransomware attackers to carry out drive-by-download attacks. To evade detection, exploit kits are usually encrypted. Exploit kit is an HTTP server-side application that depends on the returning page with a suitable set of exploits and the request headers.

The main objective is to download and execute the malicious ransomware payloads on the victim machine by utilizing the vulnerabilities of the browser (Kotov & Massacci, 2013). Once a vulnerability is exploited, a traditional piece of ransomware is loaded onto the victims' computer. Most often, hackers inject malicious code on a website that redirects the victim to a malicious site (Yaqoob et al., 2017) as shown in Figure 2.18. The malicious webpage then returns an HTML document with exploits, which are typically hidden in unseen JavaScript code.

The exploit kit identifies vulnerabilities in browsers; if it is vulnerable, it can leverage it to download ransomware and the victim gets infected. Some ransomware variants such as wannaCry ransomware propagated through a dropper component named as EternalBlue that identifies vulnerabilities in the Server Message Block (SMB) protocol, which enables ransomware to drop binary onto all unpatched, vulnerable windows machine (Yaqoob et al., 2017).

The increase of the exploit kits becomes more due to the ease of deployment and the ease way that the ransomware causes the infections. Exploit kits can be deployed easily, with no advanced exploitation knowledge required, and victims can be directed to them through a malicious redirect or simply via a hyperlink.

For instance, the normal user is misled to visit the malicious link provided by the exploit kit authors that are redirected to the hosting site of the Blackhole. Then, many exploit modules are loaded silently in the background. If the attach is executed successfully, the ransomware payload is downloaded and runs silently in the background. This kit is known to address various vulnerabilities in Adobe Acrobat, Internet Explorer, Java, Adobe Flash, and Windows (Mansfield-Devine, 2013).



1) Visit a compromized web site
Compromised Web Site
2) Redirect to an exploit kit
Victim
3) Visit an exploit kit page
4) Return exploits
5) Download ransomware
Exploit Kit

Figure 2.18: Exploit kit drive-by-download method (Kotov & Massacci, 2013).

2.6.4 Malvertising: Exploiting web advertising

Online advertising is an online malicious advertising method for distributing of malware (Sood & Enbody, 2011) used by attackers to inject malicious advertisements into trusted websites with many visitors. Since advertisements generate an important portion of the income on the web, considerable efforts are made to attract users to the advertisement pages. Malicious agents utilized this attraction and then redirect normal users to infected sites that distribute ransomware. Often, when the user opens the website, there is no need to click on the ad; loading malvertising page will connect to several different URLs that lead to ransomware infection (Bhardwaj et al., 2016) as shown in Figure 2.19. Dynamic delivery of ads is considered to be another approach of online advertising that can be appointed by malicious agents. The content of this approach is dynamically changes based on the characteristics of the user's profile.

The malvertising can appear many popular websites because the attacker can purchase ad space to install a harmful piece of code. This malicious advertising appears like daily announcements as pop-ups such as fake browser updates, free tools, banner ads, antivirus programs, etc. As a consequence, the malicious content can reach a very large audience that trusted the website. In addition, the users may not identify that they come across malicious content website while browsing reputable websites that puts them high risk (Xing et al., 2015). Malware authors employ two main methods to infect the victim's computer:

1. The advertisement that displays a kind of disturbing appeal to make you click on the advertisement. The request appears in the form of a "warning", such as a warning that you are in the risk of malware infection. Or providing a free program. These tactics utilize social engineering to persuade the user to click on a link (Sood & Enbody, 2011).
2. Another popular method is drive-by-download employing an invisible web page module to infect the user. Loading malvertising page that hosting the ad leads you to an exploit landing page, which uses all your vulnerabilities in your browser or your software protection to access your machine, and redirect several different URLs that causes to ransomware infection.

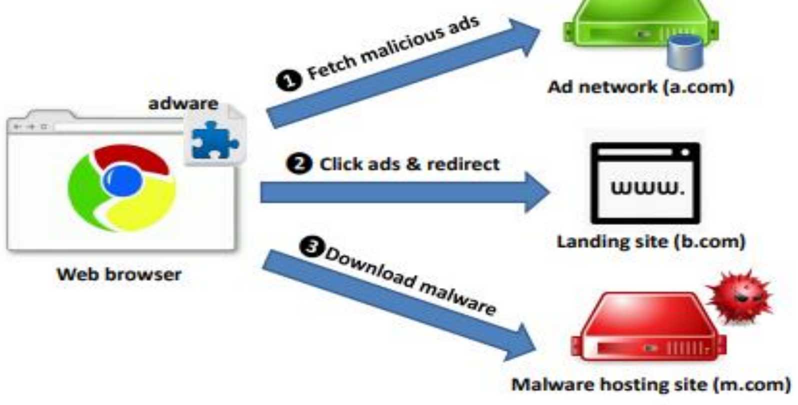


Figure 2.19: The architecture of Mal-advertisement flow (Xing et al., 2015).

2.6.5 Dropper

Another common technique for ransomware is to send a JScript file (*.js) by E-Mail victims encountering the web to execute a file. There are several ways to make ransomware undetectable. One of them is the dropper. It is an algorithm that will not link one program to another. In other terms there will be a dropper and a payload (the ransomware). The dropper is responsible for initiating the installation of the ransomware on the system as shown in Figure 2.20. It will separate the ransomware from the installer. Simply, the dropper acts like a ransomware installer. The dropper activates when the victim believes they are downloading or launching the file / program they received as an attachment. The dropper sneaks into the ransomware installer without the victim's knowledge. In other words, the dropper is a means of launching the attack, it is not in itself a danger, but it allows to cross the security layers of a computer (Raunak & Krishnan, 2017).

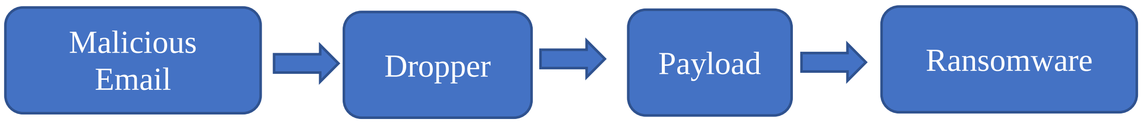


Figure 2.20: Ransomware infection vector using dropper installer

2.7 Ransomware Evasion Techniques

Malicious reverse engineering is the most common attack that aims to reverse or analyze the program in order to understand the inner working of the program and reconstruct it. Originally, this method is intended to protect the intellectual property of software developers through copyright prohibit the direct privacy of software, but it has been broadly used illegally by malware authors to evade the antivirus scanners (You & Yim, 2010). To protect the software, there are several methods such as server-side execution, native code, encryption and obfuscation, is the cheapest and simplest solution of copyright problem.

2.7.1 Code injection technique

To evade the execution, ransomware injects the legitimate programs through hooking technique. To achieve this, the malicious program has to be able to access the space memory of the victim application. To do this, the malware must have administrator rights or be able to acquire the necessary rights. Indeed, the manipulation of some of the APIs which allow injecting code is restricted. Code injection allows ransomware to execute in the context of a legitimate application, making it easier to evade detection (Francillon & Castelluccia, 2008). Injecting malicious code into a process legitimate can be achieved by different techniques, for example:

- Injection of a dynamic library; “Dll injection”,
- Adding code directly into the memory space of a process; “Direct injection”.

The first method is to force the loading of the malicious library via the execution of an attacker-controlled thread in the victim process as shown in Figure 2.21. For this, the CreateRemoteThread function is generally used. Once the malicious library is loaded, the system hands over to its entry point, the Dll main function, which can then trigger malicious behaviour.

The second method is used to inject compiled code (i.e., as before) or "shellcode", that is to say mainly the assembler as a string character. To do this, it is necessary to allocate memory in the victim process and then write to the desired address. The VirtualAllocEx and WriteProcessMemory functions are then used. Once the code is in place, the CreateRemoteThread function is called at the base address chosen to trigger the malicious behavior. This method is more complex to set up, and can affect the stability of legitimate code. The context in which the malicious code is executed is particular, the resolution of strings, but also of functions imported must be managed specifically.

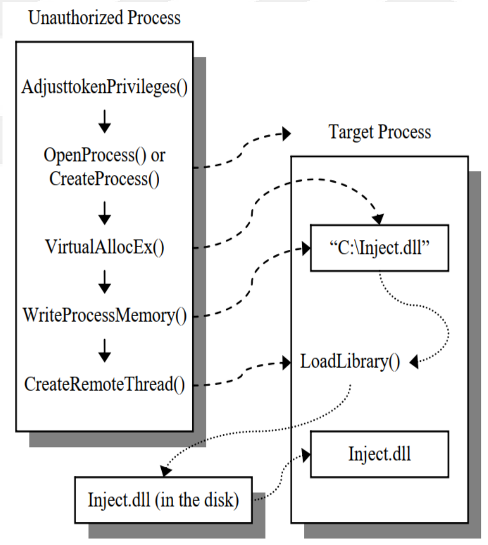


Figure 2.21: DLL Injection

Windows functionality can also be abused by malware to inject itself into a program. This is the "AppInit DLLs" technique. In fact, two registry keys specify the dynamic libraries that are loaded by user32.dll when this the latter is loaded by a process. In practice, a large number of programs use this library. Changing registry keys therefore allows ransomware under the form of a dynamic library to execute in the context of user programs that use user32.dll, and this in a persistent manner (Jiang, Wangz, Xu, & Wang, 2007).

2.7.2 Obfuscation technique

Another technique is "obfuscation". That is to say making the source code incomprehensible to humans or to a decompile. An Obfuscation is the process of application transformation that makes the program harder to understand through changing the physical appearance (Arini and Chloe, 2005). Code obfuscation techniques are exploited by polymorphic and metamorphic viruses to conceal their behavior from antivirus scanners. Obfuscation is common widely used by malware writers to avoid specific signature detection scanners through conversion of the program into a new different version. These include dead-code insertion, adding no-op instructions, unnecessary jumps and subroutine reordering (You & Yim, 2010). The most common obfuscation techniques divided into following main parties: Dead-code-insertion, Register Renaming and Code transportation.

- ***Dead-code insertion*** is the simplest technique that inserts useless codes and some ineffective instructions or a bunch of nops to a program in order to modify its physical appearance, but keep its behavior. NOP is equivalent to the (No Operation) instruction that is either not executed or has no effect on program outcomes but increases the size of the code. However, this technique is defeated by the signature-based antivirus scanners by deleting the ineffective instructions prior to analysis (Lynn, Prabhakaran, & Sahai, 2004).
- ***Instruction Substitution*** reordering and the substitution of instruction, during a set of instructions with corresponding instruction or set of instructions in a random fashion. This is achieved through creating of labels for each reorder and then employing conditional jump instructions to skip the control flow to the labels (Wing and Mark, 2006). Due to the change and reordering of the opcode sequence, this obfuscation technique is well-identified for bypassing the signature detection method, because it does not initiate too many jump instructions (Collberg, Thomborson, & Low, 2003).
- ***Register reassignment*** also known Register Renaming is another simple technique that replaces either the name of the variables or registers from used instruction with another unused instruction while keeping the behavior and program code. In contrast to other obfuscation techniques, subroutine permutation can be simply detected through signature detection, as the signature still exists in apparent view.

- ***Subroutine permutation*** is a simple obfuscation technique whereby the order of subroutines appears in the code is changed. In this type of code obfuscation will not affect the impact and functionality of the virus as the order in which a subroutine appears in the program is totally unrelated and does not affect the running and execution of the program.

In addition, transforming the code makes it possible to hide signatures or suspicious behavior. As a result, this allows you to get through certain levels of antivirus protection. But there are also other reasons to use "obfuscation". One of them is that the financial cost for such a movement is low. Especially since it is easy to set up for someone who has a minimum of experience. By the way, "obfuscation" is also used in the context of intellectual property security. This technique allows reverse engineering (a technique to reconstruct source code from its compiled form) (Collberg et al., 2003).

2.8 Ransomware Analysis

In recent years, ransomware writers transmitted their malicious executable files in encrypted form to avoid identifying by anti-virus software. To investigate the behaviour of the malicious files, ransomware analysis is necessary to get in depth understanding and learning behaviour of malicious code and how a specific piece of ransomware functions; it is a prerequisite subject and premise step of effective detection techniques. Malware analysis is useful for many purposes such as computer security incident management for organization’s response team to react to the situation whereby the potentially malicious files are discovered, and the Indicator of computer (IOC) extraction for the software solution companies. The purpose of ransomware analysis is to get knowledge about the capability of ransomware, the structure techniques, and anti-reverse techniques to hide it is self, and also the level of similarity to other malware samples (Zhang et al., 2019).

Generally, there are two main types of ransomware analysis techniques: static method and dynamic method (Gandotra, Bansal, & Sofat, 2014). There is no need to execute the ransomware samples during the static method, its analysis the characteristics of sample code, whereas dynamic analysis determines the behaviour of ransomware through execution. In this section, we will discuss static code analysis techniques and dynamic analysis, because both provide complementary information about the ransomware.

2.8.1 Static analysis technique

Static Malware Analysis (SMA) is the basic malware analysis method that analyses a program’s code without actually executing it. It provides information about data flow; programs control and other statistical features without executing the program (Al-rimy et al., 2018). It also offers the actual view of code and identification of malware files like compile dates, packers and functions used by the program. It includes many SMA methods: Basic Information Analysis, portable executable (PE) File Structure Analysis and Control Flow Analysis etc. (Liu, Ren, Liu, & Duan, 2011). Techniques are used to simulate the exploration of the program's control flow such as disassemblers, decompilers (i.e. IDA Pro) or even analyzing the source code. To prevent the malware analysis and detection, malwares adopt avoidance techniques and measures such as: polymorphism, shell code, and metamorphism, which make it harder to analysis malware.

This signature-based detection method is hampered by the avoidance techniques that ransomware employs such as obfuscation and/or packing. Such an approach is unreliable for detecting to the new ransomware, as it suffers several shortcomings such as frequently updated signature repository, and the need for expert intervention to analyses and extract attack signatures. Static analysis is therefore not retained because it is complex to implement in practice (e.g., computing power, time) (Zhang et al., 2019).

2.8.2 Dynamic analysis technique

In contrast to the static analysis, dynamic analysis requires the execution of the program and observing the actual action performed; it executes in a real or a virtual machine environment, the purpose of dynamic malware analysis is to provide insight the action performed by given malware, mostly without manual reverse engineering. The advantage of dynamic analysis is immune to the impact of obfuscation attempts and has no difficulty with self-modifying programs in execution stage (Hampton et al., 2018).

The environment in which the malicious file is executed is controlled to ensure not to affect the real system. These are complementary, the joint use of the two techniques is called hybrid analysis. In this thesis, we prefer to use dynamic analysis, for the simple reason that we are interested in the behavior of ransomware on the file system. It is much easier to observe their behaviors, i.e. their effects on the system, and thus to develop a countermeasure in this context. Many techniques can be used (e.g., encryption, obfuscation) and for the most part are current affairs issues difficult to deal with. There are two very important results:

- It is impossible to design a program that precisely detects all malicious programs.
- It is impossible to determine whether a program is a virus or not.

Because the problem is undecidable, so false positives and negatives are inevitable. In other words, there is no perfect detector and there is no algorithm that can tell exactly whether a program is malware or not. Behavior-based Malware Analysis is the most important method of dynamic analysis; it traces the behavior of a given malware in a quite accurate way. It also allows the analyst to know the exactly what actions the malware is performing like modification of registry, accessed virtual memory, created processes and created or Modified files (Liu et al., 2011).

A dynamic analysis take place in a realistic environment for the malware, so that it can run normally. Attackers aware that binaries are being actively analyzed, then they are developing defense mechanisms. In fact, they are trying to determine if they are running in an analysis environment, the activity and content of which are simulated, for example. Many artifacts are used to detect dummy environments (e.g., mouse movement). Analysis environments are called "sandboxes" or "sandboxes" and use different technologies. In addition, these are accompanied by monitoring tools, that is, introspection, to observe the effect of the code on the system. The sandbox as a whole must be as transparent as possible, that is to say indistinguishable from a conventional user environment (Kara & Aydos, 2018).

2.8.3 Virtualization

In order to conduct dynamic analysis and to reduce the impact and damage of ransomware execution on a real operating system, the virtual machine is important. A virtual machine (VM) is a software realization of a computing environment where by one or more different operating (OS) or program can be used to install and execute. Virtualization is the process of creating a software (or virtual) version of a physical entity". We then use the guest term to designate the virtualized system and the hosting term for the system that performs virtualization using the "virtual machine monitor"

(VMM) (Greamo & Ghosh, 2011). Generally, it is usually much less expensive to deploy a virtualized lab, than deploying one-to-one physical systems for all the same reasons large organizations have been doing so in their data centers. The most important reason researchers employ the use of VME's the ability to restore virtual machines to their original form in mere minutes is essential.

The virtual machine normally imitates a real computing environment, but needs for memory, hard disk, CPU, networked system and other hardware resources are controlled and managed by a virtualization layer that interprets these requests to the underlying physical hardware. We distinguish two types of virtualization:

1. Host virtualization, virtual machine and
2. Native virtualization, hypervisor.

In the first case, a host operating system is required. For the second, the hardware and the VMM manage the guest without a host operating system. The virtualized system must share the same hardware architecture as the host for (1) and (2), since most of the instructions are executed on the hardware. Emulation instead executes all instructions in software. Introspection is therefore easy. In addition, the guest does not have to share the same architecture as the host. Artifacts are inherent in virtualization and emulation, which makes solutions based on these approaches easily detectable. Many solutions exist for each of the approaches presented. This means that the victim machine seems likely to an attacker. For example, though, the files and directories present, the applications installed, etc.

The use of bare machines as a ransomware analysis platform has an advantage and such as: the total absence of virtualization or emulation. Therefore, using this bare machine as a platform is optimal transparency. Introspection, on the other hand, is more complicated to set up and has an impact on transparency (Hoopes, 2009).

Usually, guest operating systems and other programs are not known that they are executing or running on a virtual platform and, on condition that the VM's virtual platform is supported. In this case, the impact of ransomware execution has only affected the virtual PC and not the real one.

After completion of dynamic analysis, the infected virtual machine is discarded and installed by a clean one, because the installation of the new virtual machine is faster and easier than installing a real operating system on the computer. However, the general problem of VM is that Smart executable file may determine that it is running in a virtualized machine and modify its behaviour in order bypass to real PC (Al-rimy et al., 2018).

2.8.3.1 Virtual Box

Virtual Box is a free powerful x86 hardware virtualization product for Oracle. It is actively developed with frequent with extremely feature rich, high performance product for users such as snapshot which is the ability to save a machine state and later revert to it if anything goes wrong with the machine of the current state. It is also the only professional virtualization solution that is free and downloadable as Open Source Software under the terms of the GNU General Public License (GPL) (Gupta & Kumar, 2015).

Virtual box has important features which are the portability and compatibility means virtual Box can run on a large number of different 32-bit and 64-bit host operating systems versions such as Windows hosts machine host and Linux as well. The second feature is no hardware virtualization is needed. For example, virtual Box does not involve the processor features built into newer hardware like Intel VT-x or AMD-V. Unlike other virtualization solutions, Virtual Box even can be used on older hardware where these features are not at hand (Zavarsky & Lindskog, 2016).

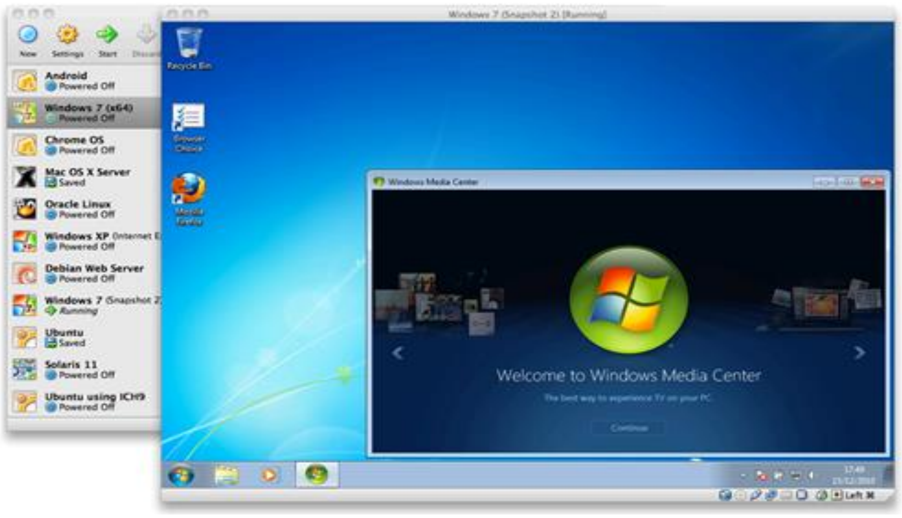


Figure2.22. Running multiple operating systems simultaneously.

The above screenshot describes how VirtualBox can be installed on a Mac computer and running multiple OS like Windows 7 in a virtual machine window. It is also providing users to execute more than one operating system in their computer simultaneously. This way, you can run software developed for one operating system on another written other operating system (for example, Windows software on Linux or a Mac) without necessary to reboot to use it (Chen & Bridges, 2017).

2.8.3.2 VMware Player

VMware Player is another free virtualization software that supports x86 virtualization. This product has the properties like Fidelity, Performance, Safety and Isolation, the most important types are VMware Player and VMware Server are both available for free download from the VMware site. It has a good out-of-the box support for seamless integration with the host operating system. The virtual machine will typically appear to run in a window on the local host, but it is not real machine (Player, 2010).

2.8.3.3 QEMU

Qemu -short for Quick Emulator- is an open-source emulation environment capable of running another operating system in a window. Qemu creates a virtual machine similar to the VMWare and VirtualBox. It is one of the isolated environments preferred by sandboxes (Yao & Wang, 2013). The memory resources of the native machine are divided between the host OS and the virtual machine means the guest OS. Some features of the Qemu environment itself are used for analysis system analysis by malicious software. Qemu emulation media do not have detection methods as much as virtualization environments. Among the features used for analysis system detection (Greamo & Ghosh, 2011).

2.8.4 Sandbox environments

A sandbox is a computer security term that designates a mechanism used to improve the security of software and web pages. The rapid increase in the number of malicious software also increased the need to prevent and examine malware. For this reason, the importance of the sandboxes, which speed up and facilitate the analysis of malware, has also increased and the number of sandboxes has increased. For an operating system, it reduces the risks when running software. This term refers to the environment that allows testing software or websites (Hoopes, 2009).

The suspicious file is run by sending it to an isolated environment. All operations performed in user mode such as file, registry and network activities are tracked with analysis tools and recorded. These recorded transactions are reported by the sandboxes when the analysis process or the determined time of the sandboxes is over. In cybersecurity, sandboxes are used for advanced malware detection: it is an additional layer of protection against new security threats, including zero-day viruses and stealth attacks. What happens in the sandbox stays in the sandbox, which prevents system crashes and prevents software vulnerabilities from spreading.

The environments created by Sandboxes provide a proactive layer of network security protection against new advanced and persistent threats (APT). APTs are targeted attacks, tailor-made and often aimed at compromising organizations and stealing data. They are designed to evade detection and often go under the radar of simpler detection methods (Greamo & Ghosh, 2011). Today there are many sandboxes. In this study, a sandbox that are accessible to everyone are used free of charge. The sandbox used are described below.

2.8.4.1 Cuckoo sandbox

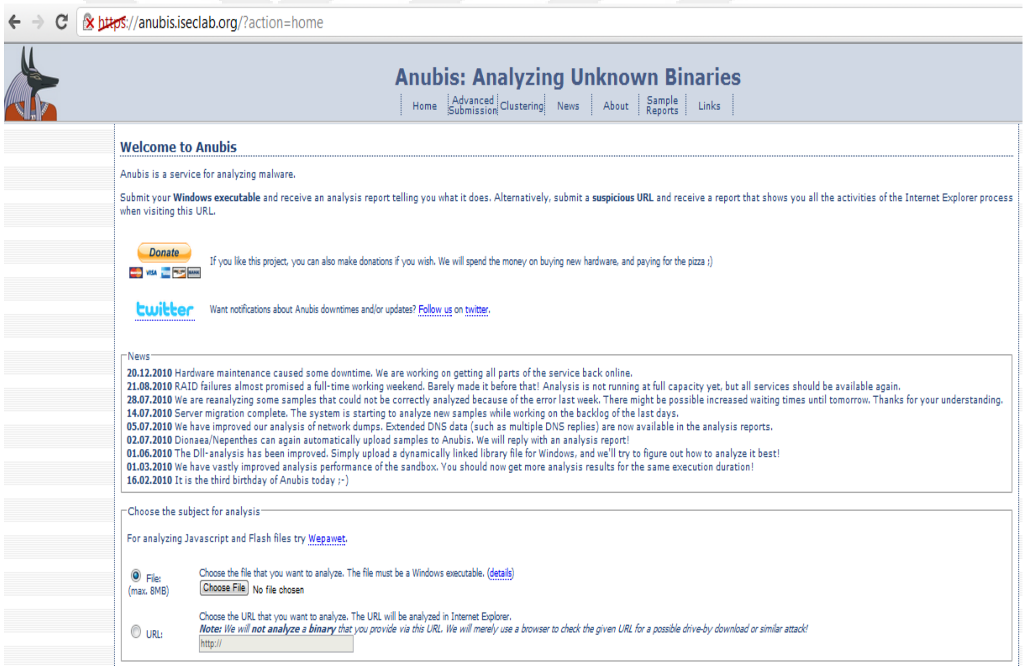
The analysis of malicious samples is carried out in Cuckoo Sandbox; the suspicious file is run on the isolated virtual or physical machine. The basic infrastructure of a cuckoo sandbox consists of one main machine and one or more isolated guest machines. The host machine sends the suspect file and analysis tools to the isolated environment. Then it runs the file and starts the analysis. While the analysis is taking place, the host also records the network traffic generated by the file.

The activities performed by the ransomware files in an isolated environment are tracked here, and when the analysis is completed, the results are sent to the main machine and reported. Cuckoo Sandbox performs analysis of Windows, Linux and Android operating system files and supports VirtualBox, VMware and KVM virtualization environments (Oktavianto & Muhardianto, 2013). Cuckoo sandbox monitors operations in user mode and generates analysis results accordingly. In this research, we applied the Cuckoo Sandbox for analyzing the ransomware sample to obtain the exact malicious behavior as we will discuss in section 3.

2.8.4.2 Anubis sandbox

Anubis sandbox is the automated malware analysis tool that uses isolated or-fully Qemu, and performs full system analysis. The operating system uses Windows XP SP2. Qemu is preferred because it has full access to the Qemu emulation environment and the basic virtual machine detection traces are not found in this environment. Within four minutes, it automatically performs and reports suspicious file analysis completely.

Anubis generates a detailed analysis report by tracking the system calls, Windows API functions, file and registry processes and network traffic in malware analysis. It can present the analysis report in HTML, XML, PDF and Text formats. Anubis sandbox is not affected by the complexation, debugging and packaging methods that may occur during the operation of the suspicious file by performing dynamic code analysis (Greamo & Ghosh, 2011).



Anubis: Analyzing Unknown Binaries

Welcome to Anubis

Anubis is a service for analyzing malware.

Submit your **Windows executable** and receive an analysis report telling you what it does. Alternatively, submit a **suspicious URL** and receive a report that shows you all the activities of the Internet Explorer process when visiting this URL.

If you like this project, you can also make donations if you wish. We will spend the money on buying new hardware, and paying for the pizza ;)

Want notifications about Anubis downtimes and/or updates? Follow us on twitter.

News

20.12.2010 Hardware maintenance caused some downtime. We are working on getting all parts of the service back online.
21.08.2010 RAID failures almost promised a full-time working weekend. Barely made it before that! Analysis is not running at full capacity yet, but all services should be available again.
28.07.2010 We are reanalyzing some samples that could not be correctly analyzed because of the error last week. There might be possible increased waiting times until tomorrow. Thanks for your understanding.
14.07.2010 Server migration complete. The system is starting to analyze new samples while working on the backlog of the last days.
05.07.2010 We have improved our analysis of network dumps. Extended DNS data (such as multiple DNS replies) are now available in the analysis reports.
02.07.2010 Dionaea/Nepenthes can again automatically upload samples to Anubis. We will reply with an analysis report!
01.06.2010 The Dll-analysis has been improved. Simply upload a dynamically linked library file for Windows, and we'll try to figure out how to analyze it best!
01.03.2010 We have vastly improved analysis performance of the sandbox. You should now get more analysis results for the same execution duration!
16.02.2010 It is the third birthday of Anubis today ;-)

Choose the subject for analysis

For analyzing Javascript and Flash files try Wepawet.

[✓ File:] (max. 8MB) Choose the file that you want to analyze. The file must be a Windows executable. (details) No file chosen

URL: Choose the URL that you want to analyze. The URL will be analyzed in Internet Explorer.
Note: *We will **not analyze a binary** that you provide via this URL. We will merely use a browser to check the given URL for a possible drive-by download or similar attack!*
http://

Figure 2.23: Anubis Sandbox Environment

2.8.4.3 Comodo automated analysis system

Comodo sandbox is a malware analysis environment that automatically analyzes the malicious files in fully isolated environments. Windows executable files are analyzed in a controlled environment. The outputs of the Comodo sandbox analysis is a report containing the actual activities performed by the malicious file and makes inferences about whether the file is harmful or not.

The Comodo sandbox also has a special feature such as the ability to show applications running in a virtualized environment from the main machine during analysis operations. Using VMware virtualization environment, Comodo keeps track of transactions in user mode (Kunwar & Sharma, 2016).

2.9 Ransomware Detection Methods

Ransomware is the most serious threat in today’s computing world. It continues to increase in huge volumes and to get out of control. It exposes various threats to the users such as stealing their sensitive information and breaking the computer system. A countermeasure of this threat is very important, and many researchers proposed a range of different malware-detection techniques. The ransomware detection techniques means any mechanism that provides the detection of any form of ransomware that threatens to the computer(Brewer, 2016). These techniques can be broadly classified into two categories: *approach-based detection* and *technique-based detection*.

Approach-signature-based detection uses its characterization of the knowledge to decide the maliciousness of a program, which means it extracts the byte code patterns of each malware and compares these patterns with byte code of a program under its repositories. However, this method cannot detect new (unknown) ransomware whose signature has not been found or generated yet.

On the other hand, approach-anomaly-based detection uses its knowledge of what considered normal behaviour to decide the maliciousness of a program under inspection. Hence this approach has benefits over signature based, because it detects any behaviour that violates the norm, and has the ability to identify new malware. Since the malware writers are familiarizing the detection mechanisms and evading detection methods through modifying their malware shapes like avoidance techniques that we discussed before(Saeed, Selamat, & Abuagoub, 2013).

Every technique has three subclasses; static approach determines the malicious of file based on structural information without execution of program. Although this approach has a drawback of not detecting obfuscated malware, but it is very fast approach. The dynamic approach has the ability to detect the malicious file during and after execution (Liu et al., 2011).

The third approach which is Hybrid is the combination of the best properties of both static and dynamic detection. The different class and sub-class of malware detection are showed in Figure 2.24.

We have generalized the ransomware detection techniques in Figure 2.24. in the remainder of this section, we focus on the ransomware-detection method on every approach ransomware detection and anomaly-based. Furthermore, we will do deeply to machine learning classification algorithms.

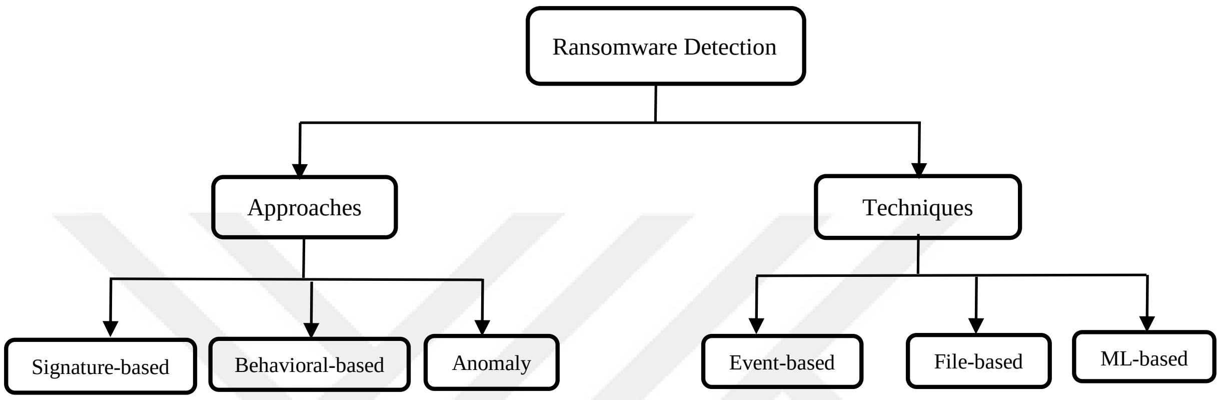


Figure 2.24: Ransomware detection and their subclasses.

2.9.1 Signature-based detection

Signature Based Detection is also known as a static approach, is considered to be the oldest technique, but associated with most popular virus detection techniques used today. This approach mainly used by antivirus companies; it is based on a signature database (fingerprints) that characterize the activity of each malware. The detection involves looking for known elements or patterns of attacks listed in the malware signature database. Most commercial antivirus scanners employ signatures which are normally a sequence of bytes within the malware code. Each virus has a unique string of bytes which becomes the signature of the virus. The fact of this technique becomes efficient is that comparing these unique strings with repository one, if matches found that file will consider malicious (Gandotra et al., 2014).

This technique can be either static or behavioural, depending on the nature of the malware signatures on which it is based. The static signature-based detection approach relies on structural properties of the program (for example, byte sequences or hash functions), while a dynamic approach will rely on information from execution (for example the systems seen on the execution stack) of the program. Typically, a static approach attempts to detect malware before the monitored program runs. On the other hand, a dynamic approach tries to detect malicious behaviour during or after the execution of the program.

The combined two approaches are called hybrid techniques, in which case static and dynamic information is used to detect malware(Kunwar & Sharma, 2016). The goal is to take advantage of the supposedly superior fidelity of dynamic analysis in the training phase, while maintaining the efficiency advantage of static detection in the scoring phase.

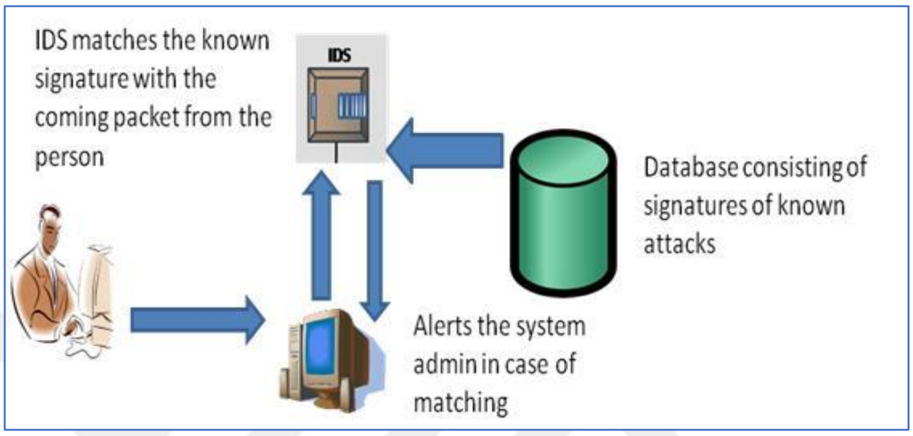

IDS matches the known signature with the coming packet from the person
IDS
Database consisting of signatures of known attacks
Alerts the system admin in case of matching

Figure 2.25: Signature-based Detection approach

The performance and accuracy of this technique is totally measured on the signature found in the database of the system. Signature-based detection systems cannot identify an unseen virus since the database will not have any detailed information about the unknown virus. Thus, the main condition of the system is to have an updated database of all the previous signature files of malware. However, previous researches have shown that signature-based detection is vulnerable to avoidance methods such as obfuscation techniques, for example, polymorphism and metamorphism(Saeed et al., 2013).

2.9.2 Behavioral-based detection

Behaviour-based detection techniques aim to reduce the rate of false positives generated during the monitoring phase of the system to be protected. During the learning phase, a behaviour-based detector is provided with a set of rules that specify all the acceptable behaviours of any application that may arise within the system to be protected. The major drawback of behaviour-based detection is the difficulty of determining the set of security behaviours that a program can exhibit during its execution within the system to be protected(Liu et al., 2011).

Rabek et al. (2003) present a method for detecting hidden malware that can be injected and generated dynamically at runtime. The detector uses a static analysis technique to obtain details of all relevant system calls embedded in the code, such as function names, addresses, and the instruction address followed by each system call. The detector maintains a record of return addresses for system calls in the code. Then, when a suspicious program is executed, the detector monitors the behavior of the executable and ensures that all calls to the system services during execution are the same as those recorded in the first step. The authors concluded based on a proof of concept study that their technique ensures that any injected and generated malicious code can be detected when it makes unexpected system calls. A major drawback of this technique is when inserting certain irrelevant API calls into malicious code, the detector may fail to match the new malicious behavior with the behavior already recorded (Rabek, Khazan, Lewandowski, & Cunningham, 2003).

Research by Wang and Karri (2013) presents the NumChecker tool, a new virtual machine monitor (VMM) founded to detect the flow of modification control of kernel rootkits in the host virtual machine (VM). NumChecker detects malicious changes to a system call in the VM guest by checking the number of certain hardware events that occur during the execution of the system call. To automatically count these events, NumChecker relies on hardware performance counters which are mainly the total number of instructions, branches, returns and floating-point operations. By using these parameters, the verification cost is considerably reduced and the protection is reinforced.

In their study, the rootkits used are SuckIT which replaces the system call table with its own copy and it uses it for redirecting to malicious system calls. Another rootkit used is Adore-ng which manipulates function pointers at the VFS layer for redirecting the execution flow of malicious routines which hide information by filtering system data. A deviation rate of 5% is set as the value above which a system call is considered to be modified (X. Wang & Karri, 2013).

2.9.3 Anomaly-based detection

The other method is anomaly-based detection systems that is designed to examine the processes going on a host machine for any variation for normal activity. If any abnormal activity is monitored, the system raises an alarm indication the possible occurrence of malware. This approach has two phases; a learning phase and a detection phase. The first phase consists of creating, automatically or manually, a profile for the monitored program (a normal model). It does not matter how the profile is created as long as the profile precisely defines the characteristics of the program being monitored. Then, in the detection phase, if a behavior deviates too much from the saved profile, the system generates an alarm indicating that the program activity is malicious (Gandotra et al., 2014). In this detection technique, the system attempts to learn normal behavior and employ the collected heuristics in order to classify an activity as normal or malicious.

The main advantage of anomaly-based detection over signature based is its ability to detect zero-day attacks, by defining the expected normal behavior, any anomaly can be detected, whether it’s part of a known attack or not, means it is more reliable and has the capable of detecting new viruses. However, the possibility of false alarm occurs relatively higher in this mechanism, because the normal behavior can change over time, although raising a false alarm is not as a potential problem as allowing a new virus. However, these systems need to have trained regularly by intruders to examine abnormal behavior as normal. Thus, the system will fail to identify the abnormal activity from normal behavior (Morato et al., 2018).

The most important of dynamic analysis is Behavior-based Malware Detection (BMD) method which utilizes the behavior information of the malware during its execution in a virtual machine such as one provided by VMware is a popular choice of the detection basis. This method can eliminate the reliance on the feature of the malware file itself. Because the behavior of malware is more unique than that of malware feature, for example, a malware can generate a new variation with different feature through shell code, Polymorphism and Metamorphism, but its behavior is still the same as the original one (Liu et al., 2011).

2.9.4 Event-based detection

Event-based detection approaches are intended for specific events that occurred and identified as compromised indicators for detecting the ransomware attacks. This technique finds the specific attributes and events necessary to identify variants of ransomware. Most event-based detection approaches do not need to create or maintain signatures to detect malicious files.

This approach usually detects an event when it is happening in the program under test or in the host system where the program will be executed. This method identifies unusual events, or unusual sequences of events, that occurred on the computing device. For an example, accessing the user's critical information on the hard disk, a server, event logs or the number of events that are less than a threshold number of events can be used to identify an unusual event or an unusual sequence of events. Ahmadian et al. (2015) proposed monitoring C&C communications to depict any encryption key, DGA needs additional data exchanged between the malicious code and its remote C&C server, to distinguish ransomware before it begins its essential functionality(Ahmadian et al., 2015). Likewise, Heldroid, suggested by Andronio et al. (2015), employs the automated dynamic method to determine the procedure of obtaining the threatening text from the C&C server in the case where such text is not entrenched in the payload of the crypto-ransomware (Nicoló Andronio et al., 2015). similarly, Le Guernic and Legay (2017) suggested a technique that monitors Microsoft's cryptographic APIs that several variants of malicious executable employs as indicators of ransomware attacks to avoid it from locking the victims' files (Palisse et al., 2016).

2.9.5 File-based detection

This technique is considered to be one of the most effective techniques for detecting ransomware. This approach detection is to monitor the resources subject to attack instead of the malicious process that carries out the attack. File-based detection is created according to the frequently examining the user-related files and documents to monitor any malicious changes. To do this, many techniques are used, such as entropy and similarity measurements. Based on the changes made by the cryptography causes to the targeted file, it is measured the entropy before and after the encryption.

This method mainly consists of verifying whether certain regions of the file and memory system are identical to a trust base containing known values of these regions. The regions that need to be analyzed are most likely to be changed such as sections of file changes (Tang, Sethumadhavan, & Stolfo, 2014) .

Many studies have been conducted, network and file integrity monitor called tripwire were presented by Ben22 when critical system files are modified, it gives an alarm to the administrator. These monitors are based on witness files and simple hash comparisons, the LanmanServer operation is denied if the witness files are altered.

However, this approach cannot guarantee the changed witness files are modified by a malicious program or a normal user.

An alternative method presented HelDroid, an automated approach that classifies known and unknown mobile ransomware and scareware using a machine-learning method. Their approach is based on detecting threatening text associated with a ransom note and the “building blocks” that are typically needed to implement a mobile ransomware application (Nicoló Andronio et al., 2015).

2.10 Machine Learning-based Detection

Recently, classification algorithms have been used to automate and extend the idea of behavioral methods. In these methods, the binary code of a file or the behavior of the running application is represented and classifiers are used to learn patterns in order to classify new (unknown) applications as malicious. In this section, we will deeply examine the ransomware detection using machine learning;therefore, it is use full to understand the concept of machine learning such as file representation, feature selection and classification algorithms.

Machine Learning is an important field of Artificial Intelligence, which aims to imitate the intelligent abilities of humans by machines through recognition. It has algorithms that allow computers to reason and make decisions based on input data (Shabtai, Moskovitch, Feher, Dolev, & Elovici, 2012).

Generally, Machine learning (ML) is typically classified into two separate areas, supervised ML and unsupervised ML. Supervised machine learning is a method that attempts to find out the relationship between input attributes (also known field, independent variables, feature) and a target attribute (sometimes referred to as a dependent variable). ML is also evolved to as classification in the statistics literature; unsupervised learning is referred to as clustering where instances are unlabelled. Both types of machine learning are concerned with the analysis of datasets containing multivariate observations(Zhang et al., 2019).

This research will focus on supervised and semi- supervised machine learning. In supervised approach is where algorithm (classier) attempts to map inputs to desired outputs using a specific function. In classification problems a classifier needs to learn several features (variables or inputs) to predict an output (response).

Supervised methods can be implemented in a variety of domains and applications such as marketing, finance and manufacturing, disease diagnosis and face recognition, but in this research, we will employ machine learning to identify unseen files either benign or malicious. Before we go deeply into the supervised machine learning algorithms, we will briefly discuss the types of machine learning.

2.10.1 Supervised learning

The formulation of the problem of supervised learning is simple: we have a finite number of examples of a task to be performed, in the form of pairs (input, desired output). We want to obtain, automatically, a system able to find relatively reliably the output corresponding to any new entry which could be presented to him. Supervised learning creates knowledge structures which have the task of classifying new instances into predefined classes(Bishop, 2006).

2.10.1.1 Regression

In regression problems, the entry is not associated with a class, but in the general case, with one or more real values (a vector). For example, for a biochemistry experiment, one might want to predict the reaction rate of an organism based on the levels of different substances administered to it (Segal, 2004).

2.10.1.2 Time series

In time series problems, it is typically a question of predicting the future values of a certain amount knowing its past values as well as other information, for example, the performance of a stock market share. An important difference between regression or classification problems is that the data typically follow a non-stationary distribution (Bishop, 2006).

2.10.1.3 Classification

In classification problems, the entry corresponds to an instance of a class, and the associated output indicates the class. For example, for a face recognition problem, the entry would be a bitmap image of a person as provided by a camera, and the output would indicate which person it is among the set of people we want the system to recognize (Dada et al., 2019). Detecting malware using this approach is accomplished in two phases:

- ***The training phase:*** A detection system must be formed with input data in order to capture the characteristics of interest;

- ***The detection phase:*** In this phase, the trained detector makes intelligent decisions on new samples based on training data.

In addition, there are two methods deployed in the training phase:

- The first method uses two categories of data, namely normal and abnormal data;
- The second method uses only one category of data. In this case, malware detectors are trained with only one class (normal or abnormal).

This means that the system will be formed only with normal system activity, which allows it to identify the presence of abnormal activity (Alzarooni, 2012). Various machine learning approaches such as association rules, support vector machine, decision trees, random forest, naïve-Bayes and clustering have been proposed for the detection and classification of malware. The classification is applied to unknown samples in the family of known malware or also underlines the samples that exhibit invisible behaviour for detailed analysis. There are generally three types of problems to which supervised learning is applied.

A. Training Phase

In the machine learning algorithms to identify unknown malicious files, it needs to have datasets composed of several characteristic features of both malicious and benign software. Combining with learning algorithms it will generate classifiers. The General process of classifying previous unseen files as either malicious or benign using ML methods is separated into two succeeding phases: training, and testing phases. The training phase is the learning phase which is supplied with a collection of sample or instances (called the training dataset) which is pre-classified into classes.

The result of the learning process is a classification model which is constructed by examining and generalizing the data provided. In fact, supervised learning focuses on modelling the input/output relationships. Its objective is to identify a correspondence from the characteristics of entry to an exiting class. The acquired knowledge can be presented in the form of a flowchart, a decision tree, classification rules which can be used later to classify a new invisible instance. Then, each file is parsed; represent the documents as vectors and each file is extracted based on the predetermined vocabulary (Anderson, Kharkar, Filar, & Roth, 2017). The representative vectors of the files in the training set and the real classification of each file (benign/ malicious) serve as input for a training algorithm. Figure 2.26. Show the process of the training phase.

To put it simply, we train an algorithm and at the end pick the model that best predicts some well-defined output based on the input data. Supervised techniques adapt the model to reproduce outputs known from a training set. In the beginning, the system receives input data as well as output data. Its task is to create appropriate rules that map the input to the output. The training process should continue until the level of performance is high enough. After training, the system should be able to assign output objects which it has not seen during the training phase. In most cases, this process is really fast and accurate (Dada et al., 2019).

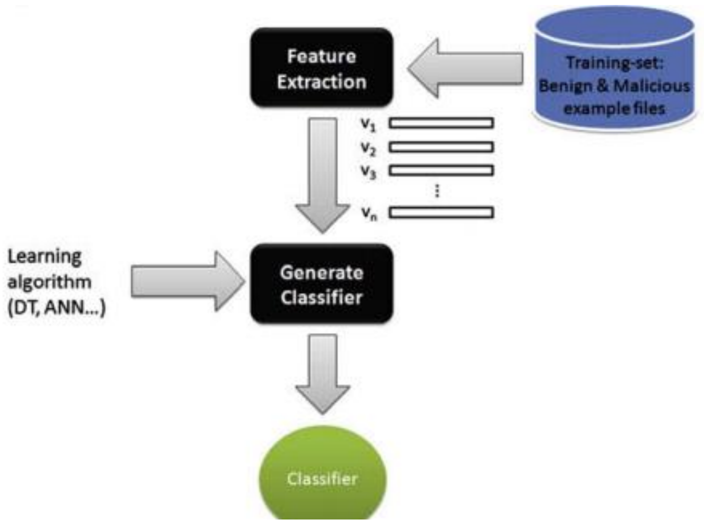

Feature Extraction
Training-set: Benign & Malicious example files
V1
V2
V3
Vn
Learning algorithm (DT, ANN...)
Generate Classifier
Classifier

Figure2.26: The process of training phase (Shabtai, Moskovitch, Elovici, & Glezer, 2009).

B. Testing Phase

The second phase is testing, during the testing phase a set of collected benign and malicious files is required that did not associate in the training set. Each file in the testing phase is prepared to pars and extracted from vector model. According to this vector, the classifier identifies a file as either malicious or benign. During this phase, the performance of the classifier can be evaluated by extracting performance criteria accuracy for measuring of the classifier's algorithms (Shabtai et al., 2009).

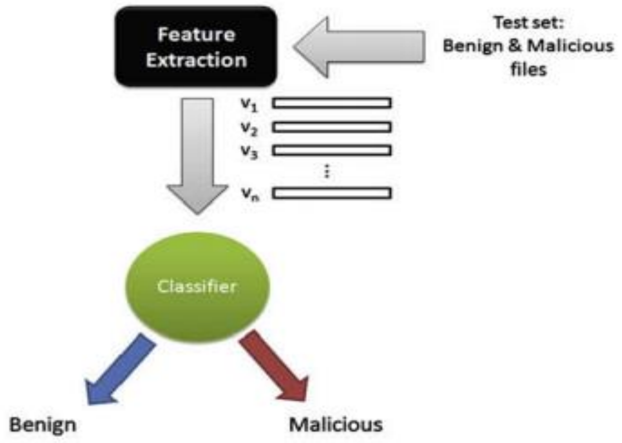


Figure 2.27: The process of testing phase(Shabtai et al., 2009).

The test phase in the classification is where the model that was built in the training phase is used to classify the new invisible instances. In the Testing phase we use data set that did not appear in the training phase because we want to evaluate the algorithms what they have learnt. Therefore, classifier will identify input attribute as benign or malicious.

2.10.1.4 Executable file representation

The first premise of implementing machine learning for malware detection is to determine the representation of executable files, because it holds a program suitable for execution. There are several representation files such as Byte n-gram, Portable Executable features, String features and OpCode n-grams. Byte n-gram features are sequences of n bytes extracted from an executable file. This is a familiar approach for training machine-learning classifiers to identify unseen malicious code (Igor et al, 2003). To achieve a representation of the executables through byte n-grams, we need to extract every possible sequence of bytes and their appearance frequency. Figure 2.28 shows the overall process of malware detection using machine learning.

One of the executable file formats used in the window is called the Portable Executable (PE) file, which is normally used by files with extensions like . EXE, .DLL, .SYS and. SCR. These features are extracted from certain parts of these extensions to indicate the modification of file such as the creation of file or infected to perform malicious activity. PE file consists of MS-DOS header that provides the compatibility with DOS environment. PE Header is the actual start of the PE file. This place important information that describes the logical structure of a PE binary is stored.

String features are considered as plain-text strings that are fixed in program executable files like windows and kernel. It used to represent files in the same way as the text categorization problem. An OpCode stands for operational code defines the sections of instructions that describe the operation to be done. A complete machine language instruction comprised of two parts an OpCode and, optionally, the pattern of one or more operands. The following section will cover feature selection methods in order to understand the machine learning process (Segal, 2004).

In order to generate various attributes representing the file Function-based feature is required to extracts functions that reside in the binary representation of a file. These functions include the size of the longest detected function, the total number of functions detected, the average size of detected functions, size of the shortest detected function, the standard deviation of the size of detected functions and etc (Shabtai et al., 2009).

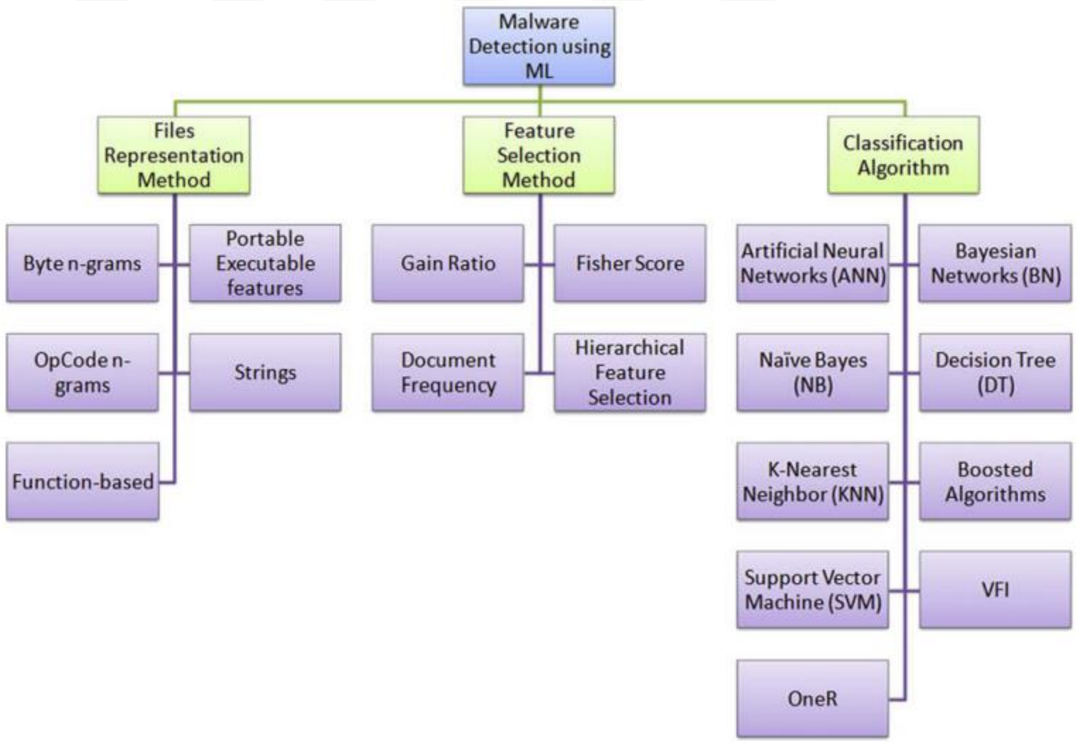


Figure 2.28: ML classifiers taxonomy for malware detection (Shabtai et al., 2009).

2.10.1.5 Feature selection methods

Feature selection is the process of detecting relevant features in the dataset and reducing everything else as redundant and irrelevant. Feature selection allows to reduce the dimensionality of the vector and to avoid vectors that may cause accuracy of classification algorithm negatively. It allows the classification algorithms to function more effectively and make the classification method easier to understand. Generally, feature selection methods can be divided into three families: filter-based, wrapper based and embedded methods. The filter approach, a measure is used to identify the correlation of each feature to the class of malicious or benign (Gu, Li, & Han, 2012).

Wrapper-based methods score the features through a learning algorithm that will finally be used. They are optimal because they look for to increase the accuracy of a classifier, tailoring their solutions to a unique inducer. Embedded methods are a combination of feature selection with the learning algorithm. In this section, we focus on filter-based methods for supervised feature selection (Vinod & Viswalakshmi, 2018).

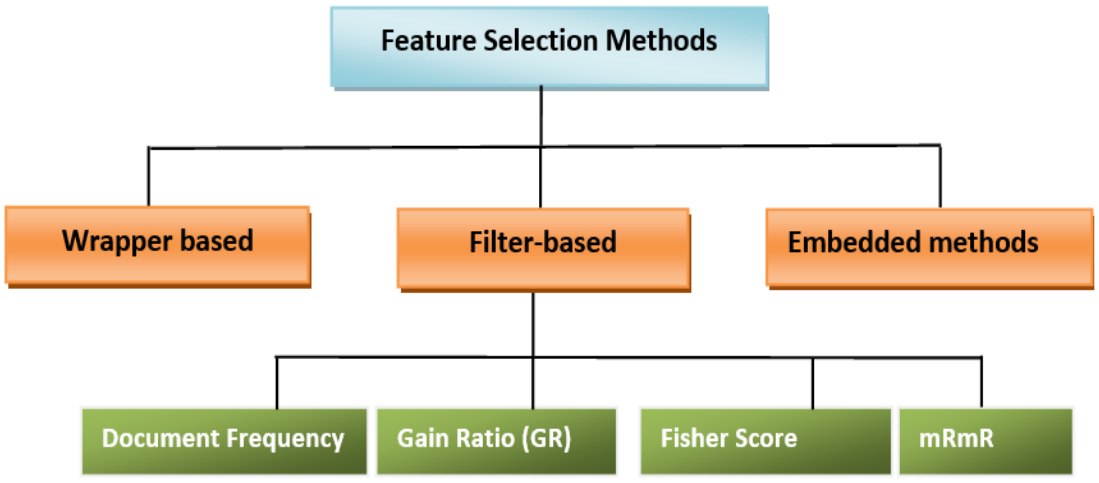


Figure 2.29: Types and subtype of feature selection methods

There are numerous filter-based methods for dimensionality reduction, including Document Frequency, Gain Ratio and Fisher Score. These methods function according to the filter approach. The feature evaluation that is employed by the feature selection method is free of any classification algorithm, thus allows comparing the performances of the different classification algorithms (Vinod & Viswalakshmi, 2018).

A. Filters

Filters select the subsets of the features by finding the correlation to the target class without involving any learning algorithm. Filters are less computational than wrapper approaches. To filter the features that contain little information, this type of method uses statistical measures for calculations based on these characteristics. This method is considered more as a pre-processing step (filtering) before the learning phase. The application of classification algorithms can only be made after this operation.

The main advantages of this type of method are computational efficiency and robustness in the face of over-learning. This type of method also has drawbacks such as ignoring the interactions between the characteristics and making the selection of the characteristics comprising redundant data instead of searching for those which have additional information. This type of method does not take into account the classification method that will be used (Das, 2001).

The features are generally evaluated by measuring the calculation for each of the feature. Let $X = \{x_k | x_k = (x_{k1}, x_{k1}, \ldots, x_{kn}), k = 1,2, \ldots \ldots, m\}$ a set of examples of learning that represents space with feature. Let $Y = \{y_k, k = 1,2, \ldots \ldots, m\}$ where y_k represents the class label of the sample y_x. Si x^i = (x_{1i}, x_{2i},….. x2i, ..., x_{mi}) represents the same feature (i = 1,2, ..., n) so, the goal of the filter evaluation method is to calculate the score to assess the degree of relevance of each of the features (x^i).

- ***Correlation Criterion:*** this score is used in the case of a binary classification $y_k \in \{-1,1\}$. It is estimated as follows

$$C(i) = \frac{\sum_{k=1}^{m}(x_{ki} - \mu_i)(y_k - \mu_y)}{\sqrt{\sum_{k=1}^{m}(x_{ki} - \mu_i)^2 \sum_{k=1}^{m}(y_k - \mu_y)^2}} \tag{2.1}$$

  Where μ_i and μ represent the mean values of the features and labels of the training set respectively. This function calculates the cosine of the angle between each of the characteristics and the label vector. In other words, for a given characteristic, a large absolute value of this measurement indicates its strong linear correlation with the vector of the labels (Y) (Amamra et al., 2015).

- ***Fisher's criterion:*** measures the degree of class separability using a given characteristic (Gu et al., 2012). It is defined by:

$$F(i) = \frac{\sum_{c=1}^{C} n_c(\mu_c^i - \mu^i)^2}{\sum_{c=1}^{C} n_c(\sigma_c^i)^2} \tag{2.2}$$

  Where n_c and ${\mu_c}^i$ represents the number, the mean and the standard deviation of the characteristic within class C respectively. The most global mean of the features, we could say that the measure is related to the interclass variance of the characteristic(Gu et al., 2012).

- ***Document Frequency (DF):*** is the number of files in which a specific n-gram appears. Gain Ratio (GR), initially showed by Quinlan in the context of Decision Trees, and was designed to get a solution of a bias in the Information-Gain (IG) measure which enumerate the expected declining of entropy caused by separation of the examples that are based according to a selected feature. The most widely used supervised feature selection methods is fisher score which selects each feature separately according to their scores under the Fisher criterion (Yang & Pedersen, 1997).
- ***Mutual information*** is a measure of dependence between the distributions of two populations (Yang & Pedersen, 1997). Let X and Y be two random variables whose instances are respectively the values of the characters and the class labels. Mutual information I (i) is defined as the Kullback-Leibler (KL) divergence between the probability $P(x^i, y)$ and the product of the probabilities $\left(P(x^i)\ P(y)\right)$. Mutual information is estimated empirically by:

$$I(i) = \sum_{x_i}\sum_{y} P(X = x_i, Y = y) log \frac{P(X = x_i, Y = y)}{P(X = x_i)P(Y = y)} \quad (2.3)$$

  where the probabilities $\left(P(x^i)\ P(y)\right)$P (xi), P (y) and P (xi, y) are estimated by the frequencies of the different possible values.

- ***Max-relevance, Min-Redundancy (mRmR)*** is a filtering method for the selection of characteristics proposed by Peng. This method is based on classical statistical measures such as mutual information, correlation. The basic idea is to take advantage of these measures to try to minimize the redundancy (mR) between the features and to maximize the relevance (MR). The authors propose two variants of their method. One for discrete data and the other for continuous data. For discrete data, the authors use mutual information to calculate the two factors mRmR (Ramírez-Gallego et al., 2017).
- ***SNR (Signal-to-Noise Ratio coefficient)*** is a score that measures the power of discrimination of a characteristic between two classes. Similar to the Fisher criterion, this method classifies features by calculating the ratio of the absolute value of the difference between the class means and the class standard deviation.

SNR formula for a characteristic and for a two-class problem is calculated by (Yang & Pedersen, 1997):

$$SNR(i) = \frac{2 \times |\mu_{C_{i1}} - \mu_{C_{i2}}|}{(\sigma_{C_{i1}} + \sigma_{C_{i2}})} \tag{2.4}$$

B. Wrappers

The Wrappers method select features based on predetermined learning algorithms, but this method tends to be computationally expensive and has overfitting problems (Shabtai et al., 2009). To seek the subset of the optimal characteristics, this approach proceeds to the exploration of the space of characteristics using a classification algorithm. The selected subsets are adapted to this algorithm and may not remain valid if another classification algorithm is used. The computation time depends mainly on the complexity of the employed learning algorithm. The enveloping methods are considered better than those of filtering due to several criteria. One of these criteria is that this type of method is capable of selecting the subsets that have a high-performance characteristic for the used classifier and in addition to smaller sizes than those of the filtering methods (Das, 2001).

The main drawback of "filter" approaches is the fact that they ignore the influence of the selected characteristics on the performance of the classifier to be used later (Das, 2001). To solve this problem, Kohavi and John introduced the concept "wrapper" for the selection of characteristics.

The wrapper methods, also known as wrapping methods, evaluate a subset of features by its classification performance using a learning algorithm. The subsets of features selected by this method are well suited to the classification algorithm used, but they are not necessarily valid if the classifier is changed. The complexity of the learning algorithm makes the wrapper methods very costly in computation time. In general, to decrease the computation time and to avoid over-learning problems, the cross-validation mechanism is frequently used. The problem of the complexity of this technique makes it impossible to use an exhaustive search strategy (NP-complete problem). Therefore, heuristic or random search methods can be used. Research is nonetheless becoming more and more impracticable with the increase in the size of the initial set of characteristics (Das, 2001).

C. Embedded

Unlike the “wrapper” and “filter” methods, the “embedded” methods (also called integrated methods) incorporate the selection of variables during the apprenticeship process. Such an integrated mechanism for the selection of characteristics can be found. In the selection methods of the wrapper type, the learning base is divided into two parts: a learning base and a validation base to validate the selected subset of characteristics.

On the other hand, integrated methods can use all the learning examples to establish the system. This is an advantage that can improve results. Another advantage of these methods is their greater speed compared to “Wrapper” approaches because they avoid that the classifier starts from zero for each subset of characteristics. Embedded feature selection methods can achieve comparable selection results with the wrapper model and have the similar efficiency with filter way (Das, 2001).

2.10.1.6 Classification algorithms

Various machine learning approaches such as association rules, Support Vector Machine, Decision Trees, Random Forest, Naive-Bayes and Clustering have been proposed for the detection and classification of malware. The classification is applied to unknown samples in the family of known malware or also underlines the samples that exhibit invisible behaviour for detailed analysis. An important task in Machine Learning is classification, where classifier distinguishes between different exemplars, based on their different patterns. It generalizes the relationship between the input attributes and the target attribute.

The overall process of machine learning classification is to collect malicious and benign files, and then represented by a vector of features, then, extraction of files from the PE-header and the binary code are needed. In the training phase, these files are employed to train a classifier. During the detection phase, based on the classifications of the classifier, an unseen file can be identified as malicious or benign (Segal, 2004). In this research, the most used learning algorithms in the literature are discussed in this section:

A. Decision Tree

Decision tree is a well-established classification algorithm that is built by recursively dividing the dataset into parts as a tree. Other words, they are Conventional graphical tree data structure contains nodes and leaves that solve if-then problems. Classifiers are represented as trees that have under nodes represents tests (attribute) of individual features and whose leaves are categorization decisions (classes). Typically, a unique characteristic of Decision Trees is the explicit creation of their knowledge, which can be easily represented as rules (Shabtai et al., 2009).

The overall process of creating a decision tree model starts with collecting a data set, called the training set. In this data set each record or element comprised of a set of attributes, one member of which represents the class for that element; all of the records have a similar structure. Specified data set, a model is derived for the class attributes as a function of the values of the other attributes. The main objectivity of the decision tree is to be able to categorize new records as accurately as possible (Tan et al., 2004). To evaluate the model's accuracy, a new data set is used which has the same structure as the training set, this is called the testing set (Dietterich & Kong, 1995).

B. Bayesian Network

The bayesian network, which is also called belief networks is a graphical representation of uncertain knowledge among a set of variables that easy to build and translate. In addition, the representation of these algorithms has formal probabilistic semantic, which generates it suitable for statistical implementation. It contains the acyclic graph of nodes and arcs representing causal dependencies among variables. The robust of the dependencies is described by conditional probability distributions. When creating the bayesian an arc is drawn from top to law or from parent node level to a child node and each node is connected with the conditional probabilities represented on its parent's variables(Shabtai et al., 2012).

Using the bayesian network is similar to that neural network; however, Bayesian network has two advantages over neural network. First, encoding process, the encoded expert knowledge can enhance the efficiency and accuracy of learning process. Second, is the corresponding approach which is the node and arcs in Bayesian network is correspond to causal relationship. In addition, Bayesian networks can handle incomplete data sets, and provide significant solution to the problem of over fitting (Heckerman, 2008).

C. Support Vector Machines

Support Vector Machine (SVMs) is a binary classifier that tries to discover a linear hyper-plane separating specified examples into the two given classes. Another terms, Support vector machines are collection of linked supervised learning methods used for classification and regression, and belong to linear classifiers family. It is a member of a class of learning algorithms known as Kernel Methods. In fact, SVM performed well on traditional text classification tasks; In addition, it is the best-known member of Kernel Methods. KMs are well suited for pattern analysis, the main purpose is to find and study different types of data relations for examples, rankings, clusters, correlations, and classifications (Dietterich & Kong, 1995).

SVMs are very well performed for data mining tasks such as classification, novelty detection, and regression. In real life scenarios, SVMs have been successfully implemented in areas such as:

1. Identification of particles
2. The process of recognition faces
3. The Problem of categorization text
4. Bioinformatics
5. Database marketing

SVM has been found to be successful when applied in many various applications, especially, for pattern classification problems and face recognition. In this thesis, we will focus on SVM for classification, because we are dealing with malware detection for identifying benign files and maliciousness programs. SVM is suited for data classification. Unlike, Neural Networks, SVM is power full and easy to use. However, occasionally insufficient outputs are received. A classification task typically associates with training and testing data comprised of some data examples. Every instance in the training set includes one target and several various attributes. The goal of SVM is to generate a model that estimates expected results from data instances in the testing set, which are given only the attributes (Zhang et al., 2019).

D. Artificial Neural Network

An Artificial Neural Network (ANN) (Carrasquilla & Melko, 2017) is an information processing paradigm encouraged by the way work of biological nervous systems, such as the brain, the information processing. The structure information processing system is the main important element in this mechanism. A neural network contains a collection of highly interrelated entities working together, called nodes or units. Each unit is intended to emulate its biological counterpart in the neuron. Each accepts a different weighted set of inputs through training algorithm such as back-propagation (Shabtai et al., 2009).

Neural Network has been implemented in the field of anti-malware mechanisms the overall process of classifying virus executable and normal file is based on the set of features that neural network employed. In the first process malicious files are analyzed in order to be appropriate input, then networks should be trained based on specific features. Then, the network model can be employed to identify viruses that contain of the most features present in the model. These network models are common for detecting viruses that have different family as the training model, but possess some of the malicious features from the training model (Segal, 2004).

The effectiveness of these network models also based upon the value of the threshold for the minimum amount number of selected features to be present in a test file. A higher threshold value is used to train the network model in order to classify viruses only from the particular virus family where by a lower value threshold generates a result of higher false positive rates. This detection technique was applied by the IBM Antivirus program to detect boot sector viruses. The program created the reliability of detecting the boot sector viruses effectively with less amount of low false positive rate (McIntosh et al., 2018). Table 2.2 provides some advantages and disadvantages of the above classifiers.

E. Random forests

A random forest is a mixture of the two techniques of “Bagging” and “Random Subspace” applied to decision trees (Breiman, 2001). At each iteration, a “bootstrap” sample is drawn randomly in order to build a binary decision tree. The search space for constructing tree nodes is limited by randomly drawn features. The performance of the method depends directly on the parameter P. A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models. A small value of risk of degrading the performance of the classifier. In (Breiman, 2001), the author has empirically shown that the optimal value of Pest: P is the total number of characteristics. The random technique of the approach has shown its relevance and its effectiveness, especially on high-dimensional data (with a high number of features). This technique allows for a better exploration of the representation space. Random forests are also used to solve classification problems in several areas, such as biomedical imagery (Breiman, 2001) . So, the advantages of the random forest algorithm as follows:

- Classification and the regression task can be employed on the same random forest algorithm
- The random forest classifier will handle the missing values.
- When we have more trees in the forest, random forest classifier will not overfit the model.
- Can model the random forest classifier for categorical values also

Using Random Forests based on classification data, the Gini index is most employed, the formula used to determine how nodes on a decision tree branch will be as follows:

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2 \tag{2.5}$$

This formula practices the class and probability to regulate the Gini of each branch on a node, deciding which of the branches is more likely to happen. Here, *pi* shows the comparative frequency of the class and observes in the dataset and *c* that represents the number of classes (Breiman, 2001).

As the above Table 2.2 presented, there are some obvious strengths of working with different classifiers; the most important advantage is the amount of data to train the classifier, the accuracy of the classifier and processing time. SVM can facilitate these relationships between input data and the target. In addition to, the error can be controlled explicitly.

In addition, ANN has the ability to continue operation if the one of the elements fails to work. However, there are also some clear limitations to these classifiers; for example, in order to perform training data ANN needs high processing time and large amount of data set. Moreover, other classifiers may be well suited for only certain parts or field. Therefore, combining these ML algorithms where they are most effective will increase the accuracy of detection rate.

2.10.1.7 Ensemble methods in machine learning

Ensemble methods are learning algorithms that construct a set of classifiers and then classify new data points by taking a weighted vote of their predictions. The main idea behind an ensemble method is to construct a set of classifiers in order to obtain better complex global classifiers. Ensemble methods are a member of learning algorithms that combine a bulk of classifiers and then classify new data. The ensemble of classifiers is more accurate than any of its individual members is if the classifiers are accurate and diverse. An accurate classifier is one that has an error rate of better than random guessing. For example, Neural Network is a suitable method for ensemble; an ensemble neural network is a learning pattern where a set of a finite number of neural networks is employed to train for the same task (Dietterich, 2000).

Table 2.2. Summary of Strengths and Weakness of the above Classifiers

Classifier	Strengths	Weakness
Decision Tree	They are inexpensive to construct and very fast at identifying or classifying unknown records. And also, the Accuracy can be compared with other data mining techniques (Tan *et al.*, 2004).	The ID3 and C4.5 algorithms are not reliable to find the simplest trees, because they only use heuristics and they operate as a black box mechanism, so optimization of tree for complex data will be difficult to understand visually (Kohavi and Quinlan, 1999)
Bayesian Network	Bayesian Network can readily handle incomplete data sets. It facilitates learning about causal relationships, and Avoid over fitting (David, 1995).	Spatial and temporal dynamics. It Continuous data representation therefore there is no feedback loops (David, 1995).
Support Vector Machines	SVM is power full and easy to train, it is no need for local optimal. Scales are well performed to high dimensional data and the trade-off between classifier complexity and error can be control explicitly **(**Vikramaditya, 2007).	In order to perform mapping of the attributes of the input space to the feature space SVM need for a good kernel function. Because it enables the operations to be done in the input space rather than the potentially high dimensional feature space **(**Vikramaditya, 2007).
Artificial Neural network	In case of failure occur, the neural network element continues without any problem by their parallel nature. It only needs learning, but not reprogrammed later, therefore and can be implemented in any application. (Ben and Patrick, 1996)	Neural network needs high processing time and training to operate. It has different architecture from the architecture of microprocessors therefore needs to be emulated. (Ben and Patrick, 1996)
Random Forest	Random Forest can be used to solve both classification as well as	Random Forest creates a lot of trees and combines their outputs. To do so,

	regression problems. Random Forest works well with both categorical and continuous variables. Random Forest can also automatically handle missing values.	this algorithm requires much more computational power and resources. Random Forest require much more time to train as compared to decision trees as it generates a lot of trees and makes decision on the majority of votes.

So as to make the ensemble more efficiency, there should be some kind of diversity between the classifiers. We can say two classifiers are diverse if they produce different errors on new data points. To see the importance of accuracy and diversity are good, let us take example, we have an ensemble consists of three classifiers: {h1, h2, h3} and assume a new case x. If the three classifiers are similar (i.e., not diverse), then when h1(x) is going to be wrong, h2 (x) and h3(x) will also be wrong. However, if the errors made by the classifiers are uncorrelated, then when h1(x) is wrong, h2(x) and h3(x) may be correct, that is the advantage of ensemble method (Dietterich, 2000) .

2.10.2 Unsupervised learning

In unsupervised learning, there is no notion of desired output;there is only a finite number of learning data, consisting of "inputs", without any label being attached to it.

2.10.2.1 Density estimation

In a density estimation problem, we seek to properly model the distribution of the data. The obtained estimator $f(x)$must be able to give a good estimate of the probability density to a test point from the same (unknown) distribution as the learning data (Tang et al., 2014).

2.10.2.2 Partitioning

The partitioning problem is the unsupervised counterpart of the classification. A partitioning algorithm attempts to partition the input space into a number of “classes” based on a finite learning set, containing no explicit class information. The criteria used to decide whether two points should belong to the same class or to different classes are specific to each algorithm but are very often linked to a distance measurement between points (Tang et al., 2014).

2.10.2.3 Dimensionality reduction

The goal of a dimension reduction algorithm is to succeed in "summarizing" the information present in the coordinates of a point in large high dimension $(X \in R^n, n\ grand)$ by a smaller number of characteristics $y = f(x), y \in R^m, m < n$. The hoped-for goal is to preserve “important” information, to highlight it by dissociating it from noise, and possibly to reveal an underlying structure that would not be immediately apparent in the original high-dimensional data. The most classic example of a dimension reduction algorithm is Principal Component Analysis (PCA) (Tang et al., 2014).

2.10.2.4 Deep learning

Machine learning (ML) systems are able to learn the desired behaviours of samples in the databases. In addition, such systems can be recycled regularly as more and more new data appear. Very sophisticated software systems, boosted by machine learning, are able to change their behaviour so radical without making big changes to their code. Deep Learning (DL) has become a revolutionized industry technology. Modern machine translation, search engines and assistant’s translations are all powered by deep learning. This trend of deep learning spreads its ability to build robotics, to products pharmaceutical, energy and all other areas of modern technology. Deep learning models try to imitate as much as possible the information and communication processing observed in the nervous system biological, such as neural coding, which attempts to define and describe existing relationships between multiple stimuli and associated neural responses in the brain (Dietterich, 2000).

Definition ***2.3:*** deep learning (DL) is a class of machine learning techniques (ML), in which the information is processed in hierarchical layers to understand the representations and features of data at different times (LeCun, Bengio, & Hinton, 2015).

For the increasing levels of complexity, deep learning is also called hierarchical learning or structured deep learning. Learning data representations could be done through semi-supervised, supervised or unsupervised approaches. In practice, all deep learning algorithms are Neural Networks, which share some common basic properties. They are all made up of interconnected neurons organized in layers. What sets them apart is the network architecture (or how neurons are organized in the network) and sometimes the way they are formed (Alhawi, Baldwin, & Dehghantanha, 2018). Deep learning can be classified into three main classes according to the objectives which they were designed:

- ***Deep networks of unsupervised or generative learning*** aim to capture a high-order correlation of input data for recognition purposes or synthesis of models. When this class is used to characterize for distributions of common statistics that are observed the data and their associated classes, networks have a generative mode and could be transformed into discriminating networks for deeper learning (LeCun et al., 2015).
- ***Deep supervised learning networks*** are used when data the target labels are available, models can directly provide a power of discriminating for classification purposes.
- ***Hybrid deep networks*** are the combination of the two types of networks that are mentioned above, so unsupervised deep networks could provide an excellent initialization on the basis of which discrimination could be examined (Shabtai et al., 2009).

2.11 Gab Analysis and Directions

Generally, in this section, we refer to the activity and research related to ransomware detection using supervised machine learning. The idea of employing symmetric key cryptography in cyber extortion started in 1989 when the AIDS Trojan has begun to infect machine through floppies. The use of public key cryptography for extortion was first introduced in (A. Young & Yung, 1996). They have presented how cryptography can be implemented in ransomware through Trojan. The authors also proposed countermeasures to monitor the access of the cryptographic tools. Nevertheless, this preventive approach is unable to detect the advanced ransomware variants.

A work of (Kharraz, Robertson, Balzarotti, Bilge, & Kirda, 2015) proposed a method to monitor the Master File Table (MFT) for activity and sorts of I/O Request Packets (IRP) of the file system to detect zero-day ransomware attacks. They suggested the mitigation strategy of employing the decoy technique to detect the maliciousness of the file. However, it is not clear whether the normal user would access the decoy resource before the attack occurs. Later work, they enhanced and introduced a new method called Unveil that is designed to detect the attack when ransomware tampers the user's data, typically by creating a fake user environment (Kirda, 2017). This approach is able to protect the user's files. However, the victim should sacrifice some data before the UNVEIL identifies the attack.

Some researchers employed a static analysis approach for the detection of ransomware. A recent work of Zhang investigated the opcode sequences feature for detection and classification of ransomware static analysis approach to map ransomware into families. The author extracted opcode from ransomware samples created N-grams sequences and calculated for each N-grams using term frequency-inverse document frequency (TF-IDF) to select the informative features between families. Then, applying five machine-learning algorithms achieved the best accuracy of 91.43% (Zhang et al., 2019). Similarly, Poudyal extracted assembly and dll level of ransomware binaries statically to perform multi-level analysis, then cosine similarity is used to measure the similarity between these binaries. Eight supervised machine learning classifiers are employed to classify ransomware and benign sample. The proposed framework achieved an accuracy of 97.95% when both extracted features are combined (Poudyal, Subedi, & Dasgupta, 2018) . However, the sophisticated packing techniques used by newly emerged ransomware can easily evade the static analysis. Furthermore, is not efficient for early ransomware detection since there is no need to execute the malicious samples during the static method, while some variants exhibit their malicious activities on the runtime.

To overcome the limitation of the static analysis, many studies have been conducted for the detection of ransomware based on the behavioural-centric approach. This method monitors and records the malicious activities done by the ransomware during the execution phase. For the purpose of detection and classification, most researchers utilized machine learning classifiers with the behavioural features extracted from the file with different filter and wrapper-based feature selection techniques.

Hampton explained the salient features of the ransomware using windows API call features on 14 various ransomware strains. The frequency of the system calls for the ransomware and baseline applications are compared to evaluate the similarity between them. The experimental evaluation of this work claims that the ransomware activities can be identified through a unique low-level system calls that are present in the ransomware (Hampton et al., 2018).

Ransomware detection scheme based on the sequence of API call history using SVM classifier has proposed by Takeuchi to monitor the execution of the windows API calls; samples are executed in a controlled environment. They deeply examined the sequences of API calls by creating a standardized vector representation of q-grams extracted from the output logs. The proposed SVM-based scheme showed an accuracy of 97.48% (Takeuchi et al., 2018).

Similar work has presented by Sgandurra to describe a set of the behaviour of ransomware using Windows API calls, registry key operations, and file system operations that captured in its early phases of ransomware at run-time. Authors proposed EldeRan, a framework to observe some unique actions performed by ransomware to dynamically analyses features that support ransomware detection. The authors selected the informative binary features using mutual information criteria and then applied a Logistic Regression classifier that achieved a 96.3% detection rate (Sgandurra et al., 2016) . They assigned a threshold of 30 seconds for the sample to execute. However, setting a fixed time does not apply to all ransomware samples, since some variants exhibit their malicious activities after human interaction or discovering the executing environment.

Qian and Bridges used an automated method for the extraction of ransomware features from the sandbox output logs, they analysed WannaCry ransomware and two polymorphic samples in isolated environment. To rank the most significant ransomware features, term frequency-inverse document frequency (TF-IDF) is used to weight the 74 features from the generated behavioural logs and the top 43 informative features are selected to discriminate the malware from benign samples. They claim that the method can accurately extract features from the logs, and the TF-IDF approach provides deeply analysis of WannaCry malware than other extraction algorithms. Nevertheless, the number of used WannaCry samples cannot describe the characteristics of ransomware (Chen & Bridges, 2017).

Similarly, an interesting behavioural early detection framework is proposed in Bander to detect zero-day crypto-ransomware using machine learning techniques with data-centric and semantic features. The detection module of the framework contains behavioral and anomaly detection scheme to improve the accuracy of the detection in the early stage before the encryption is carried out. However, the proposed framework was not implemented empirically(Al-rimy et al., 2017). Microsoft's Cryptographic API (MS CAPI) calls were presented by the Young explaining the method to encrypt the sensitive user's data and to produce the key by using MS CAPI with eight types of API calls (A. L. Young, 2005).

Another work presented by palisse et al. [2], which is a detection mechanism that enables users to decrypt their files by getting the advantage of the weak chaining mode that are used by some ransomware with cipher algorithm. The authors also propose another detection method based on the intercept calls used by Microsoft's Cryptographic API (Palisse et al., 2016). However, the proposed countermeasures are insufficient to detect other types of ransomware that use Cipher Block Chaining (CBC) mode. In addition, the protection is implemented, after the files are encrypted. The detection of high survivable ransomware was first proposed by Ahmadian, the authors implemented 2entFOX framework that extracts 20 static and dynamic features and their statistical possibilities. For classification case, they applied the Bayesian belief network to detect high survivable ransomware (Ahmadian & Shahriari, 2016) . However, the detection rate of this method was low due to the high dimensional feature space used.

An alternative method presented HelDroid, an automated approach that classifies known and unknown mobile ransomware and scareware using a machine-learning method. Their approach is based on detecting threatening text associated with a ransom note and the "building blocks" that are typically needed to implement a mobile ransomware application (Nicoló Andronio et al., 2015). Another work presented by Alhawi introduced the NetConverse, a supervised machine learning approach to detect ransomware using conversation-based network traffic features. They analysed 9 ransomware families, extracted 13 features using TShark and feed 6 classifiers such as Bayes network (BN), Decision Tree (J48), K-Nearest Neighbors (IBK), Multi-Layer Perceptron, Random Forest and Logistic Model Tree. The highest accuracy performed the Decision Tree (J48) classifier that showed 97.1% of detection rate with less positive (Alhawi et al., 2018).

Vinod proposed a supervised approach for the detection of obfuscated malicious samples with the extraction of the mnemonic n-gram features using the Minimum Redundancy Maximum Relevance (mRmR) filter and Principal Component Analysis (PCA). For a classification, several supervised machine learning algorithms such as NB, SMO, IBK, J48, Adaboost ADA and RF were used and obtained 94.1% detection accuracy with mRmR generated features (Vinod, Laxmi, & Gaur, 2012).

A multi-stage feature reduction method for analyzing the commonly-adapted network traffic features have presented by the Iglesias and Zseby. In their work, many feature selection techniques such as mRmR, WMR SAM, and LASSO were employed to reduce 41 traffic features into 16 features. They utilized different classification algorithms such as DT, kNN, NB, LASSO-LAR, ANN and SVM with fivefold cross-validation and achieved detection accuracy ranging from 0.27 to 95.48 with mRmR generated features

(Iglesias & Zseby, 2015). Huda proposed a non-signature-based framework for the detection of malicious executables on API calls using hybrids of support vector machine wrapper and filter-based approach. Authors combined the feature's ranking score generated by the filters with wrappers approach to select the optimal feature sets that can characterize the real behavior of the malware.

Regarding the filter's methods, mRmR is employed to compute the API call's scores, through which SVM based wrapper heuristics is used as a classification algorithm. In the experimental results, mRmR–SVMS reveals a detection accuracy of 94.362% with 291 APIs (Huda et al., 2018). Darshan and Jaidhar presented a malware detection system MDS for distinguishing malicious files from the benign executables on the portable executable optional header fields (PEOHF) features using four filter-based techniques. In their work, they evaluated the performance of the classifiers along with features recommended by the filter's methods (Darshan & Jaidhar, 2018). However, the aforementioned approaches are insufficient to detect the advanced ransomware traits as they deal with from the malware perspective. Unlike the normal malware, the ransomware's effect is irreversible, therefore, it is important to identify the relevant features that describe the actual behaviour of the ransomware in the earlier stage of the attack.

Considering the available literature on ransomware detection using Windows API calls are suffering a massive amount of irrelevant and redundant system calls invoked by the malicious executables during its execution. Due to these independent calls can mislead the actual execution flow of ransomware binaries, and easy defeat these detection approaches. To fill the gaps in the current methods, we propose an approach for selecting the optimal windows API call features that can describe the real behaviour of the ransomware, it also reduces the size of the system call traces by discarding those system calls that do not have a strong indication for the behaviour characteristic of ransomware. Such an approach can be used as a detection method for the ransomware in the earlier phase of the attack.

We also propose an efficient dimensionality reduction technique for the system call traces collected from the dynamic analysis logs using the filter approach. The model development and performance evaluation are also discussed in the following sections. We will represent the gap among the researches, in order to identify the direction. The following Table 2.3 describes the overall detection techniques based on ML used by the researchers, including the amount of data set, the representation of the file, classifier algorithms used and result achieved.

Table 2.3. Summary of related research on ransomware detection.

Author, and title of the paper	Year	Data Sets	Analysis Approach	Detection Method
Gazet, A.Comparative analysis of various ransomware virii	2010	15	Static analysis	Author investigated the foundation of ransomware threats beyond the phenomenon. This study relies on a comparative analysis of various ransomware. The reverse engineering and a technical review are done at different levels: quality of code, malwares' functionalities and analysis of cryptographic primitives if any.
Ben22. Cryptolocker - using Powershell as a tripwire," Reddit	2013	41	Dynamic analysis	The tripwire idea utilizes the witness files that were monitored for modification or deletion. If a witness file is tampered with, the Lanman server service is stopped
Kharraz1 *et al.*Cutting the Gordian Knot: A Look Under the Hood of Ransomware Attacks	2015	1,359	Automati c analysis	Master File Table (MFT) and the types of I/O Request Packets (IRP) of the file system to detect multiple different types of destructive ransomware attacks that target users' files.
Shukla *et al*. POSTER: Locally Virtualized Environment for Mitigating Ransomware Threat.	2016	27	Dynamic analysis	Presented the gap in the existing state of art and extended the existing literature by adding new behavioral traits for new variants and describe a dynamic system which learns new behavior while under attack.
Chris. Detecting Ransomware with Honeypot techniques	2016		Static analysis	To implement a honeypot to detect ransomware activity, the File Screening service of the Microsoft File Server Resource Manager feature and EventSentry is done in Windows Security logs.
Ahmadian *et al*. 2entFOX: A Framework for High Survivable Ransomwares Detection	2016	1359	Dynamic analysis	Proposed a framework for high survivable ransomwares detection based on twenty appropriate features. In 2entFOX, after providing data and preprocessor step they designed a detection system with the help of Bayesian belief network to use extracted features and their statistical possibilities.

Kharraz *et al.*UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware	2016	148,223	Dynamic analysis	Present a novel dynamic analysis system called UNVEIL that is specifically designed to detect ransomware. To mount a successful attack, ransomware must tamper with a user's files or desktop. UNVEIL generates an artificial user environment and detects when ransomware interacts with user data.
Sgandurra *et al.*"Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection"	2016	582	Automati c analysis	Proposed an EldeRan framework for detection of ransomware that identifies the most important features of ransomware and machine Learning has shown is a viable and effective approach to detect new variants and families of ransomware for subsequent analysis and signature extraction
Palisse1 *et al.* Ransomware and the Legacy Crypto API	2017	39	Dynamic analysis	The first one takes advantage of the weak mode of operation used by some ransomware. The second one intercept calls made to Microsoft's Cryptographic API.
Saleh *et al.* "A 0-Day Aware Crypto-Ransomware Early Behavioral Detection Framework"	2018		Dynamic analysis	Proposed an early detection framework for Cyrto-ransomware that protects users using machine learning algorithms, to improve the accuracy of the detection authors also proposed anomaly detection technique to update from normal profile from extracted features
Zhang *et al*. "Classification of ransomware families with machine learning based on N-gram of opcode"	2018	1787	Static analysis	Proposed static analysis approach with opcode sequences feature for detection and classification of ransomware and also deal with ransomware that can fingerprint the environment. Multi-classification results indicate that this approach maps ransomware into families.
Poudyal *et al*. "A Framework for Analyzing Ransomware using Machine Learning"	2018		Static analysis	Eight supervised machine learning classifiers are employed to classify ransomware and benign sample. The framework achieved an accuracy of 97.95% when both extracted features are combined.
Hampton *et al*. "Ransomware behavioural analysis on windows platforms"	2018	14	Dynamic analysis	The experimental evaluation of this work achieved that the ransomware activities can be identified through a unique low-level system calls that are present in the ransomware.
Kok *et al*. "Prevention of Crypto-Ransomware Using a Pre-Encryption Detection Algorithm"	2019	582	Dynamic analysis	Authors proposed a pre-encryption detection algorithm (PEDA) that consisted of two phases. In, PEDA-Phase-I, with API and the PEDA-Phase-II, the signature repository that allows the detection of crypto-ransomware in the pre-execution stage.

2.12 Summary

In this section, we explored the various types of ransomware, including fake ransomware, Lockers, and crypto-ransomwares. We further classified the ransomware based on the threat type, the target approach that infects the victims, and the nature of infecting the systems. To execute successfully and to encrypt the user's related files, ransomware requires to carry out attack phases; this will lead the ransomware to spread and infect the machine. As common, we explained these phases briefly. Ransomware authors employ a range of different sophisticated techniques to spread their malicious intents; we highlighted the most common ransomware propagation method such as: spam emails, phishing, malware advertising websites, drive-by-download and exploit kits. To deeply understanding these infection vectors, malware analysts can effectively prevent the ransomware from spreading. To hide its malicious behaviour and intents, ransomware uses an avoidance technique, therefore, we also demonstrated the various techniques such as code injection that the ransomware injects the legitimate programs that enable ransomware to execute in the context of a legitimate application, making it easier to evade the detection. Malware writers use avoidance techniques in order to evade the detection of the antivirus software, such as encryption, compression data and obfuscation techniques. These techniques beat signature-based detection through transformation and changing the physical appearance of the virus. For extortion, ransomware needs to inform the ransom payment which is vary depending on the type of the ransomware variant and the worth digital currencies rates. Ransomware authors normally determine the ransom payments in bitcoins. The most popular cryptocurrencies are bitcoin. It is an electronic currency (there are no notes or coins) invented by Satoshi Nakamoto in 2008, so, we highlighted these payment methods in details. In addition, to get knowledge about the capability of ransomware, the structure techniques, and anti-reverse techniques used to hide it is self, and also the level of similarity to other malware samples, we need malware analysis. There are two types of malware analysis techniques; static and dynamic analysis. In static method, the malware is analysed and extracted the program's code without execution. But dynamic method executes the malware and exams its behaviour. Both provide complementary information about the malware. In this section, we discussed the ransomware detection methods in terms of the countermeasure of this threat which is very important, and many researchers proposed a range of different malware-detection techniques. Ransomware detection techniques mean any mechanisms which provide the detection of any form of ransomware that threatens to the computer. These techniques can be broadly classified into two categories: *approach-based detection* and *technique-based detection*. Signature based detection uses its characterization of the knowledge to decide the maliciousness of a program, that means it extracts the byte code patterns of each malware and compare these patterns with byte code of a program under its repositories. However, this method cannot detect new (unknown) ransomware whose signature has not been found or generated yet. On the other hand, anomaly-based detection uses its knowledge of what considered normal behavior to decide the maliciousness of a program under inspection. Hence this approach has benefits over signature based, because it detects any behavior that violates the norm, and has the ability to identify new malware. Since the malware writers are familiarizing the detection mechanisms and evading detection methods through modifying their malware shapes. Finally, we presented virus detection methods using supervised machine learning to map inputs to desired outputs using a specific algorithm. In supervised machine learning we need to collect data sets and to train the classifiers, the testing phase, the classifier identifies the previous unseen malicious files based on the training phase.

3. RESEARCH METHODOLOGY

This section presents the research methodology and provides hypothetical basis and foundation for our work. It also shows the general frameworks that are developed by the researchers based on the experiments. In this section will discuss the operational research frameworks, including data collection, analysing of data, experimental setups, proposed models and later the overall research plane. The main purpose of this section is to design a framework to detect ransomware files using supervised and semi-supervised machine learning as discussed in the literature review. For the performance metrics of this research, “effectiveness” and “performance” are important terms intended to introduce the overall accuracy of a specific machine learning algorithms. The output of this is to classify the files into benign and malicious based on these performance criterions which is false negative rates, and false positive, in addition to the average error rates of the outputs from supervised machine learning algorithms. The remaining of this section is organized as follows; the second subsection will discuss the proposed research frameworks that consist of three different methods. Every method contains a specific framework that illustrates the follow of the proposed method. In subsection 3, we describe the performance metrics to measure the accuracy of the classification algorithms. Finally, the conclusion of the section is also discussed.

3.1 The Proposed Methods

In this section, we demonstrate the research methods including the experimental design of the proposed frameworks and describe the method of the behavioural analysis approach in the sandbox. We also present the dimensionality reduction of the features for training and testing purposes. This section contains three different proposed research methods as illustrates in the following sections:

3.1.1 Method one

In this section, we present our proposed framework for the identification and detection of high survivable ransomware as shown in Figure 3.2. To make our methodology visual and understandable, we propose a methodology framework that consists of three main phases. Data collection and preparations phase that includes obtaining ransomware and benign dataset from a variety of sources, checking whether datasets are malware or not and vice versa and labelling the malware family using VirusTotal service. The second phase is to analyses samples using Cuckoo sandbox that generates JSON format report. The collected behavioural log files are passed to pre-processing tasks such as removal of duplicate files, file type identification, and parsing. The relevant features are extracted from the analysis file logs to get valuable feature sets. We have applied the term frequency and inverse document frequency (TF-IDF) algorithm for feature selection. Finally, supervised machine learning algorithms were implemented for the classification of ransomware and benign sample.

3.1.1.1 Experimental setup

To gain an in-depth behavioural analysis of ransomware requires executing samples in a controlled environment. Therefore, we built our malware analysis lab following the best practices suggested in (Nicolo Andronio, 2015). Cuckoo Sandbox is used, a well-known leading open source tool to automate malware analysis. Ubuntu 16.04 LTS Desktop fully updated was our host operating system while installing Cuckoo sandbox. WindowsXp_server_Pack3 32bit was selected as a guest machine due to its weaker security protections that enable us to observe more ransomware behaviour (Oktavianto & Muhardianto, 2013). To perform the analysis in a secure, Virtual box machine was used with controlled access to the Internet -host-only adapter- to enable commands and controls (C&C) communication, and to prevent the spread of ransomware.

Anti-virus, security updates, firewall, and user account control of windows XPSP3 guests were disabled to execute ransomware successfully. Some commonly third-party applications such as Microsoft Office, Acrobat Reader, Google Chrome and Mozilla

Firefox were installed in the guest operating system. Python agent was also installed that runs inside the guest and acts as cross-platform for communication and the exchange of data between cuckoo and the guest OS. Finally, in Windows XP several normal user files inside directories (e.g., My Documents, My Pictures and Videos, valid browsing history) were created to observe the behaviour activities of ransomware.

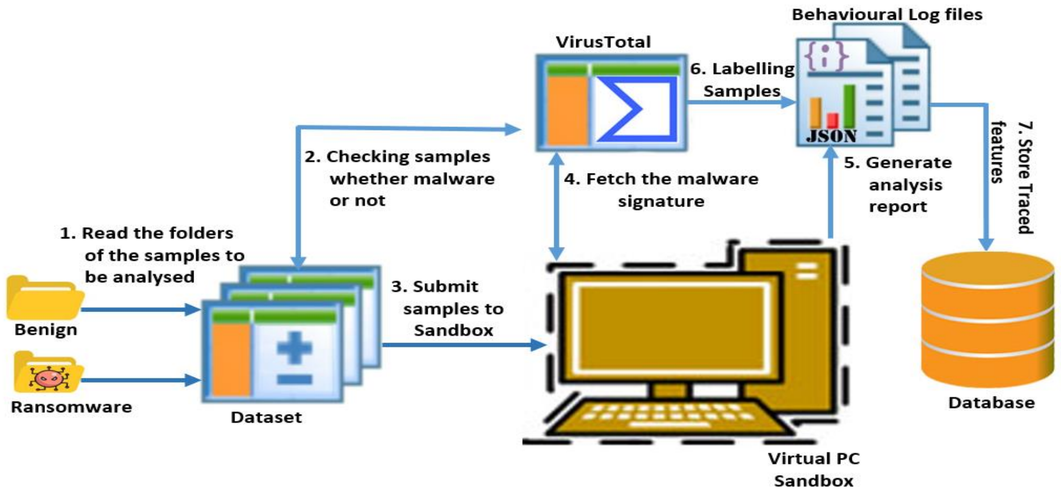

Behavioural Log files
VirusTotal
6. Labelling Samples
JSON
2. Checking samples whether malware or not
4. Fetch the malware signature
5. Generate analysis report
7. Store Traced features
1. Read the folders of the samples to be analysed
Benign
Ransomware
3. Submit samples to Sandbox
Dataset
Database
Virtual PC Sandbox

Figure 3.1: The Environmental setup for the behavior-based Ransomware detection.

3.1.1.2 Dataset description

The data set for this study consists of ransomware and benign. We collected 1,254 ransomware samples of 14 different families from several sources such as VirusShare and VirusTotal, - which are publicly computer virus repositories on the net-, we crawled malware repositories and online forums that share samples. We also downloaded and collected 1308 benign applications that hosted from the most trustworthy sources such as software.informer, and system files located in the “System32” directory of a fresh installed Windows 7 Professional.

To build a realistic dataset, we used in our experiments benign applications that have ransomware behaviour as shown in Table 3.1. The acquired samples are stored in separate files on both malware and benign group. To verify that the downloaded benign applications do not contain malicious components inside their payload, we double-checked the MD5 hash values from Virus Total service that has 57 common different antivirus software. To obtain the exact family name of ransomware is a very challenging task especially when you have a large number of malicius files. We applied Antivirus vendors' labelling scheme in terms of the popularity of ransomware classes among Antivirus Engines.

The general problem we encountered is mislabelling some samples by antivirus engines as specific ransomware family. Therefore, we parsed the labels by the set of AV engines that commonly used to assign malware labels using python script with a threshold value of 85 that aggregated the labels from the pool of AV in VirusTotal repository. We consider ransomware to be a specific family if 85% of AV engines described it as belonging to this family name.

Table 3.1: Distribution of malicious and benign files

Ransomware Class	**Samples**	**First Seen**	**Goodware Class**	**Application Name**	**Samples**
WannaCry	74	2017	Compression	Winzip, 7-zip, WinRAR, PeaZip, IZArc	225
Reveton	50	2012			
Torrent Locker	108	2012	Encryption	BitLocker, Disk Cryptor, VeraCrypt, TrueCrypt	172
Dirty Decrypt	51	2015			
CryptLocker	173	2013	Data Destruction	CBL Data Shredder, HDDErase , MHDD, PCDiskEraser, KillDisk , SDelete	401
Cerber	171	2016			
Trojan	82	2013			
Kollah	73	2014	Drivers Updater	Driver Booster, DriverPack Solution, DriveTheLife	230
Citroni	67	2015			
Pgpcoder	46	2015	Browsers	Chrome, Firefox, Opera ,Safari , Netscape, Internet Explorer	152
Kovter	23	2013			
Petya	89	2016	Multimedia tools	Canva, Animoto, Photopeach , Picasa, Livestream	182
CryptoWall	151	2014			
TeslaCrypt	96	2015	Others		96
Total Samples	**1254**				***1308***

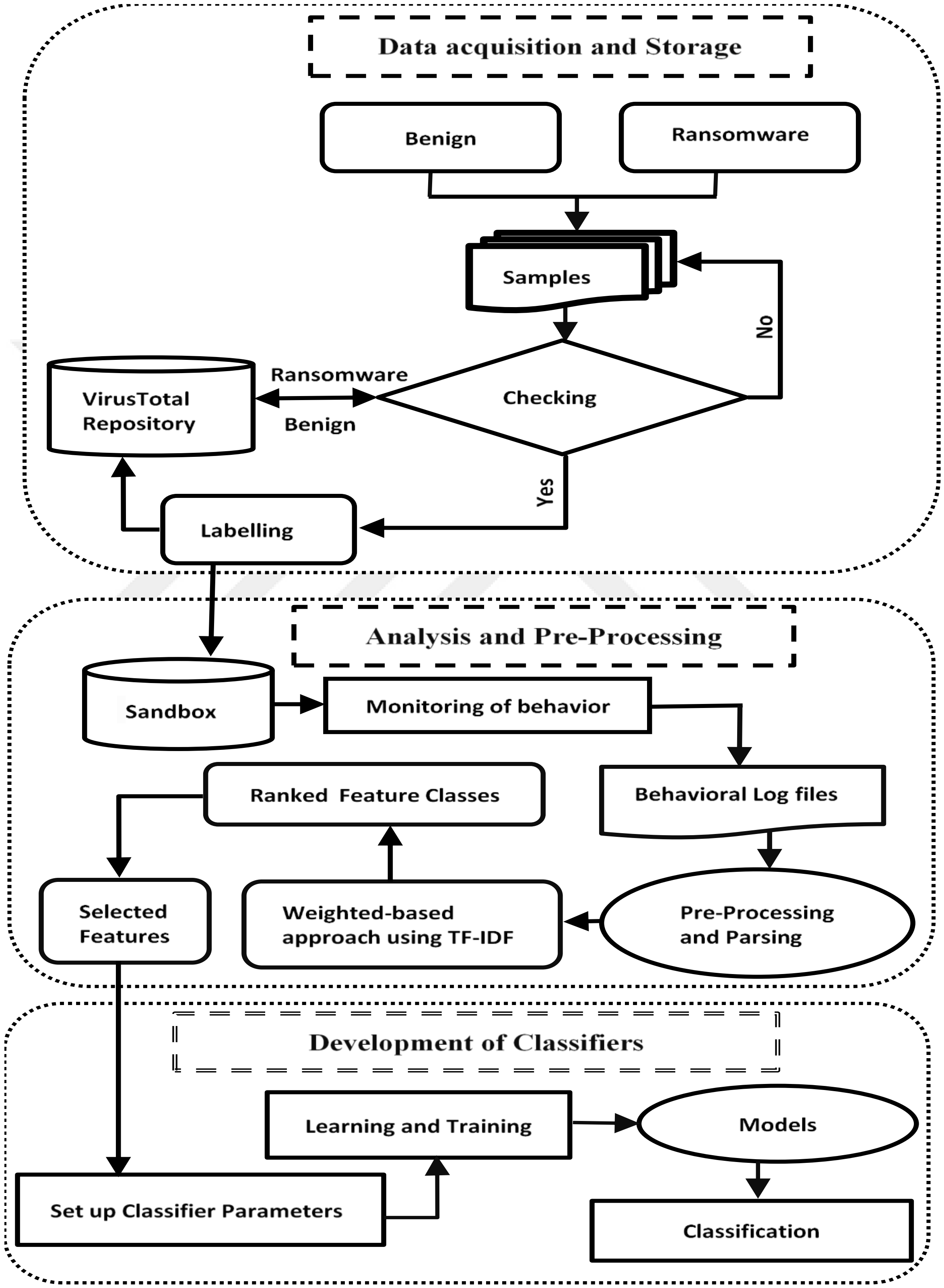
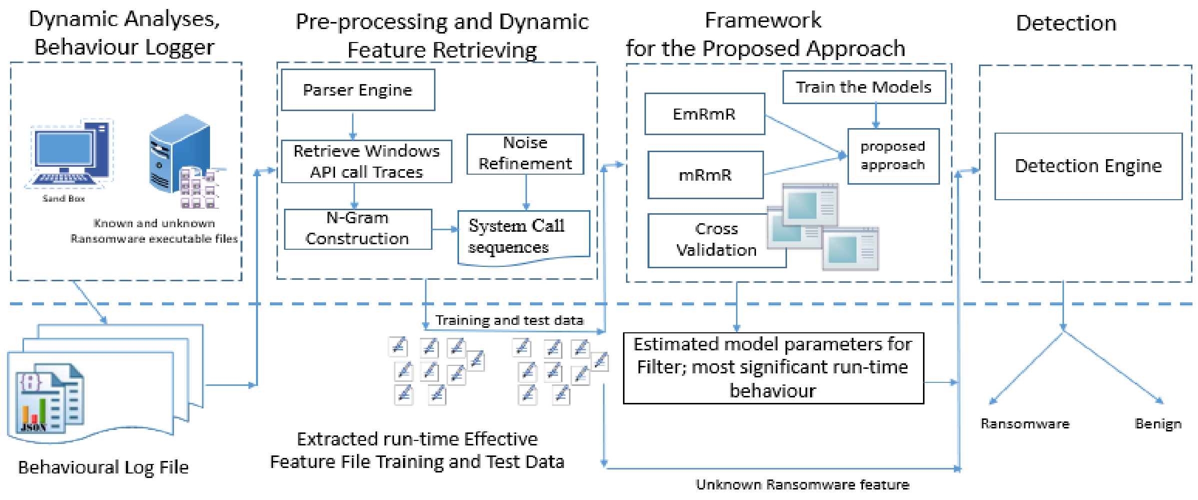


Figure 3.2. The Proposed framework architecture

3.1.2 Method two

In this section, we propose a non-signature-based detection framework for ransomware. This framework uses a behavioural-based analysis to identify malicious behaviour of applications and detect the ransomware in the earlier phases of the attack. The proposed framework includes different detection supervised algorithms to monitor running an application on a machine. For each application, the framework constructed a normal model based on the ransomware detection algorithm selected. This framework consists of four different phases as we will discuss in the following sections briefly.

To build a representative framework, we first need to collect the data set when the target system runs on the monitored environment. In this work, we monitor system calls because they are provided by the kernel and are used by programs running in user space. Indeed, all requests for applications such as network communication, file management or process-related operations must go through the kernel using the system call interface before they are executed. This system call provides precise information about the behaviour of an application.



Dynamic Analyses, Behaviour Logger
Sand Box
Known and unknown Ransomware executable files
Behavioural Log File
Pre-processing and Dynamic Feature Retrieving
Parser Engine
Retrieve Windows API call Traces
Noise Refinement
N-Gram Construction
System Call sequences
Training and test data
Extracted run-time Effective Feature File Training and Test Data
Framework for the Proposed Approach
Train the Models
EmRmR
mRmR
proposed approach
Cross Validation
Estimated model parameters for Filter; most significant run-time behaviour
Unknown Ransomware feature
Detection
Detection Engine
Ransomware
Benign

Figure 3.3: The architecture of the dynamic characteristics of behavior-based Ransomware detection.

The data collected from the dynamic analysis approach will then be passed to the processing unit for standardization. In order to simplify the process of analysing the data and creating a model of normal system behaviour, we process the data collected and we retrieve the traces from generated files. To create the sequences of the system calls, we contract N-gram construction to reduce the size of the system calls. Normally, Windows API calls are suffering a massive amount of irrelevant and redundant system calls invoked by the malicious executables during its execution. The performance of machine learning depends on the presence or the absence of noisy data. The existence of such noise in the data set could adversely impact the induction of ML models such as the increase in processing time, more storage requirement and the difficult analysis of real malicious intention that can lead overhead and poor prediction ability. Therefore, the Noise refinement process is applied to filter those system calls that do not have strong indication to the real behaviour of the program

3.1.3 Method three

In this section, we proposed a ransomware analysis and identification framework based on the runtime behaviour of ransomware and deep learning-based semi-supervised technique. Deep learning is a robust unsupervised approach that can extract the hidden intrinsic patterns from unsupervised feature space through a non-linear transformation and layered structure in which upper layers compute more abstract forms of features presenting the latent sources of variabilities in the feature space. The novelty of this proposed approach is that deep learning-based semi-supervised technique can extract dynamics of behavioural patterns from the new variants of ransomware obtained from the wild and can integrate the latent sources to the supervised classifier, making the detection engine independent of manual signature generation and robust to the changes.

The new contributions of this proposed model are that the model can extract the attack patterns of the ransomware through the deep learning-based semi-supervised method, ransomware from 14 families with a large number of features have been considered and a novel feature extraction procedure has been developed. Moreover, our model is highly scalable and adaptive. Since the model can learn the frequently changing behaviour of the ransomware and apply this knowledge to detect them, it ensures the zero-day detection.

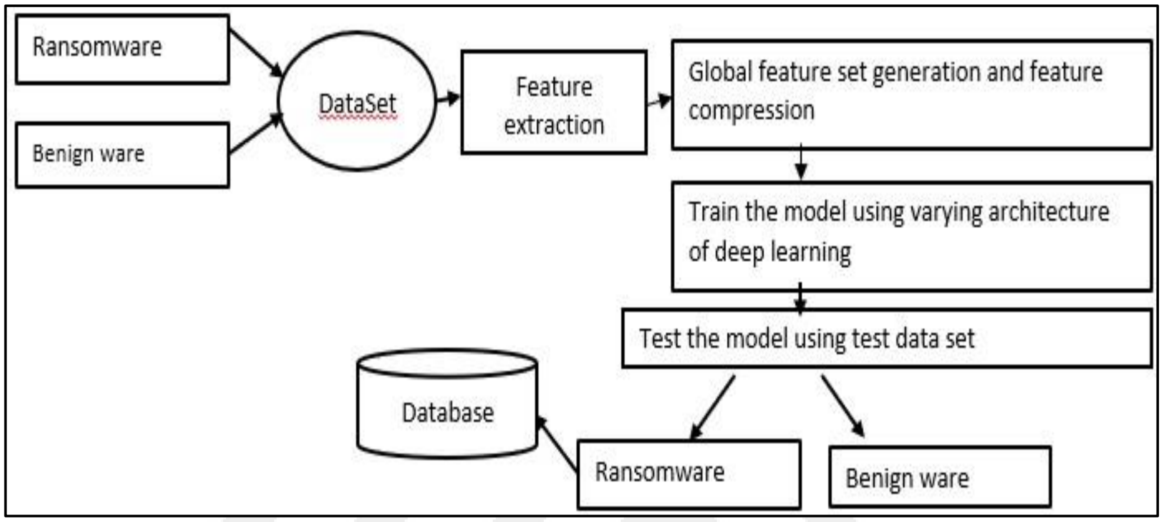

Ransomware
Benign ware
DataSet
Feature extraction
Global feature set generation and feature compression
Train the model using varying architecture of deep learning
Test the model using test data set
Database
Ransomware
Benign ware

Figure 3.4: Ransomware detection system using deep learning

We proposed a detection framework to detect the ransomware using deep learning-based approach. Deep Learning approach has the benefit of training the model using the extracted and selected features and behavioural patterns through hidden nodes in different layers. Since the cyber-attack patterns have been changed very frequently, the inherent cyber-attack patterns can be extracted using the multiple layers of abstraction of Deep Learning and represent he actual attack patterns to a non-linear and higher abstraction of the real scenarios which benefits the detection model. This key advantage of deep learning facilitates our model to achieve a higher accuracy rate. The data collection is the very first task of our detection model. The data set contains ransomware and benign ware. Pre-processing and feature extraction are done in the second phase. We have generated the global feature set which contains a large number of features, total 15972. FastICA has been considered to compress the features (Hyvärinen & Oja, 2000). After the feature selection, we generate the feature vector. The classifier is trained using the train data set. We have considered 10-fold cross validation to train and test the model. The performance of our detection model is evaluated using the test data set.

3.2 Performance Criteria

In this section, the classifier performance is evaluated using standard accuracy measurement. The best classifier models among the tested models are compared. The evaluations of these models based on their classification measurement such as True Positive Rate (TPR), is the case in which the proportion of positive samples, like ransomware that is identified correctly as shown in equation (10). False Positive Rate (FPR) is the case in which the proportion of negative instances wrongly identified as positive as shown in Equation (11). True Negative (TN): is the case in which samples are correctly classified as benign programs. False Negative (FN): is the quantity of numbers that are misclassified malicious programs.

$$TPR = \text{Sensitivity} = \frac{|TP|}{|TP|+|FN|} \tag{3.1}$$

$$FPR = Specificity = \frac{|FP|}{|FP|+|TN|} \tag{3.2}$$

The Total amount of accuracy is the ratio of properly identified samples, either negative or positive, divided by the total samples as defined in equation (12).

$$Total\ Accuracy = \frac{|TP|+|TN|}{|TP|+|FP|+|TN|+|FN|} \tag{3.3}$$

The total accuracy of the generated classifier determines the effectiveness and performance. Another method of identifying the performance of the classifier is the use of the ROC curve which is points of a plot that shows the trade-off between a classifier's FP rate and its TP rate.

For a clearer and more efficient representation of the TP rate as a function of FP, we used the receiver efficiency function also known as ROC (Receiver Operating Characteristic) curve name. This curve provides an estimation of the optimal value of the detection threshold allowing a compromise to be achieved between the TP and the FP. A ROC curve is, therefore, a plot of true positive rates against the false positive rate for different detection thresholds.

We also measured the area under the ROC curve (AUC: Area Under Curve). AUC is generally used to compare the performance of detectors independently of decision thresholds. AUC = 1 indicates a perfect detector that detects all anomalies without false alarms (TPR = 1, FPR = 0), while a random detector will have an AUC = 0.5. The larger the AUC value, the more the curve moves away from the line of the random classifier (linear straight line) and approximates the angle of the ideal classifier.

It's good to have a TP of 100% and a FP of 0%, but these alone do not allow us to properly estimate the performance of the detector. For this, we also measured the number undetected ransomware attacks, calculating the Total Accuracy Rate (ACC) according to formula 3.3, where TN denotes the rate of true negatives (the proportion of normal traces correctly classified as normal on the total number of normal traces in the test set) and FN denotes the rate of false negatives (the proportion of ransomware attacks incorrectly classified as normal to the total number of abnormal traces in the test set).

3.3 Summary

This section presented an overview of the research methods. We discussed in detail the operational research frameworks which show the overall project operation. These frameworks consist of three different methods; we highlighted every unique method by presenting the steps to be followed. The researcher also discussed the procedures inside the operational frameworks, such as the steps involved in data pre-processing, ransomware analysis approach, refining process and the N-gram constructions. The approach to extract the Windows API calls function features in the pre-processing are argued which should be performed before data set are passed to the ransomware classification part. For defining which supervised machine model is the best, each algorithm should apply and evaluate separately. After evaluation all possible combinations, the best combination models determine final proposed scheme. At the end of this section, the performance criteria were discussed.

4. ANALYSIS AND DATA PREPROCESSING

This section describes the preparation and the analysis of the data of this research derived from the previous section. The reason for the data pre-processing is to create appropriate, clean and normalized data format that to be input into machine learning-based classifiers. The format of our data set is portable executables based on the windows family that compromised of clean and malicious programs. The first subsection in this section highlights the executable file format used.

The second subsection discusses the analysis of the malicious file executable to monitors the behaviour of the ransomware using a sandbox environment. Feature engineering that contains feature retrieving which is the process of extracting data from specific files to get a set of informative and non-redundant feature. In this subsection two different features are extracted; integrated features that contains combined seven important ransomware behaviour; and the system calls which is an interface that the program requests a service from the kernel operating system. In the third subsection of this section, we presented a system call refinement process to reduce the size of Windows system call traces gathered from the dynamic analysis by removing those system calls that do not discover the main behaviour of the ransomware.

Finally, the feature selection method to eliminate the processing overheads of the dataset in training and testing phases, and improve the accuracy rate of classifiers are described in this subsection. We demonstrated the different feature selection methods such as term frequency-inverse document frequency (TF-IDF) to weight the term based on its inverse document frequency, Maximum-Relevance and Minimum-Redundancy (mRmR) which is a well-known filter algorithm that intended to find features with high relevance with the target class and low redundancy among other features. In addition, another important feature selection is employed, which is a FastICA that was developed from Independent Component Analysis (ICA), and the concepts and principles of independent component analysis still apply to constrained independent component analysis that has been considered to compress the features.

4.1 Executable File Format

Files can be divided into two main categories; data file which is designed to store information, and executable files that consist of a collection of information describes some task in the computer. Generally, computer files composed at least one executable file with the help of data file, as the same situation in malware. The structure of a portable executable is important because the writer of malicious files use portable executable (PE) as a vehicle to transform their malicious intent to the target.

Microsoft uses portable executable as standard to all executable file formats under 32-bit and 64-bit versions of Windows family operating systems. The term portable indicates the portability and versatility of the file in different windows environments. The PE format "is a data structure that encapsulates necessary information so that Windows OS loader can manage wrapped executable code"(T.-Y. Wang, Wu, & Hsieh, 2009).

The PE file can be divided into two main portions; header and body. The header which contains essential information used to load a PE file such as MS-DOS header. The content of a PE file is composed of section that is intended to store information. This includes, Section Tables, API import and export Tables, resource management data, PE section and any other information related to the windows loader in order to execute code successfully (Faruki, Laxmi, Gaur, & Vinod, 2012).

The first part of the PE file structure starts with the MS-DOS header called an IMAGE_DOS_HEADER. This is the place where PE is stored and contains two primary values called e_magic and e_lfanew. A PE file Header consists of the PE Signature, the File Header, the Optional Header, and also it determines how many sections are in the PE. The section Table which is an array of IMAGE_SECTION_HEADER structures contains Data Directories that gives the information related to sub sections such import table, export table and resource table. The export table provides the name of the relative virtual machine of all exported functions in the given machine. Therefore, this project will focus on the imported and exported function to extract the features (T.-Y. Wang et al., 2009).

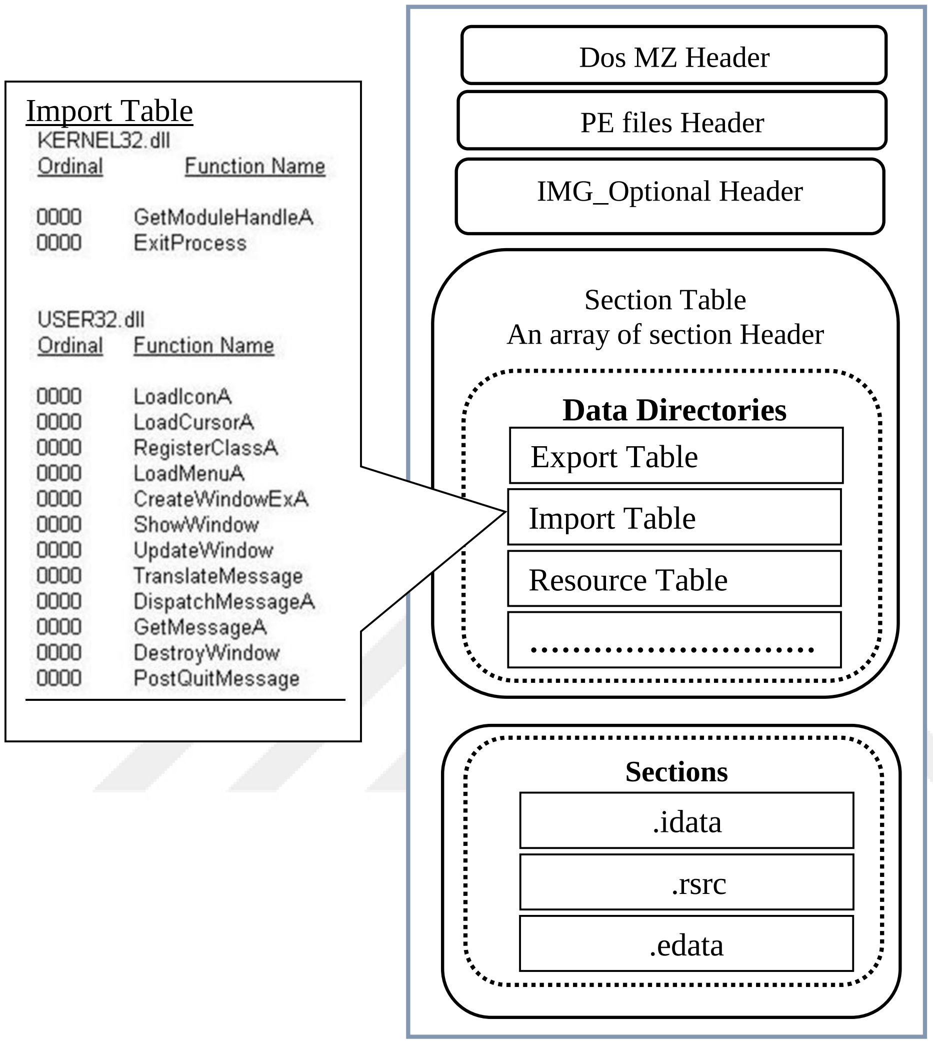


Figure 4.1: PE File Structure

4.2 Analysis of Executable Files

In this section, an automated dynamic analysis was used to analyses the samples in an isolated environment. This analysis collects a large number of ransomware samples and monitors their behaviour using a sandbox environment. Although the total original dataset was 2562, after removing samples that did not execute correctly, or cuckoo terminated the analysis because of the maximum timeout that we set samples to run, or those that many

AV assigned different ransomware family names, 673 ransomwares from 14 distinct families and 742 benign samples were analysed.

Given the finite size of the RAM, it is necessary to limit the analysis to about 5 or 15 minutes. It depends on the sample activity and the number of different samples that are analysed or intercepted (e.g., writing, reading, etc.). Otherwise, the lack of space leads to the loss of the analysis. Therefore, every sample up to a range of 4 until 9 minutes were analysed to show to their malicious behaviour and to capture the execution traces of the samples using a cuckoo sandbox, while the ransomware sample is running on the host. Cuckoo monitored and recorded information in terms of the API calls, network traffic, changes of files and folders, processes and memory dumps(Oktavianto & Muhardianto, 2013). We used virtualization software to take a snapshot of the guest machine before the execution of each malware sample, after execution, the entire system was reverted to a previous clean state before the infection.

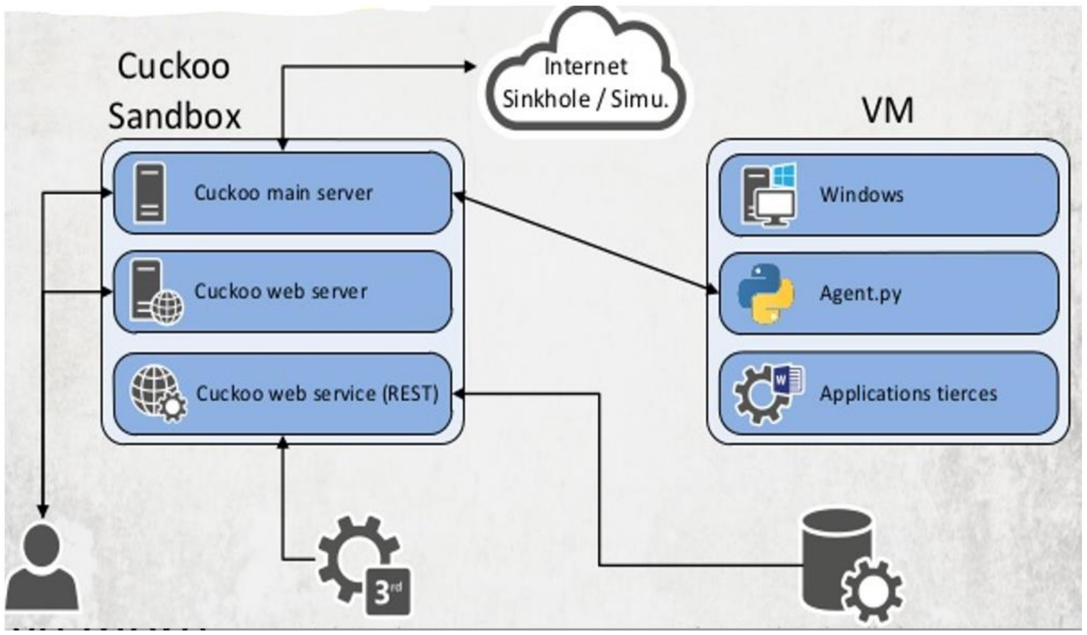
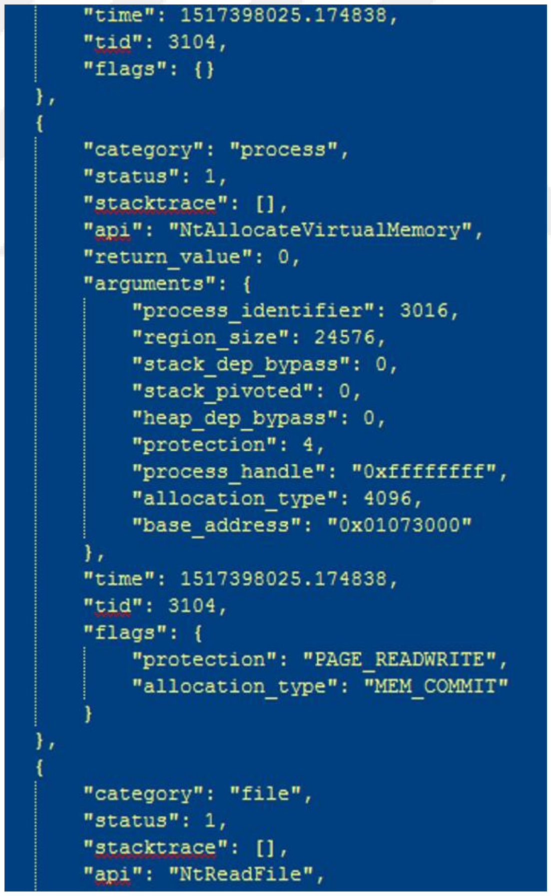


Figure 4.2: The architecture of the Sandbox

Indeed, some ransomware seeks to detect controlled environments and avoid expressing their malicious behaviour if they succeed. While other ransomware samples wait for human interaction like mouse or keyboard event before executing their malicious activities, thus, we used a python script that performs basic user's activity such as browsing websites, clicking and deleting documents and folders on the desktop. During the analysis of the samples, we observe ransomware variants used both symmetric and asymmetric algorithms to encrypt the user's data. Crypto ransomware creates a randomly symmetric key with the AES algorithm in victim's machine and then encrypts files along with that generated key. After encrypting the data, it encodes the secret key with asymmetric encryption. At the end of execution, the time taking ransomware samples to encrypt files is variety, ranging from 27 seconds like Petya up to 2 minutes for CryptoWall.



```
    "time": 1517398025.174838,
    "tid": 3104,
    "flags": {}
},
{
    "category": "process",
    "status": 1,
    "stacktrace": [],
    "api": "NtAllocateVirtualMemory",
    "return_value": 0,
    "arguments": {
        "process_identifier": 3016,
        "region_size": 24576,
        "stack_dep_bypass": 0,
        "stack_pivoted": 0,
        "heap_dep_bypass": 0,
        "protection": 4,
        "process_handle": "0xffffffff",
        "allocation_type": 4096,
        "base_address": "0x01073000"
    },
    "time": 1517398025.174838,
    "tid": 3104,
    "flags": {
        "protection": "PAGE_READWRITE",
        "allocation_type": "MEM_COMMIT"
    }
},
{
    "category": "file",
    "status": 1,
    "stacktrace": [],
    "api": "NtReadFile",
```

Figure 4.3: The generated JSON format

At the end of the analysis, a trace containing a format Compressed JavaScript Object Notation (JSON) is sent to the analysis phase. The trace in JSON format is then easily comprehensible to understand how the ransomware behaves. The information is stored with the granularity of a thread (i.e., thread). An uncompressed trace ranges from 20 MB to 200 MB. Traces are also analysed manually, to eliminate samples with similar behavior which do not are not triggered a second time.

4.3 Feature Engineering

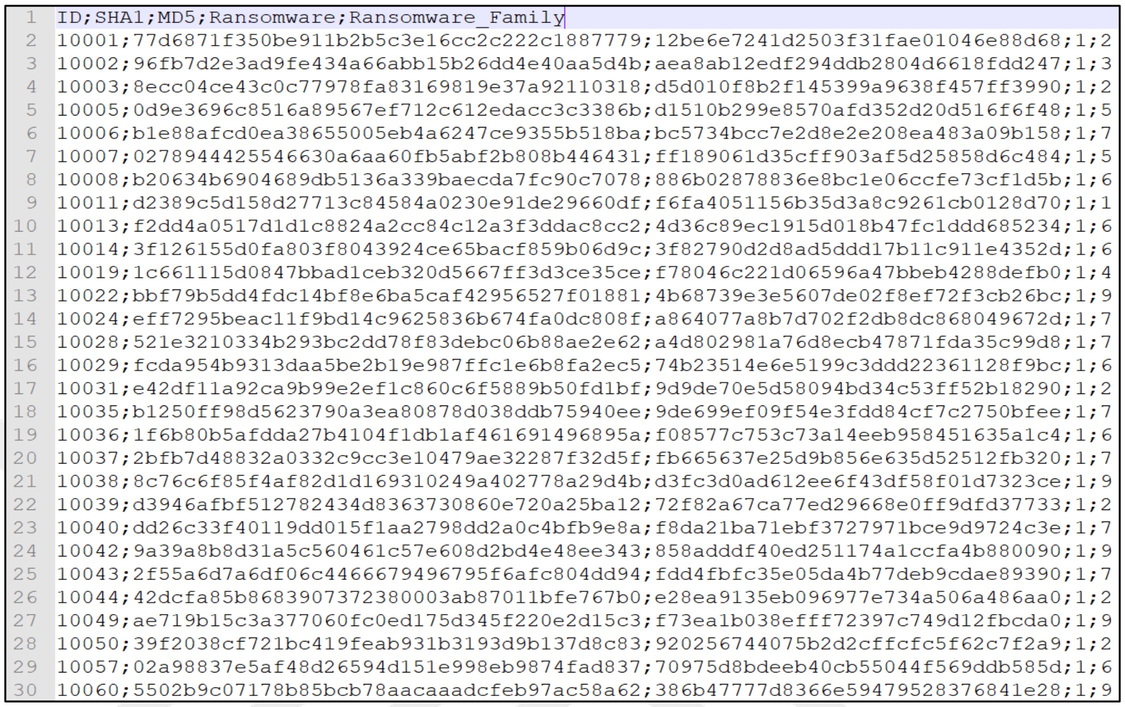
In this subsection, we present the reduction of the data dimensionality of this research. The process of extracting data from specific files is called feature extraction and the aim is to get a set of informative and non-redundant data not only made by a selection of certain features (Yang & Pedersen, 1997). After we construct new features, we will select the important features by removing those features that have the same behaviours. The following subsections are described several commonly used reduction techniques. They are generally grouped into two categories: feature retrieving or extraction and feature selection.

4.3.1 Feature extraction

In this subsection, we will present the process of retrieving features from generated JSON files. in this approach we extract two main types of features, integrated features that includes seven types of features such as Registry, files Operation and etc., and Windows System calls features, the following are explained in details

4.3.1.1 Extracting integrated features

Once the analysis is completed, cuckoo generated human-readable JavaScript Object Notation (JSON) report for each analysed malware sample. In this study, the most time dedicated to the extraction of the indicative and accurate behavioural features from JSON report, which is not an easy task. After we collected the results of the analysis, we need to retrieve the key elements from the JSON reports such as SHA1, MD5, ransomware ID and ransomware family as shown in Figure 4.4. These elements indicate the importance of the ordering samples, it also prevents the sample extraction redundancy by following the SHA1 and MD5 unique numbers.



```
ID;SHA1;MD5;Ransomware;Ransomware_Family
10001;77d6871f350be911b2b5c3e16cc2c222c1887779;12be6e7241d2503f31fae01046e88d68;1;2
10002;96fb7d2e3ad9fe434a66abb15b26dd4e40aa5d4b;aea8ab12edf294ddb2804d6618fdd247;1;3
10003;8ecc04ce43c0c77978fa83169819e37a92110318;d5d010f8b2f145399a9638f457ff3990;1;2
10005;0d9e3696c8516a89567ef712c612edacc3c3386b;d1510b299e8570afd352d20d516f6f48;1;5
10006;b1e88afcd0ea38655005eb4a6247ce9355b518ba;bc5734bcc7e2d8e2e208ea483a09b158;1;7
10007;0278944425546630a6aa60fb5abf2b808b446431;ff189061d35cff903af5d25858d6c484;1;5
10008;b20634b6904689db5136a339baecda7fc90c7078;886b02878836e8bc1e06ccfe73cf1d5b;1;6
10011;d2389c5d158d27713c84584a0230e91de29660df;f6fa4051156b35d3a8c9261cb0128d70;1;1
10013;f2dd4a0517d1d1c8824a2cc84c12a3f3ddac8cc2;4d36c89ec1915d018b47fc1ddd685234;1;6
10014;3f126155d0fa803f8043924ce65bacf859b06d9c;3f82790d2d8ad5ddd17b11c911e4352d;1;6
10019;1c661115d0847bbad1ceb320d5667ff3d3ce35ce;f78046c221d06596a47bbeb4288defb0;1;4
10022;bbf79b5dd4fdc14bf8e6ba5caf42956527f01881;4b68739e3e5607de02f8ef72f3cb26bc;1;9
10024;eff7295beac11f9bd14c9625836b674fa0dc808f;a864077a8b7d702f2db8dc868049672d;1;7
10028;521e3210334b293bc2dd78f83debc06b88ae2e62;a4d802981a76d8ecb47871fda35c99d8;1;7
10029;fcda954b9313daa5be2b19e987ffc1e6b8fa2ec5;74b23514e6e5199c3ddd22361128f9bc;1;6
10031;e42df11a92ca9b99e2ef1c860c6f5889b50fd1bf;9d9de70e5d58094bd34c53ff52b18290;1;2
10035;b1250ff98d5623790a3ea80878d038ddb75940ee;9de699ef09f54e3fdd84cf7c2750bfee;1;7
10036;1f6b80b5afdda27b4104f1db1af461691496895a;f08577c753c73a14eeb958451635a1c4;1;6
10037;2bfb7d48832a0332c9cc3e10479ae32287f32d5f;fb665637e25d9b856e635d52512fb320;1;7
10038;8c76c6f85f4af82d1d169310249a402778a29d4b;d3fc3d0ad612ee6f43df58f01d7323ce;1;9
10039;d3946afbf512782434d8363730860e720a25ba12;72f82a67ca77ed29668e0ff9dfd37733;1;2
10040;dd26c33f40119dd015f1aa2798dd2a0c4bfb9e8a;f8da21ba71ebf3727971bce9d9724c3e;1;7
10042;9a39a8b8d31a5c560461c57e608d2bd4e48ee343;858adddf40ed251174a1ccfa4b880090;1;9
10043;2f55a6d7a6df06c4466679496795f6afc804dd94;fdd4fbfc35e05da4b77deb9cdae89390;1;7
10044;42dcfa85b8683907372380003ab87011bfe767b0;e28ea9135eb096977e734a506a486aa0;1;2
10049;ae719b15c3a377060fc0ed175d345f220e2d15c3;f73ea1b038efff72397c749d12fbcda0;1;9
10050;39f2038cf721bc419feab931b3193d9b137d8c83;920256744075b2d2cffcfc5f62c7f2a9;1;2
10057;02a98837e5af48d26594d151e998eb9874fad837;70975d8bdeeb40cb55044f569ddb585d;1;6
10060;5502b9c07178b85bcb78aacaaadcfeb97ac58a62;386b47777d8366e5947952837684 1e28;1;9
```

Figure 4.4: A snippet of extracted key elements from the JSON file.

The size of the report generated by the sandbox occupies hundreds of MBs, analysing and examining each report manually is experimentally infeasible, therefore, we build our own parsing algorithm to convert JSON formatted string representations to key-pair objects. The feature parsing reads the JSON files from all sandbox output reports and then parsed to get the required features to reduce the search space. The feature parser functions as a structure to correlate a ransomware sample's feature calls into states. The parser maps the Registry paths, Windows API calls, files Operation, Strings, Directories, Drops and libraries into seven different states.

Indeed, a set of matrices is created from the features set. For example, a matrix of calls is created to the Windows API, which denotes the presence or absence of a function during an analysis. Similar behaviour is achieved for other characteristics. This allows you to add many combinations. These approaches make it possible to model binary variables, for example the absence or the presence of a registry key. Such a model makes it possible to link an event (i.e., ransomware) to a combination of variables (i.e., features).

Algorithm 4.1: Extracting Features from JSON Report

Input: Set of JSON report path J_R that contains a number of behavioral and static features f.

Output: Parsed files

```
1.  Function ObtainFeature ($S_{data}$, states) {
2.      for process in json_data['behavior'] ['processes'] do
3.       if json_data_process is equal to states then
4.       set first_seen_temp=process_first_seen
5.          if first_seen is greater than first_seen_temp or equal to
            zero
6.             set first_seen= first_seen_temp
7.          for features in json_data_process[F] do
8.             if features [F] not in our_Dictionary_features then
9.                 set our_Dictionary_Features [F] and timestamps =
                   f and Feature_time
10.                Our_Dictionary_ Features [F][count]=1
11.            Else set our_Dictionary_ Features and timestamps=
               Features_F_time and append_F
12.            our_Dictionary_Features [F][count]+1, return
               first_seen, our_Dictionary_Features [F]}
13.  Function Json_Files (JR, $P_R$ )
14.     for ($J_R$, $P_R$) in G_file() do //traverse the file names in a
        directory tree
15.     for i, name in enumerate [all_files] do
16.           If name ends with ('json') then
17.               $F_{name}$= initialize files that matches (name)
18.               Open the json data ($J_R$ + $F_{name}$)  as ($json_f$)
19.               Set $S_{data}$= load ($json_f$)
20.               Call the function of ObtainFeature ( $S_{data}$ ,true)
21.           Print($P_R$)}
22. In the main function {
23.     Input $J_R$ ← parse_directory//initialize the director to be
        parsed
24.     Input $P_R$ ← F   // set the place where the result will be
        stored
25.     Store the user's input in the ($J_R$, PR ) variables
26.     Json_Files( $J_R$, $P_R$ )
27.        if name is equal to main then
28.          exit the system
```

Every state represents the presence or the absence of that specific call for this feature. The Feature parser creates a matrix containing the feature and its states. For instance, in ransomware phases, when the attached completed, the ransomware deletes all the original victim's data while keeps the encrypted one, in this case, RegDelete method is used. So, the parser creates a matrix by investigating whether this specific registry key operation was performed or not.

Table 4.1: Feature classes and the number of extracted features

#	Feature Classes	No of Features	Analysis Type	Feature Class Description
1	Registry Paths	3208	Dynamic	Registry key operations such as registry keys opened, read, written and deleted
2	Windows API Calls	4661	Dynamic	Windows API calls the traces of invocations of native functions and API calls
3	Files Operations	3210	Dynamic	File operations such as read, open, write and delete operations
4	Printable String Information (PSI)	836	Static	Is a sequence of characters that provide hints about the functionality of a program
5	Directory Operation	582	Dynamic	Operations performed on a directory
6	Cryptographic Libraries	948	Dynamic	Contains and implements several popular cryptographic algorithms and standards.
7	Dropped Files	186	Dynamic	During installation application dropped set Extensions of files

4.3.1.2 Extracting System Calls

In this section, we first present the process of capturing system call traces by executing samples in a controlled environment. The behavioural log files collected from the sandbox are then passed to the extraction stage to parse and obtain the valuable Windows API calls. In this work, we monitor system calls because they are provided by the kernel and are used by programs running in user space. Indeed, all requests for applications such as network communication, file management or process-related operations must go through the kernel using the system call interface before they are executed. This data provides precise information about the behaviour of an application (Zavarsky & Lindskog, 2016). This is motivated by the assumption that once compromised, the system call traces of an application will be different from those generated by its original version. The following subsections explain the process in more detail.

A. Tracing System Calls

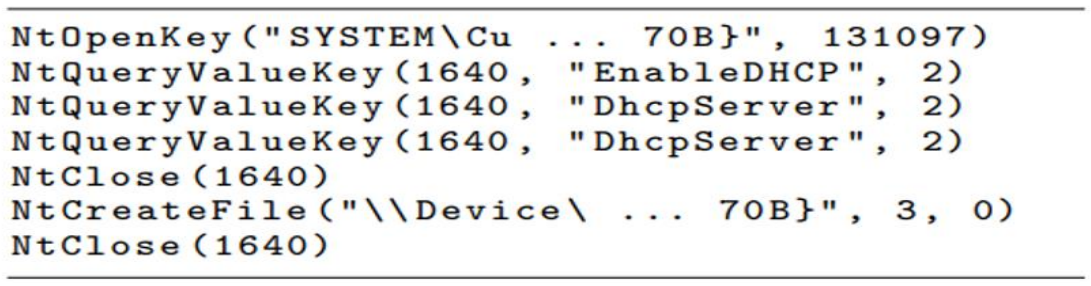
To execute the suspicious payload and infect the host system, ransomware needs to invoke the system API calls successfully. These system calls are identified as a feature for distinguishing malicious files from the benign ones. Since system calls are an interface that the programs use to request service from the operating system's kernel, tracing these system calls could help to indicates the ultimate intent of the program.

Definition 4.1: *System call trace* is a robust technique to record the execution of the process dynamically in a controlled environment, let P be a process and the input of the process is I. If the process P invokes the system call S, where $P \in S$ with input of I , the system call trace can be defined as:

$$S_t = (\mathrm{P} \in S\,, I) \tag{4.1}$$

The entire trace consists of a list of system calls with different parameters and return-values of the process. To capture the program's dynamic behaviour traces, ransomware sample was executed in an isolated environment. Every sample up to a range 70 until 90 seconds were analyzed to show to their relevant malicious indicator. The sandbox intercepted and recorded the Windows API calls, while the ransomware sample is running on the host.

We used virtualization software to take a snapshot of the guest machine before the execution of each ransomware sample. After execution, the entire system was reverted to a previous clean state before the infection. Similar to the ransomware samples, benign applications are also executed in the same isolated environment to obtain their behavioural features.



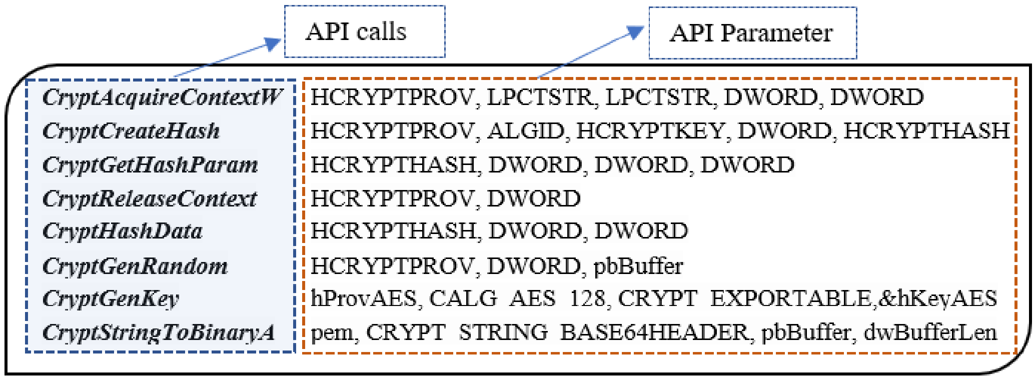
```
NtOpenKey("SYSTEM\Cu ... 70B}", 131097)
NtQueryValueKey(1640, "EnableDHCP", 2)
NtQueryValueKey(1640, "DhcpServer", 2)
NtQueryValueKey(1640, "DhcpServer", 2)
NtClose(1640)
NtCreateFile("\\Device\ ... 70B}", 3, 0)
NtClose(1640)
```

Figure 4.5: An example of System Call traces

To trace the system calls, ransomware and benign samples need to complete its execution time, but some ransomware waits for human interaction like clicking the mouse or pressing a button in the keyboard event before executing their payload. Thus, we utilized a customized python script that works in conjunction with the Sandbox to perform basic user's activities such as browsing websites, clicking and deleting documents and folders on the desktop. After the execution, the output of the sandbox is a log of JSON (human-readable JavaScript Object Notation) file for each analysed ransomware and benign sample. Although this report contains different categories of ransomware analysis results, we limited our scope to the behavioural category that describes the dynamic characteristics of the ransomware.

B. Obtaining features from System Call traces

Once the behavioural log was collected, the size of the report generated by the sandbox occupies hundreds of MBs. Analysing and examining each report manually is experimentally infeasible, hence, we built a parser engine as shown in algorithm 4.2 to retrieve the behavioural features from the output file. The feature parsing reads the files from all Sandbox output reports and then parsed them to get the required system calls to reduce the search space. To decrease the trace sequences, only the list of the window API call function names that correspond to the process was extracted while the parameters and return-values were ignored. A snippet of the critical cryptographic API calls and its parameters are shown in Figure 4.6.


API calls
API Parameter

CryptAcquireContextW	HCRYPTPROV, LPCTSTR, LPCTSTR, DWORD, DWORD
CryptCreateHash	HCRYPTPROV, ALGID, HCRYPTKEY, DWORD, HCRYPTHASH
CryptGetHashParam	HCRYPTHASH, DWORD, DWORD, DWORD
CryptReleaseContext	HCRYPTPROV, DWORD
CryptHashData	HCRYPTHASH, DWORD, DWORD
CryptGenRandom	HCRYPTPROV, DWORD, pbBuffer
CryptGenKey	hProvAES, CALG AES 128, CRYPT EXPORTABLE,&hKeyAES
CryptStringToBinaryA	pem, CRYPT STRING BASE64HEADER, pbBuffer, dwBufferLen

Figure 4.6: Log of Critical Cryptographic API Calls

To prune the collected behavioural logs of the sample's execution trace, we applied the following rules:

- **First**, the system process ID of the ransomware samples is identified.
- **Second,** we explored the exact position where the process created the new thread or child process by iterating the system call logs.
- **Third,** the system calls were sorted chronologically based on the timestamp.
- **Finally,** relevant function calls are extracted semantically from the log to construct the representative feature vector.

Not only the extracted output traces contain the original system calls, but also irrelevant and noisy calls generated by the ransomware to hide its malicious behavior. This adversely affects the performance of classifiers and increases the false positive rate. To enhance the detection accuracy and to obtain the valuable system calls, the retrieved features are transferred to another process to refine the collected traces, as described in the following subsequent sections.

It is possible to modify the control flow of a program in the user space using a hook, more commonly known as a hook. A malicious program can thus hide files, processes, network connections, etc., to one or more of its peers. For this, the ransomware must access the memory space of the victim process and modify one or more addresses in virtual memory. One method is to modify the IAT to intercept the call to the desired legitimate function than to transmit it to a routine that the attacker controls (Francillon & Castelluccia, 2008). This technique is easily detectable and many more techniques evolved to exist.

Algorithm 4.2: Effective System calls Generation

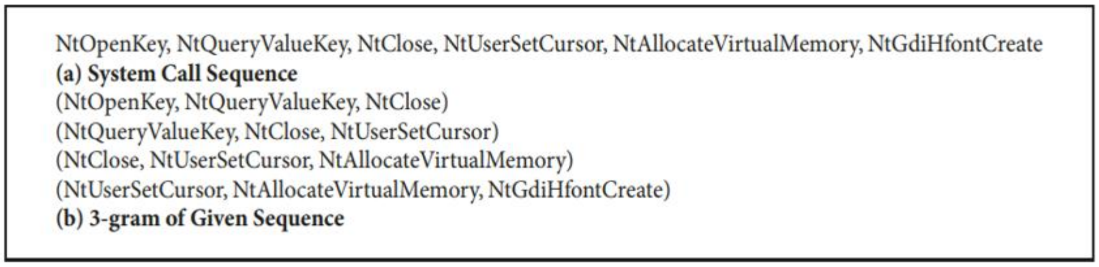
Syntax Definitions:

Let p be *process*, F_s be first seen, log file $\{l_f\}$, *System Call* $\{s\}$, Database $\{DB\}$, Dynamic behaviour $\{B_d\}$, *timestamps* $\{T_s\}$ *and directory tree* J_R

```
Input: File contains the system call traces $S_t = (P \in S, I)$
       and features $(f_1, f_2, f_3, \ldots f_n)$.
  Output: List of system calls $l \in \{s_1 s_2 s_3 \ldots s_t\}$
1    Procedure FindSysCalls ($S_t$, Condition) {
2        for each p in $l_f$ ['$B_d$'] ['$P_d$'] do
3         if $l_f\{P_d\}$ = Condition then
4          $F_s$\{temp\} ← $F_s$
5          if $F_s$ >= $F_s$\{temp\}
6             $F_s$← $F_s$\{temp\}
7             for each traces in $l_f\{P_d\}$ [$S_t = (P \in S, I)$] do
8               if traces [$s$] do not match any SysCall then
9                  Id= $s$, update $DB$ and $T_s$= $s\{T_s\}$
                   and $DB\{s\}$[count]=1
10              Else $DB\{s\}\{T_s\}$ ← append [$s\{T_s\}$]
11              $DB\{s\}$[count]+1, return $DB\{s\}$
           end for
12    Procedure Loadfile (JR, $P_R$ )
13       for ($J_R$, $P_R$) in G_file() do//traverse the directory tree
14       for i, N in enumerate [all_files] do
15              If N ends with ('json') then
16                          $F_n$ ← (N) //file initialization
17                          $S_{data}$ ← load ($J_R$ + $F_n$)
18                          FindSysCalls ( $S_{data}$ ,true )
19   End
```

C. N-Grams Construction

To construct the sequential behaviour on the system call log of each sample and remove the subsequence that have a little effect on the detection, a text-based n-grams method is employed to combine the system call traces that appear in a consecutive order. We created N-grams of length 2, 3, 4 and 5 from the system call traces corresponding to each analyzed sample. We just extracted n-grams system calls that collected from the same category, for instance, the *NtTerminateThread*, *NtTerminateThread*, and *NtSuspendThread* are 3-gram system call sequences for process and thread that aim to terminate and suspend a specified threat. The aim is to calculate the n-gram distribution by generating a fixed- size slice window through enumerating the appearance of each gram. The extracted result of the n-gram system call trace construction is the vector V that contains the occurrence of every possible *n-gram* as shown in Box 4.1.


NtOpenKey, NtQueryValueKey, NtClose, NtUserSetCursor, NtAllocateVirtualMemory, NtGdiHfontCreate
(a) System Call Sequence
(NtOpenKey, NtQueryValueKey, NtClose)
(NtQueryValueKey, NtClose, NtUserSetCursor)
(NtClose, NtUserSetCursor, NtAllocateVirtualMemory)
(NtUserSetCursor, NtAllocateVirtualMemory, NtGdiHfontCreate)
(b) 3-gram of Given Sequence

Box 4.1: A snippet of 3-gram System Call sequences

After generating n-gram frequencies, the total number of features extracted from the system call trace using *2-grams,3-grams, 4-grams,* and *5-grams* are ***32134*** and ***76908***, ***186908***, ***227151,*** respectively. However, the volume of these features is quite large in terms of training time and memory usage. Since all these features do not contribute to the classification of malicious samples, the noisy subsequence with little influence on the detection accuracy is eliminated from the sequence trace.

4.3.2 System Call Refinement Process

In this section, the refinement process of system calls used to discover the accurate system call traces and to achieve better detection performance is described. The massive amount of system calls invoked by the malicious executable during its execution, some of these calls are irrelevant and redundant. The aim of this section is to filter out system calls by discarding irrelevant and redundant features to identify real malicious behavior. This approach reduces the size of the traces to train the algorithms. We will explore effective system refinement approach in more detail in the following subsections.

4.3.2.1 The Problem of noise features

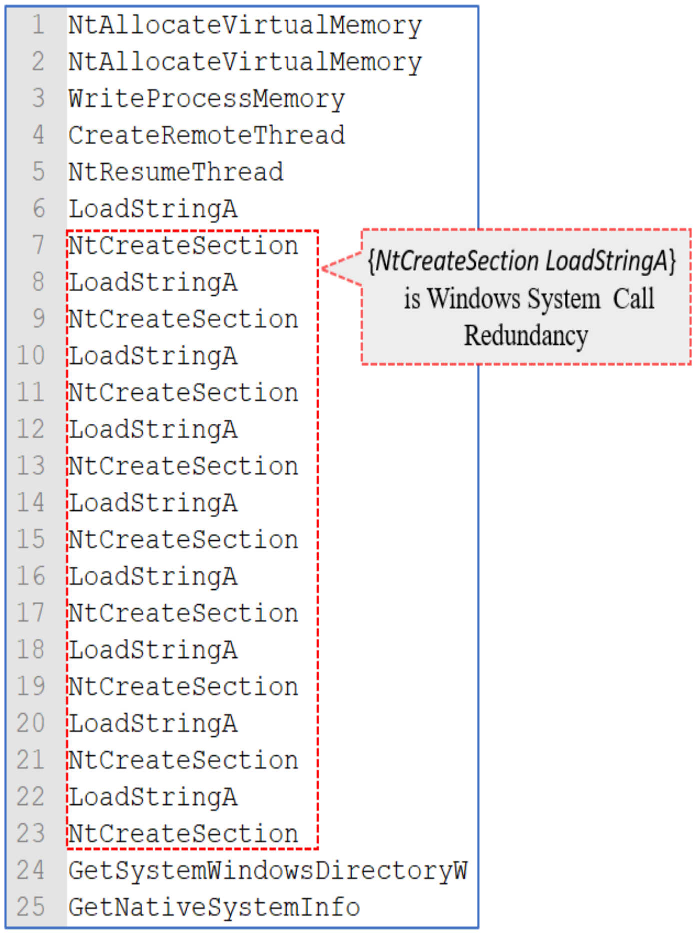
To evade the detection, ransomware writers disguise themselves by inserting a large number of irrelevant and redundant dynamic call sequences to hide its run-time malicious behaviour and to mislead the flow of execution sequences (Anderson et al., 2017). Because of this, the amount of malicious system calls sequences is enormous that produce the high noisy behavioural sequence. We consider undesirable or uninformative system call traces as noise because they don't contribute to the quality of the detection.

Definition 4.2: System call *S* is expressed as a redundant System call ***SR*** if the feature vector $\boldsymbol{f_i}$ and its correlation $\boldsymbol{f_j}$ have a higher value of $SR_{i,j}$ near or equals to 1. That means the two features are considered to be redundant if their values are completely associated and have no contribution to the target. Generally, the squared cosine similarity is employed to measure the correlation of a given two system call vectors (f_i, f_j) as expressed in the following equation:

$$SR_{ij} = \cos^2(f_i, f_j) \tag{4.2}$$

The performance of machine learning depends on the presence or the absence of noisy data. The existence of such noise in the data set could adversely impact the induction of ML models such as the increase in processing time, more storage requirement and the difficult analysis of real malicious intention that can lead overhead and poor prediction ability (Xiao et al., 2015).

To reduce the complexity of the model and to improve the performance of the algorithm, the irrelevant and redundant system calls that do not help in increasing the accuracy of the detection should be eliminated from the tracing records. This refinement approach removes the noisy system call subsequence as shown in Box 4.2 from the original feature space to characterizes the behaviour of the ransomware and its nature.



```
NtAllocateVirtualMemory
NtAllocateVirtualMemory
WriteProcessMemory
CreateRemoteThread
NtResumeThread
LoadStringA
NtCreateSection
LoadStringA
NtCreateSection
LoadStringA
NtCreateSection
LoadStringA
NtCreateSection
LoadStringA
NtCreateSection
LoadStringA
NtCreateSection
LoadStringA
NtCreateSection
LoadStringA
NtCreateSection
LoadStringA
NtCreateSection
GetSystemWindowsDirectoryW
GetNativeSystemInfo
```

{NtCreateSection LoadStringA} is Windows System Call Redundancy

Box 4.2: A snippet of Redundancy API calls

4.3.2.2 Refining Module

The refining module's purpose is to reduce the size of system call traces gathered from the dynamic analysis by removing system calls that do not discover the main behaviour of the ransomware. It is common in practice that, the more time the program takes to execute, the more system calls functions are generated(Vinod & Viswalakshmi, 2018). Therefore, analysing and inspecting all system calls is computationally infeasible because there are too many calls that do not represent the suspicious behaviours of the program. To address this, we filtered the collected program's traces to make the performance of the detection process faster by examining the system calls made by the ransomware through the descriptions provided by the Microsoft's website using a customized python script. To achieve this goal, we construct an accurate system call in the following ways:

First, we exclude system calls that have no strong indication for the behaviour characteristic of ransomware. For example, ***VirtualAlloc*** is a memory management system call used to increase the size of the heap when malloc cannot find enough memory in the present heap. Some system call does not depict a valuable behaviour of the program, they only exist for transferring information between the application and operating system. for example, the ***setTimer*** system call adjusts the timer for the process. Certain system calls are no longer available for current Windows versions, for example, ***NtQueryInformationProcess*** and ***NtQuerySystemInformation*** are used to get information about the process and the system, these functions are present in the older versions of Windows such as Windows 2000 and Windows XP, but altered or unavailable in the later version of Windows. Therefore, these system calls are ignored(Vinod & Viswalakshmi, 2018).

Second, failed system calls do not define the characteristics of the suspicious program, for example, if the program attempts to read a file twice and fails it, and succeeds in reading the file for the third time, the first two ***unsuccessful*** system calls are considered to be identical calls. To avoid storing duplicates of the same system calls, the failed system calls should be removed.

4.3.3 Feature selection

Feature selection is a technique for selecting the most important and relevant features that are suitable for the detection of ransomware models. A feature selection phase constitutes an important module for the classification of ransomware and benign samples. This approach has several advantages such as the reduction of the quantity of sample (less features). On the one hand, this reduction makes it much easier to manage the data, and improve the accuracy of the classification, it also helps the authors to have a better understand of the results provided by the detection model. The feature selection gathers different techniques allowing to select the subset of features set among the whole features, we employed various selection methods and techniques as we demonstrate in the following subsections (Yang & Pedersen, 1997).

4.3.3.1 Term Frequency-Inverse Document Frequency

After extracting features of all ransomware and benign samples, the total number of the extracted integrated features in section 4.4.2 was ***13, 631*** and this number of features is too large for processing and feeding to classification algorithm, therefore, we selected ***3930*** as prominent features as expressed in Figure 4.7. Selecting the most relevant subset features from the original features can improve classifier performance and the accuracy of classification operation; hence, the effective feature set was identified using term weight as the criterion of feature selection.

We applied term frequency-inverse document frequency (TF-IDF) feature selection method for setting the weight to a term based on its inverse document frequency and evaluating how important feature is a document in the collection (Chen, Islam, Haswell, & Bridges, 2019). The purpose of using TF-IDF weighting is to eliminate those features that commonly occur in many vectors while giving more attention to features that are less frequent in the vectors. The formula for the TF-IDF expressed as follows:

$$W_i = TF(\omega_i, d) \times IDF(\omega_i) \tag{4.3}$$

Where W_i is the weighting scheme of word ω_i in document $d \in D$, and $TF(\omega_i, d)$ is the frequency of term of ω_i in document d, and IDF (Inverse Document Frequency) is then defined as:

$$IDF(\omega_i) = log(\frac{|D|}{DF(\omega_i)}) \tag{4.4}$$

Where $DF(\omega_i)$ represents the appearance of $\underline{\boldsymbol{\omega_i}}$ in a document D. After experiments and study of many technical reports, seven feature classes were extracted as shown in Table 4.2 with a brief explanation.

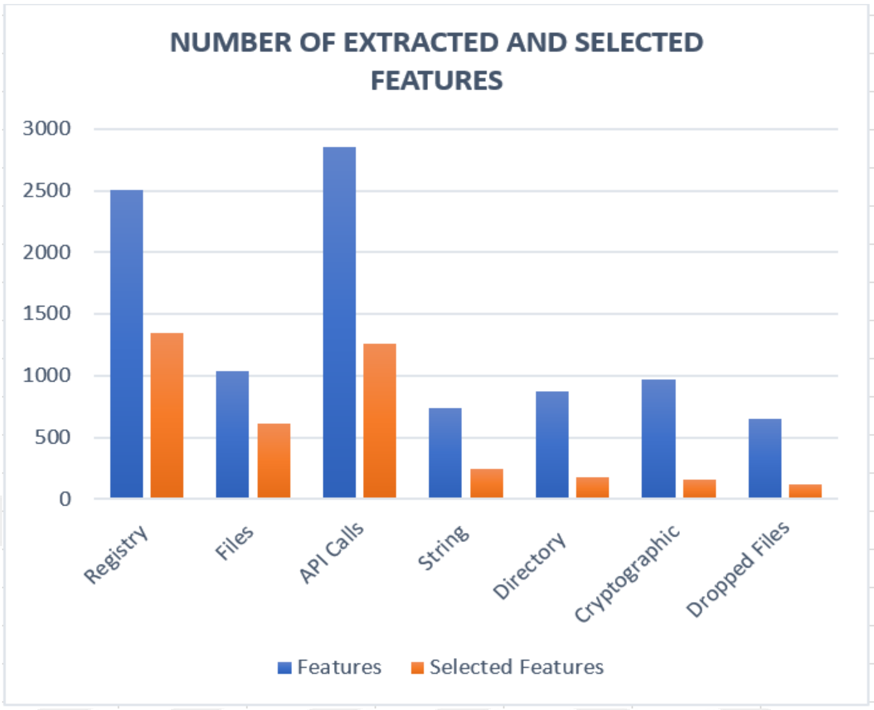


Figure 4.7: Number of extracted and selected features

We observed in our experiments that the highest scores counted by TF-IDF are Registry Keys and API Stats. These are the two most indicative among all other feature classes. Dropped files feature are scaled down due to some normal operations frequently occur in the entire analysed log files.

Table 4.2: Weights of selected feature classes using TF-IDF algorithm

#	Feature Classes	TF-IDF Score	Analysis Type	Percentage
1	Registry Paths	2.682	Dynamic Analysis	21.44%
2	Windows API Calls	2.596	Dynamic Analysis	20.72%
3	Files Operations	1.583	Dynamic Analysis	12.64%
4	Printable String Information	1.452	Static Analysis	11.60%
5	Directory Operation	1.420	Dynamic Analysis	11.36%
6	Cryptographic Libraries	1.391	Dynamic Analysis	11.12%
7	Dropped Files	1.290	Dynamic Analysis	10.32%

4.3.3.2 Enhanced Maximum Relevance and Minimum Redundancy

The sequences extracted by the N-Grams in section 4.4.3.3. include many noise instances that will not uniquely identify the function of malicious files. The automatic identification of these discriminative subset features from the system call logs is a very indispensable task in the analysis phase since it is time consuming and requires many efforts to clean the data. To address this issue, we employed the Maximum-Relevance and Minimum-Redundancy (mRmR) feature selection method proposed by the Peng (Ramí rez - Gallego et al., 2017).

The mRmR is a well-known filter algorithm that intended to find features with high correlation (relevance) with the target class and low correlation (redundancy) to other features. mRMR has been successfully implemented in many different applications including microarray gene expression data analysis, video processing and intrusion detection (Acid et al., 2011). Maximum relevance selects the optimal feature related to the behaviour of a malicious class without considering relationships among features. If S is a set of malicious features, $MI(f_i, C)$ represents the mutual information between the features f_i and the ransomware class, the maximum relevance is calculated by equation 4.5:

$$Max\_Relevance(f_i, C) = \frac{1}{|S|} \sum_{f_i \in S} MI(f_i, C) \tag{4.5}$$

For discrete features, the relevance and redundancy of the feature are measured by calculating the mutual information between the features. Two system call features are considered to be independent if the shared mutual information score is the minimum value of $MI(f, f)$ (Darshan & Jaidhar, 2018). The marginal probability represents $p(x)$ and $p(y)$ whose joint probability distribution is $p(x, y)$. So, the mutual information of feature pairs is defined in equation 4.6:

$$MI(f, f) = \sum_{x,y} p(x, y) log \frac{p(x, y)}{p(x)p(y)} \tag{4.6}$$

Selecting features based on maximum relevance (mR) can identify the actual behaviour relevant to the ransomware. However, mR brings redundancies, since a feature f_i might be highly dependent on some features selected previously (Ramírez - Gallego et al., 2017). To handle the repetition problem in mR, minimum Redundancy (mR) criterion is utilized and defined as:

$$Min\_Redundance(f_i, C) = \frac{1}{|S|^2} \sum_{f_i, f_j \in S} MI(F_i, F_j) \tag{4.7}$$

Where $MI(F_i, F_j)$ is the minimum value of mutual information between feature F_i and feature F_j. The relevance of the system call features to the target **Ω** is maximized and redundancy is minimized by merging these two conditions: $Max\_Relevance(f_i, C)$ and $Min\_Redundance(f_i, C)$ into a single score function denoted by mRMR using mutual information difference ($mRmR\ MID$) defined as below:

$$MID = MAX_S \left[MI(f_i, \Omega) - \frac{1}{|S|} \sum_{f_i, f_j \in S} MI(F_i, F_j) \right] \tag{4.8}$$

Regarding mRmR, subset features $\{f_i \ldots\ldots f_q\} \in F$ are selected as incremental search method by finding the features having a maximum value of mutual information $MI(F_i, F_j)$. These features are ranked based on their relevance to class, and the redundancy of the features are diminished.

Table 4.3: List of notations used in this thesis

Symbol	***Meaning***
$S_t = (P \in S, I)$	The system call trace contains a process P that invokes system call with input I
$\{f_i \ldots\ldots f_q\} \in F$	The subset features are extracted from the feature sets
$C \in \{M, B\}^m$	The target class C that contains malicious M and benign B labels
$f \in R^{mxn}$	The matrix containing m samples and the n features
(f_i, f_j)	The system call vectors
$MI(f_i, C)$	The mutual information of the pair features and the class C
$p(x, y)$	The joint probability distribution of MI
$l \in \{s_1 s_2 s_3 \ldots s_t\}$	The list of the system call sequences
$(f_i, f_j) \in S$	The input of the jth and the ith features in the system call sequences

One limitation of mRmR is the unnecessary computation of the mutual information among pairs of features which gives us the motivation of this Enhanced mRmR method. This is due to the mRmR uses a forward selection technique to find K feature in the features set that maximize the $MID(f_i, C)$ function. The method starts with an empty set, selects the best improvement features, adds to the subset of features $\{f_i \dots\dots f_q\}$, and removes from original feature set F. This process continues until all subset features become equal to the k. In this repetition, the mRmR algorithm calculates the MID value on $(n - k + 1)$ feature at each iteration (Han, Huang, & Qin, 2017).

For example, to decide the $\{f_i \dots\dots f_q\}$, the algorithm computes the value of $MID(f_i, f_j\ )$ on the same feature f frequently, due to the subsequent iterations, the computation complexity of the algorithm becomes high when a large number of noisy features is used. The existing mRMR form is not suitable for ransomware detection because it is computationally expensive due to a large number of system call features generated by n-gram. Therefore, we need a lighter version of mRmR to overcome this difficulty.

To overcome this limitation, we put forward the Enhanced Maximum Relevance and Minimum Redundancy (EmRmR) method to construct an effective system call feature set with a small number of evaluations and less computational complexity. To achieve this, we introduced an associative process based on the assumptions for accumulating the $MI(f_i, f_j\ )$ values on the feature pairs in each iteration to avoid redundant computation. We predefined a maximum threshold k and the items of the list as shown in Equation 4.9.

$$L = \sum_{i=k+1}^{k} MI(f_i, f_j\ ) \tag{4.9}$$

In this work, we preserved the idea of the mRmR algorithm but adapted it to solve the computational problems of the algorithm. The proposed method selects the same subset features as the original mRmR algorithm. In the beginning, the algorithm accepts discretised data set as input $D = (f, C)$, where $f \in R^{mxn}$ is a matrix containing m samples and n features, $C \in \{M, B\}^m$ is a target class consists of malicious M and benign B labels, and the number of features to be selected $\{f_i \dots\dots f_q\}$.

Algorithm 4.3: Enhanced Maximum Relevance and Minimum Redundancy (EmRMR)

Input: Discretised data d, number of features in d is F, subset features $\{f_i \ldots\ldots f_q\} \in F$,class C, number of features to select q, S_f selected features

Output: selected output features F

```
    Initialization
1   $S_f \leftarrow 0$;
2   for $f_i \in F$ do
3           Relevance ($S$) ← Mutual_Info($f_i, C$);
4           Aggregate_ Redun =0;
5   end
    for
6   $S_f$=Max (Relevance ($S$))
7   $l \in \{s_1 s_2 s_3 \ldots\ldots s_t\} \leftarrow [f_i \in F \mid S]$//To store the highest scorer
8   for t1:  q-1 do
9   size ← $len$ ($l \in \{s_1 s_2 s_3 \ldots\ldots s_t\}$)
10  while $v$ ++ < size do
11          for $f_j \in F$ do
12            $Rel_f \leftarrow$ Relevance ($S$)
13            Aggregate_ Redun =$\left(k + 1 + MI\left(f_i, f_j + 1\right)\right)$
14          end for
15          $R_f \leftarrow Rel_f$ -  Aggregate_ Redun;
16  end for
17  return selected feature subset $\{f_i \ldots\ldots f_q\}$;
```

The function Mutual-info(f_i, C) in the algorithm calculates the relevance of features to the target class as Equation 4.7, the feature with the highest score is extracted and stored as $f_i \in F$ in S variable.

To avoid the redundant computations of the MI, we then created an empty list sequence to store the sum of the mutual information output features and use it as a reference for the next step. A loop is then performed for the remaining features and the mutual information between the selected feature and the unselected features is computed again as Equation 6. Finally, the algorithm creates R_f variable to store features with the maximum value of relevance and minimum redundancy.

4.3.3.3 Feature selection using FastICA

The extracted features are then applied to the Algorithm 4.4 to generate the global set of features and the generation of the feature vector. The global set of features is generated by combining all the features of the data set. In our data set, we have accumulated a total 15972 features in the global set. In the feature vector, the columns represent the features and the rows represent the ransomware or benign ware. The values of this vector will be {0,1} where {0} represents the absence of the feature and {1} represents the presence of the feature. Since the global set contains 15972 features and the data set contains 1237 samples, the two-dimensional matrix will contain 1237 rows and 15972 columns.

The challenging task is to handle such a large number of features. Since ICA (Independent Component Analysis), a statistical procedure to solve the Blind Source Separation can be used in the selection process where the components are statistically independent. FastICA (Hyvärinen & Oja, 2000) is considered as the feature compression method in our model. The high computation complexity of FastICA did not affect our system as we have used the GPU based operation where we have used 8 parallel GPUs. It uses a fixed-point iteration scheme that has been found in independent experiments to be 10-100 times faster than conventional gradient descent methods for ICA. Another advantage of the FastICA algorithm is that it can be used to perform projection pursuit as well, thus providing a general-purpose data analysis method that can be used both in an exploratory fashion and for estimation of independent components. We have selected 40,50, 80 and 100 features using FastICA. These selected features are then applied to the model.

The performance of deep learning mostly depends on the size of mini batch, initial settings of weight, number of epochs, learning rate, momentum and number of hidden layers and units. The different architectures of nodes in hidden layers also affect the performance of the deep learning-based model (Zhang et al., 2019). In the first phase of our experiment, we have varied the epoch number.

Algorithm 4.4: Global Feature Set Generation and Feature Vector Generation

```
input ← $Samples\ in\ Data\ Set$
Output  $Global\ Feature\ Set, Feature\ Vector$

1.  Begin
2.      $Global\ Feature\ Set \leftarrow \emptyset$
3.  for each $feature$ in each sample in input dataset do
4.      if $feature$ not in $Global\ Feature\ Set$ then
5.      $Global\ Feature\ Set \leftarrow feature$
6.  end for
7.  $Feature\ vector \leftarrow \emptyset$
8.  for each sample in input data set do
9.      $Row \leftarrow \emptyset$
10.     $Row['is\_Ransomware'] \leftarrow \{0\ for\ benign, 1\ for\ Ransomware\}$
11. If $Feature$ is present in the sample
12.     $Row[Feature] \leftarrow 1$
13. else
14.     $Row[Feature] \leftarrow 0$
15.     Append $Row$ to $Feature\ vector$
16. end for
17. return $Global\ Feature\ set, Feature\ vector$
18. END
```

We have varied the epoch number from 50 to 500. In the result section, we have mentioned only the performance result using 500 epochs because the 500 epochs show the best performance. Then, we have developed the three different node arrangements-Architecture1, Architecture2 and Architecture3. The following Table 4.3 shows the nodal arrangements that are considered in our model.

TABLE 4.4: ARCHITECTURE1, ARCHITECTURE2 AND ARCHITECTURE3

Architecture Name and Definition	Combinations considered in our model
Architecture 1 with equal nodes in L1 and L2	32 nodes in L1 and L2, represented as L32_L32
	64 nodes in L1 and L2, represented as L64_L64
	128 nodes in L1 and L2, represented as L128_L128
	512 nodes in L1 and L2, represented as L512_L512
	1024 nodes in L1 and L2, represented as L1024_L1024
Architecture 2 with less nodes in L1 and more nodes in L2	32 nodes in L1 and 64 nodes in L2, represented as L32_L64
	64 nodes in L1 and 128 nodes in L2, represented as L64_L128
	128 nodes in L1 and 512 nodes in L2, represented as L128_L512
	512 nodes in L1 and 1024 nodes in L2, represented as L512_L1024
Architecture 3 with more nodes in L1 and less nodes in L2	64 nodes in L1 and 32 nodes in L2, represented as L64_L32
	128 nodes in L1 and 64 nodes in L2, represented as L128_L64
	512 nodes in L1 and 128 nodes in L2, represented as L512_L128
	1024 nodes in L1 and 512 nodes in L2, represented as L1024_L512

4.4 Summary

The main purpose of this section is to create appropriate input for the classification algorithms and to describe the steps involving data pre-processing. This process consists several sections; the first section explained the portable executable file format. In this section sample analysis process to identify the run time behaviour of the ransomware is discussed. This process mainly focused on the dynamic analysis of a binary that executes one or more times to observe the ransomware behaviour. The environment in which the binary is executed is in a controlled environment that guarantee its containment. To extract the set of features such as Register, file operations, Windows API function calls, and file paths from the PE executables feature extraction process are discussed in the third section.

Data pre-processing that contains the identification of executable files and refinement model. The massive amount of API calls invoked by the malicious executable during its execution, some of these calls are irrelevant and redundant. To filter out API calls, we discarded the irrelevant and redundant features to identify the real malicious behaviours. Feature reduction method is also discussed, the majority of feature is not contributing the accuracy and the speed of the classification algorithms, therefore reducing the less important feature would improve the efficiency of the algorithms. Several feature selections were used to select the informative feature that would be input to train the machine learning algorithms.

5. RESULTS AND DISCUSSIONS

This section describes the experimental results obtained from the previous section. We describe the behavioural detection methods and present their accuracy based on the performance metrics aforementioned in section 3. The performance of the classifier depends on how the model was built up and how the parameter values assigned, so the first section of this section illustrates the process of setting the parameters. We select various parameters and test our models individually to select the best model among them. The Second section, we highlight the experimental results of the research, so, we divide our result into three different detection models and we demonstrate the performance of each result as tabular forms. The first experiment, we performed an automated dynamic behavioural analysis of real-world ransomware samples that infect Windows platforms, to distinguish these malicious files from benign files we utilized supervised machine learning algorithms like Support vector machine (SVM) and Artificial Neural Network (ANN).

The dimensionality reduction of the Windows System Calls features for the ransomware detection are also presented in the second experiment. Naturally, Windows API calls are suffering a massive amount of irrelevant and redundant system calls invoked by the malicious executables during its execution. Therefore, in this experiment, we introduce a refinement process to reduce the size of the system call traces and to filter the Windows API calls. Finally, the third experiment discusses a model framework based on the runtime behaviour of ransomware and deep learning based semi-supervised technique. Deep learning is a robust unsupervised approach that can extract the hidden intrinsic patterns from unsupervised feature space through a non-linear transformation and layered structure in which upper layers compute more abstract forms of features presenting the latent sources of variabilities in the feature space. Finally, we compared the proposed methods against previous similar works.

5.1 Setting Experimental Parameters

Parameters selection values for the machine learning algorithms have an extreme effect on the model's performance and the generalization error. There is no certain method and technique to set on the optimal parameter values and ranges, only to do an exhaustive repetitive search over the parameter space to find the best setting. We described the various parameters and tested our models individually to select the best model among them. In this study, SVM kernel functions such as the linear kernel, polynomial kernel and Radial basis function (RBF) are used as shown in Table 5.1. The parameters values for kernel functions can have an extreme effect of the model's performance and the generalization error(Takeuchi et al., 2018)

Table 5.1 Selected parameters value of SVM kernels

Kernel functions	**PARAMETER VALUES**					
	Value	***Complexity parameters C***				***Build logistic model***
		1.0	10	100	1000	
Linear kernel	(λ=1)	✓	✓	✓	✓	Assigned True
Polynomial kernel	(λ=2)	✓	✓	✓	✓	Assigned True
	(λ=3)	✓	✓	✓	✓	
	(λ=4)	✓	✓	✓	✓	
Radial Base Function kernel	(γ= 0.01)	✓	✓	✓	✓	Assigned to True
	(γ= 0.125)	✓	✓	✓	✓	
	(γ= 0.25)	✓	✓	✓	✓	
	(γ= 0.5)	✓	✓	✓	✓	
	(γ= 1.0)	✓	✓	✓	✓	

Therefore, we set the parameters of these kernel functions with the incremental regularization parameter λ and the cost parameter of C. These selection parameters require an exhaustive repetitive search over the parameter space to find the best settings. The above Table 5.1 illustrates the kernel functions used and values of their subsequent parameters. The parameter C regulates the trade-off between the complexity of the model and the empirical risk of the model. There are two values lambda and gamma both has a decimal and floating-point with consideration of the complexity control starting 1.0 until 1000 to reduce overfitting. For proper probability estimation, the BuildLogisticModel was set to true for all kernel functions.

A user-defined neural network parameter such as hidden layers, momentum and learning rate was assigned values to test the network performance. The learning rate parameter is the specified user value that controls the step size when weights are iteratively adjusted. The momentum parameter is helpful to prevent the algorithm from converging to a local minimum (Dietterich & Kong, 1995). Setting a high momentum parameter value can facilitate to increase the speed of convergence of the algorithm. However, selecting too high momentum parameters can generate a risk of overshooting the minimum, which can cause the algorithm to become unstable. Hence, in this study, four values of learning rate and momentum parameters are picked as: 0.1, 0.3, 0.6 and 0.9 respectively as shown in Figure 5.2. Finally, we created many different models with a variety of outputs.

Table 5.2: Selected ANN user defined parameter values

Parameters	**Values**	**Auto build**	**Graphical user interface**
Epochs	100,500	True	True
Hidden layers	1,2,3	True	True
Number of neurons in hidden layer	15,30,60	True	True
Momentum	0.1,0.3,0.6,0.9	True	True
Learning rate	0.1,0.3,0.6,0.9	True	True

On the other hand, the Deep Learning approach has the benefit of training the model using the extracted and selected features and behavioural patterns through hidden nodes in different layers. Since the cyber-attack patterns have been changed very frequently, the inherent cyber-attack patterns can be extracted using the multiple layers of abstraction of Deep Learning and represent the actual attack patterns to a non-linear and higher abstraction of the real scenarios which benefits the detection model (LeCun et al., 2015).

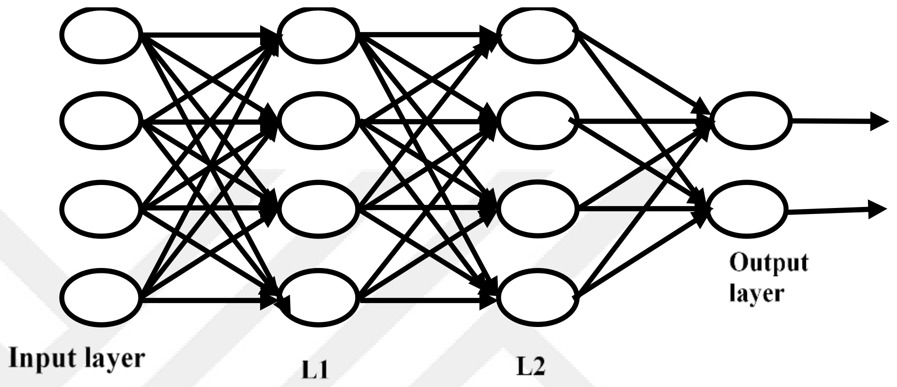


Figure 5.1: Architecture 1 with equal nodes in L1 and L2

Since deep learning-based model handles multiple hidden layers, we need to decide how many hidden layers will be suitable for our model. Too many hidden layers will cost us more computational complexity, we have chosen two hidden layers. The nodes of the hidden layer can vary and form different architectures of nodes. As we have considered only two layer- we have named it as L1(first layer) and L2 (second layer). The following Figures show the adapted architectures of our model.

Three different architectures have implemented in our modelArchitecture1 in Figure 5.1, Architecture2 and Architecture3. Architecture1 has equal number of nodes in L1 and L2. Architecture2 has a smaller number of nodes in L1 and more nodes in L2 where as Architecture3 has more nodes in L1 and less nodes in L2 as shown in Figure 5.2.

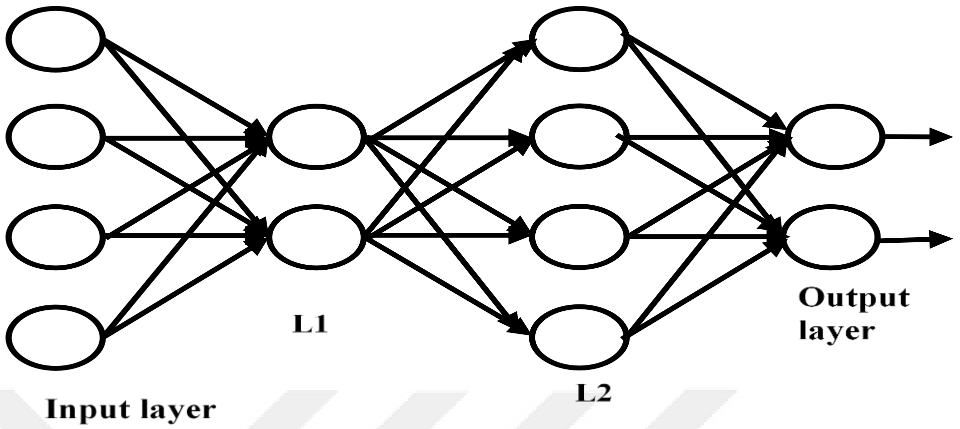


Figure 5.2: Architecture3 with more nodes in L1 and less nodes in L2

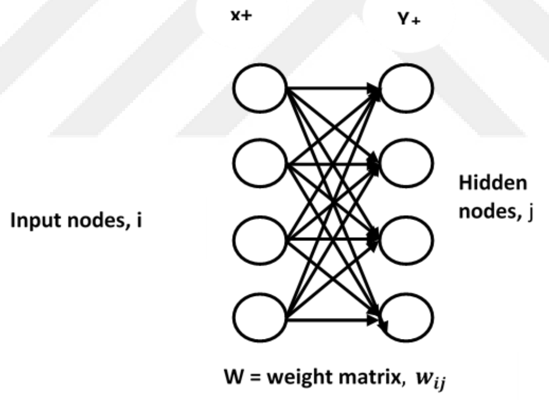


Figure 5.3: RBM layers with input and hidden layer nodes

In Figure 5.3, the input layer is denoted as "i", hidden layer as "j", x is the bias of input nodes as shown in Figure 5.3, y is the bias of hidden nodes and w is the weight then the structure of input $(k)$and hidden nodes(l) holds the energy as follows:

$$ERG(k,l) = -\sum_i x_i\, k_i - \sum_j y_j l_j - \sum_{ij} k_i l_j w_{ij} \tag{5.1}$$

The nodes will generate 0 or 1 using the following equations

- For input nodes, $S(k_{i=1}|l) = \sigma(x_i + \sum_j w_{ij} l_j)$ (5.2)
- For hidden nodes, $S\left(l_{j=1}\middle|k\right) = \sigma(y_i + \sum_i w_{ij} k_i)$ (5.3)

In these equations, $\sigma(t) = 1/(1 + e^{-t})$ where $t = x_i + \sum_j w_{ij} l_j$ and $t = y_i + \sum_i w_{ij} k_i$ respectively. Equation 5.4 is followed in tanning section to maintain the stochastic ascent algorithm.

$$S(k) = (1/t) \sum_j e^{-ERG(k,l)} \tag{5.4}$$

Let, the learning rate is μ. μ should be more than zero. The change of weight matrix is accumulated by the following equation where the expectation of data is denoted by β and expectation of reconstruction is denoted by γ.

$$\Delta w_{ij} = \mu(\beta - \gamma) \tag{5.5}$$

5.2 Experimental Results

In this section, we will present the results obtained from the experiments of this research. We carried out three main different experiments including supervised and semi-supervised machine learning algorithms. In every category we will highlight and explain the results gathered from that experiment.

5.2.1 Experiment one

In this experiment we employed a behavioural malware detection framework mentioned in section 3, subsection 3.2.1 in Figure 3.2 for ransomware using two supervised machine-learning approaches like Support Vector Machine (SVM) and Artificial Neural Network (ANN) algorithms. We performed an automated dynamic behavioural analysis for 673 real-world ransomware samples that infect Windows platforms. We focused on the malicious behaviours of 14 newly emerged ransomware families. In the following subsection, three different experiments are conducted to train and test the classifiers.

5.2.1.1 Train-test splitting method

The purpose of this experiment is to evaluate the performance of the proposed integrated features by employing a train-test split method, which is dividing the whole data set into two subsets: training and testing data. first, we split our dataset randomly with a uniform distribution of 80: 20% ratio as training and testing respectively. The experimental results of ANN showed an accuracy of 0.958 with 0.101 false positive rates while SVM presented higher false positive of 0.109 compared to ANN and the accuracy of 0.932.

Table 5.3: Result of the train-test splitting method

	FP Rate	TP Rate	Precision	Recall	AUC	Detection Rate
SVM	0.109	0.853	0.923	0.926	0.904	0.932
MLP	0.101	0.956	0.945	0.951	0.965	0.958

The ROC curve of this experiment is presented in Figure 5.4, and the Table 5.3 shows the results of the FPR, TPR, AUC, precisions and the recalls and the accuracy of the classifier based on the training and testing splitting method.

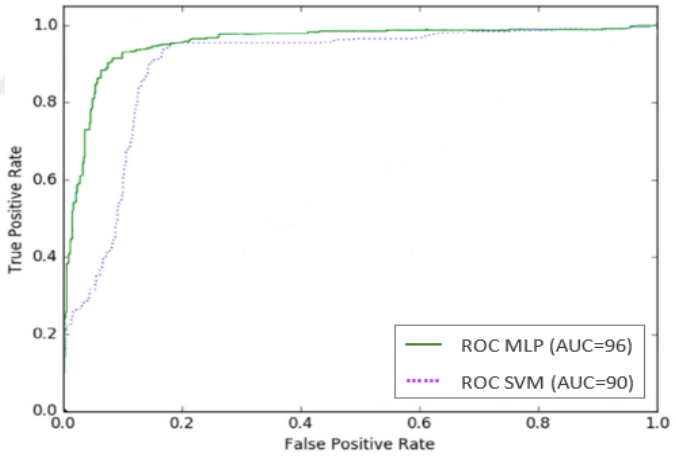


Figure 5.4: ROC curve of the classifiers on train-test splitting method

5.2.1.2 Cross-validation method

In the train-test splitting method, once the data set is divided into a ratio that does not relevant each class of the experimental samples, the result of the holdout error rate will be inaccurate. To overcome this limitation, we applied the 10-Fold cross-validation technique to prevent the overfitting problem and to estimate the effectiveness of our models. In this approach, the entire data set was randomly shuffled and divided into 10 equal-sized of subsets such that, each repetition (10-fold) we build our model with 10-1 folds of the data set for the evaluation of the trained model and the remaining one-fold constitutes for testing.

In this experiment, we have evaluated the performance of classifiers using 10-fold cross-validation to train and test the algorithms. The results achieved by the classifiers in this experiment on the whole dataset were quite satisfactory. The best accuracy reached SVM by presenting 0.982 of AUC with less than 0.035 of false positive rate. It is important to examine the ability of the classifiers for distinguishing the ransomware from benign samples, therefore, precision and recall are applied to both datasets and presents 0.945 and 0.942 respectively. SVM also shows a fairly better accuracy of 0.952 comparing to MLP that shows 0.945 of detection rate and 0.036 of the false positive rates as presented in Figure 5.5 and Table 5.4. This indicates that SVM has super generalization ability and is quite tolerant for training the iteration of 10-fold set size.

Table 5.4: Result of the 10-fold cross validation method

	FP Rate	**TP Rate**	**Precision**	**Recall**	**AUC**	**Detection Rate**
SVM	0.035	0.962	0.945	0.942	0.982	0.952
MLP	0.036	0.982	0.931	0.932	0.971	0.945

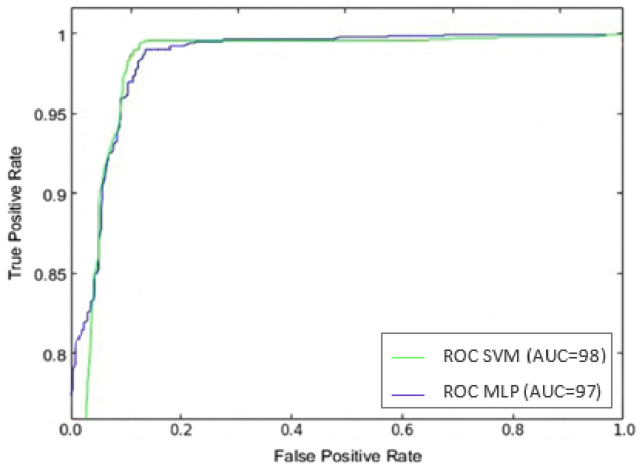


Figure 5.5: ROC curve of the classifiers on the 10-Fold validation method

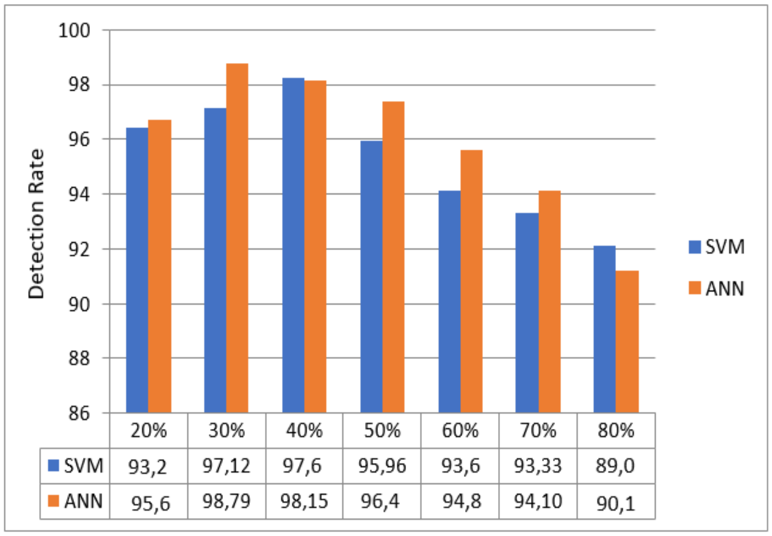
5.2.1.3 Testing with selected subset features

The aim of this experiment is to evaluate how selected subset features can effectively contribute to the performance of the algorithms. The selection of subset features eliminates redundant and irrelevant features and reduces the dimensionality of the dataset. In this experiment, we divide our features into seven subset features by considering their importance and ranking based on the aforementioned feature selection algorithm presented in section 4, subsection 4.6.1. We created the most prominent features as Top-N feature set: top20, top30, top40, top50, top60, top70, and top80.

Table 5.5: FPR, TPR, AUC and accuracy for SVM and ANN with subset features

	Support Vector Machine				**Artificial Neural Network**			
	FP Rate	*TP Rate*	*AUC*	*Det. Rate*	*FP Rate*	*TP Rate*	*AUC*	*Det. Rate*
20	0.371	0.625	0.948	0.932	0.033	0.952	0.972	0.956
30	0.041	0.959	0.976	0.971	0.007	0.988	0.986	0.987
40	0.006	0.993	0.977	0.976	0.012	0.987	0.982	0.981
50	0.071	0.935	0.974	0.959	0.035	0.962	0.985	0.964
60	0.160	0.839	0.951	0.936	0.035	0.964	0.978	0.948
70	0.041	0.958	0.973	0.933	0.034	0.951	0.980	0.941
80	0.103	0.837	0.238	0.891	0.036	0.841	0.186	0.901

The experimental results demonstrated that ANN showed the highest accuracy of 98.79% when top30 of the feature set was used as training and testing. However, this classification accuracy had dramatically decreased to 95.63% when top20 of the feature set was used. The other hands, the best model of SVM presented an accuracy of 97.6% when top40 of the feature was applied for training the model. Although SVM performed ratio of 0.993 of TPR and 0.0371 of FPR, this indicates that SVM has a higher ratio of false positive rate compared to ANN.



	20%	30%	40%	50%	60%	70%	80%
🟦 SVM	93,2	97,12	97,6	95,96	93,6	93,33	89,0
🟧 ANN	95,6	98,79	98,15	96,4	94,8	94,10	90,1

Figure 5.6: Comparison of SVM and ANN classification accuracy with subset features

The experimental result implies the importance of considering the selection of different subset features. Figure 5.6 compares SVM and ANN classification accuracy across different subset features. By inspecting Figure.5.6, both ANN and SVM had low classification accuracy when top80 of the feature set used to train and test the model. This indicates that more features do not improve the performance of the classifiers as Table 5.5 shows the results of each classifier based on selected features.

5.2.2 Experiment two

In this section, we present the results obtained from various extensive experiments. To determine the best model for the detection of ransomware, we tested the performance of five supervised algorithms such as Decision Tree (DT), K-Nearest Neighbour (kNN), Logistic Regression (LR), Random Forest (RF) and Support Vector Machine (SVM) on the system call sequences made by the malicious and benign samples. The aforementioned performance metrics are used to evaluate the models. The proposed architecture of the dynamic characteristics of behaviour-based ransomware detection is presented in section 3, subsection 3.2.2 in Figure 3.3. In the following subsection, we describe the three different experiments conducted to train and test the classifiers.

5.2.2.1 Windows System Calls with N-gram features

In this experiment, we evaluated the performance of the classifiers on $n-grams$ of length 2,3,4 and 5 from each system call sequence by employing the train-test split method, which divides the data-set into two subsets: training and testing data. Before building the model, we randomly splitted the dataset with a uniform distribution of 80: 20% ratio as training and testing respectively. The accuracy of the classifiers and the values of the Area under Curve (AUC) were used as evaluation metrics in this experiment.

Table 5.6: The accuracy of the Tran-test splitting method

	KNN	LR	SVM	RF	DT
2-grams	0.584	0.723	0.663	0.678	0.636
3-grams	0.958	0.924	0.984	0.782	0.981
4-grams	0.751	0.865	0.899	0.613	0.815
5-grams	0.729	0.783	0.887	0.592	0.751

Table 5.6 and the Figure 5.7 compares the accuracy of each classifier trained and tested with various $n-gram$ sequences. The SVM with 3-grams achieved the highest accuracy among all classifiers, ranging from 66% to 98% with less 0.0261 false-positive rate. The accuracy of this classifier remained stable when 4-grams and 5-grams of feature sequence are employed. From the experimental results of the Table, the accuracy of kNN and DT algorithms are nearly similar. These classifiers with 2-grams of features achieved a lower accuracy of 58% and 63% respectively, while the accuracy increased significantly when 3-grams of bytes were used.

The RF classifier produced the lowest accuracy among the classifiers when the model was trained and tested using different lengths of $n-$ grrams. More specifically, the classifier showed fairly good accuracy of 67% and 78% with 2-grams and 3-grams sequences respectively. However, the classifier started a low accuracy of 61% and 59% when 4-grams and 5-grams of system call sequences are trained and tested to the classifier, this poor performance is probably due to the high computational complexity that the RF

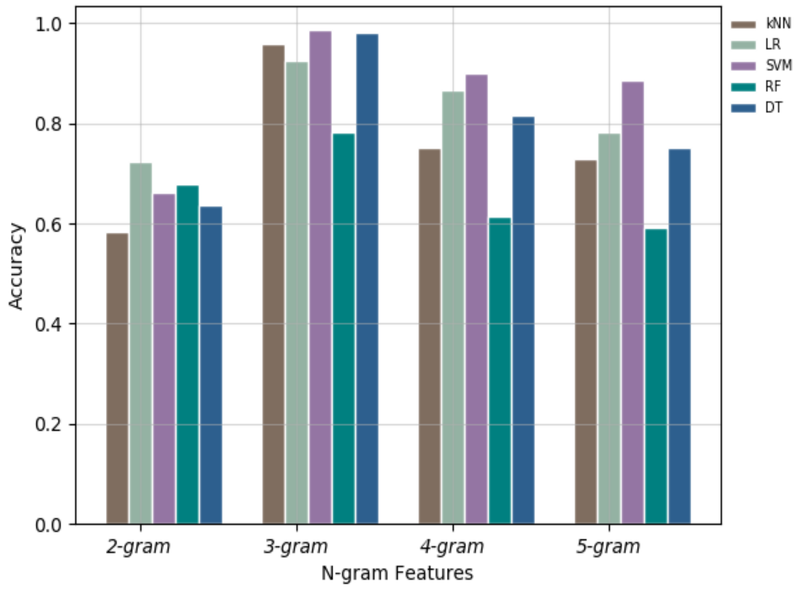


Figure 5.7*:* The classifier's accuracy on n-gram with train-test splitting method

requires to build the model as we enlarge the size of the n-grams, it also suffers overfitting problem during the training that the model is unable to generalize the new features in the testing phase. Interestingly, the LR significantly outperformed the other classifiers and showed 0.723 of accuracy, it also presented less FPR of 0.413 when$2-gram$ features were trained in the model. This is due to the classifier doesn't involve high computational power when the model is trained with a smaller size of n-grams.

Figure 5.8 shows the trends for the ROC curve of the classifiers with 2-gram, 3-gram, 4-gram, and 5-gram. Overall, the AUC values decreased significantly when 2-gram-based features were used to train the model. This is due to the small length of $n-grams$ that can result in a poor characterization of the ransomware behaviour, which leads to the increase the false positive alarms. We expanded the feature dimensionality by increasing the size of the window to 3-gram.

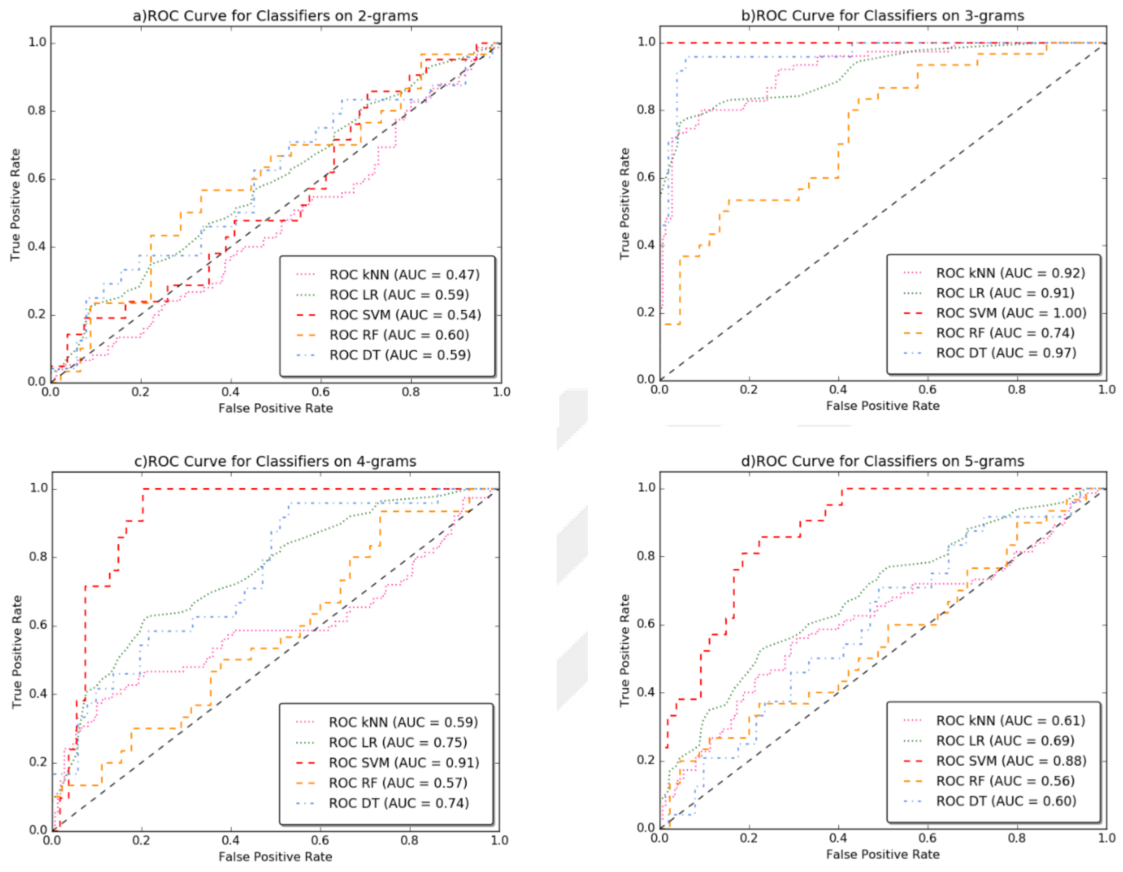


Figure 5.8. ROC Curve of the Classifiers on N-grams with train-test splitting method

The AUC values of the classifiers also increased dramatically. From Figure 5.8 (c), it can be seen that the AUC declined slightly at an average of 0.16% for $n = 4$. Moreover, Figure 5.8 (d) compares the ROC curves of longer sequences for $n = 5$ on the classifiers. The result shows a considerable drop in AUC values. One possible reason is that a long sequence of the $n - grams$ pattern is not useful for distinguishing the behavior of the ransomware from the benign. In addition, increasing the length of N-grams sequences also increases the training time of the model.

5.2.2.2 Windows System Calls on N-grams with cross-validation method

The purpose of this experiment is to evaluate the performance of the classifiers on various sizes of n-gram features with $10 - Fold$ cross-validation technique. There is a limitation in the train-test splitting method, once the dataset is divided into a ratio that does not relevant each class of the experimental samples, the result of the holdout error rate will be inaccurate. To overcome this problem, we applied the $10 - Fold$ cross-validation technique to prevent the overfitting problem and to estimate the effectiveness of our models. In this approach, the entire data set was randomly shuffled and divided into 10 equal-sized subsets such that, each repetition (10-fold) we build our model with 10-1 folds of the data set for the evaluation of the trained model and the remaining one-fold constitutes for testing.

We tested the accuracy and the Roc curve of every sliced window of different feature lengths. The results of this experiment show an accuracy variation among the classifiers (accuracy 52-93%) when 2-grams, 3-grams, 4-grams and 5-grams length with 10-fold cross-validation is used to train and test the algorithms as shown in Table 5.7. These results were quite satisfactory compared to the previous experiment.

Table 5.7: The accuracy of N-gram with the 10-fold cross-validation

	KNN	LR	SVM	RF	DT
2-grams	0.526	0.782	0.752	0.536	0.829
3-grams	0.769	0.827	0.912	0.751	0.842
4-grams	0.746	0.812	0.923	0.572	0.782
5-grams	0.582	0.683	0.632	0.652	0.752

Figure 5.9 shows the classifier's accuracy on n-grams of length 2,3,4 and 5 from each system call sequence with the 10-fold cross-validation technique. The best accuracy among all classifiers was achieved by the SVM with 0.899 of AUC. It also shows the accuracy of 0.923 when $n = 4$ with less than 0.124 of false-positive rate. This indicates that SVM has super generalization ability on a large windows size, and is quite tolerant for training the iteration of 10-fold set size. It can also be observed that kNN performed a low accuracy of 52% and 63% on both $n = 2$ and $n = 5$ respectively. However, the classifier shows a fairly better accuracy on $= 3$. The detailed results of the accuracy of each classifier based on various windows size is also presented in Table 3 and Figure 5.10.

Figure 5.9 illustrates the AUC variations for the five classifiers on the different $n-$ grams. Three classifiers (LR, SVM, and DT) have a high AUC rate while the other two classifiers (kNN and RF) have a slightly lower AUC rate when features are sliced into $3-grams$ and $4-grams$. By inspecting Figure 6 (d), all classifiers had a significantly low AUC rate when $5-grams$ of the feature set used to train and test the model. This is due to the number of n-gram sequences that increases whenever the window size increases.

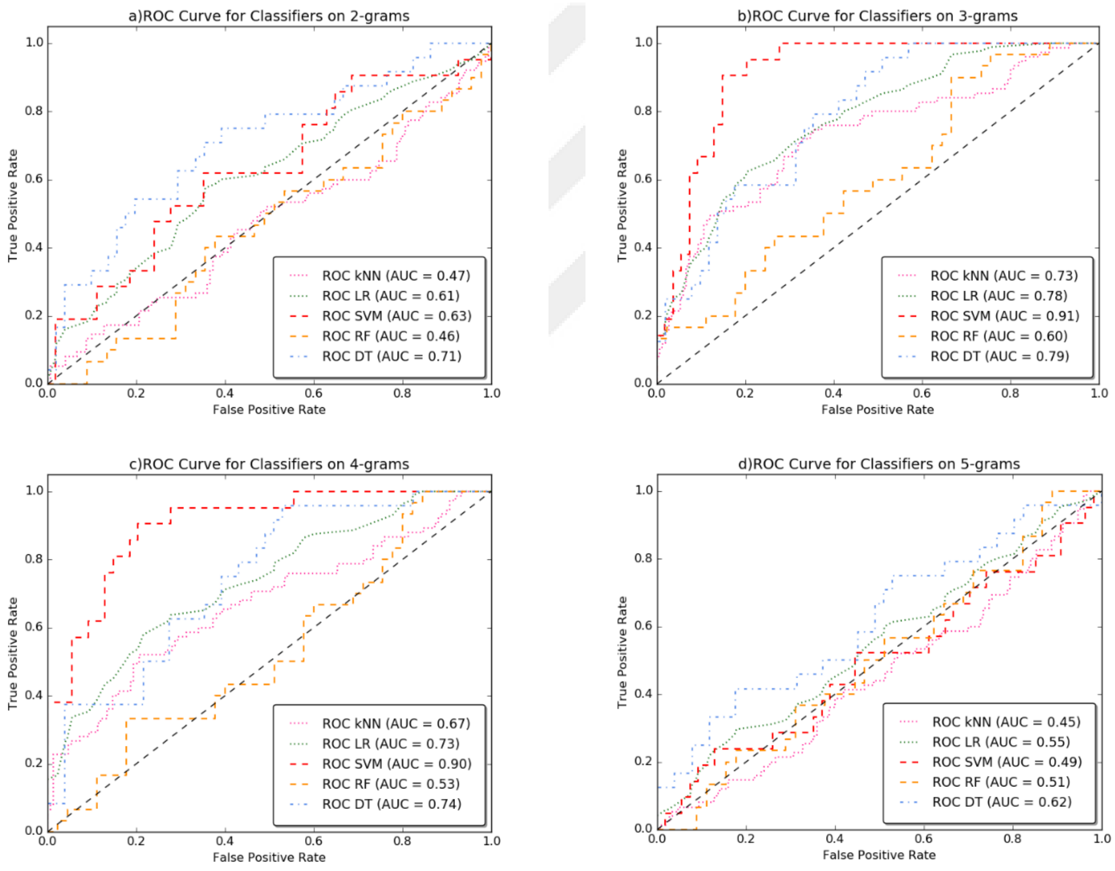


Figure 5.9: ROC Curve of the Classifiers on N-grams with 10-fold cross-validation

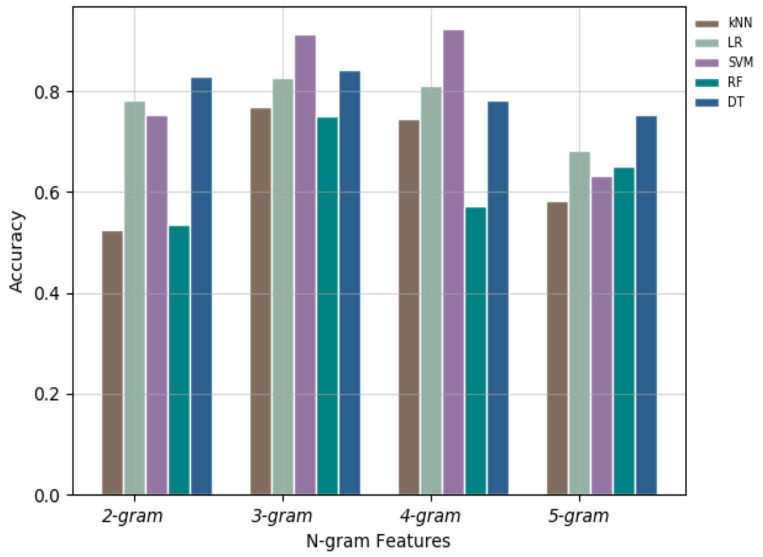


Figure 5.10: The classifier's accuracy on N-gram with 10-fold cross validation

5.2.2.3 Windows System Calls on k-features

The aim of this experiment is to evaluate how the sub selected k features generated by the Enhanced mRmR in section, subsection 4.6.2 can effectively contribute to the performance of the algorithms. In this experiment, we divided the features into k based features by considering their importance and ranking determined by the Enhanced mRmR. Similar to the previous section 5.3.2.2, the $10 - Fold$ cross-validation technique was employed to train and test the model. We evaluated five algorithms with the most prominent features indicated as the Top-k feature set: 30, 60, 90 and 120, and their detection accuracies were recorded and compared.

Table 5.8: The accuracy of the classifiers with k feature dimensions

	KNN	LR	SVM	RF	DT
30	0.567	0.766	0.853	0.674	0.657
60	64.47	0.887	0.853	0.724	0.949
90	0.782	0.801	0.974	0.576	0.805
120	0.701	0.779	0.842	0.589	0.783

Table 5.8 shows the detection accuracy of the classifiers with selected subset k features. In this experiment, the highest accuracy of 97.4% achieved by the SVM with low false-positive rate of 0.016 when the top90 k of the feature set was used. However, this detection accuracy had dramatically decreased to 85.3% when top30 and top60 of the feature set were employed. From Table 5.8, it is obvious that the kNN classifier exhibits the lowest detection performance of 56.7%, 64.4% and 78.2% at all experimented k features.

Table 5.9: Precision, Recall, F1-measure and FPR with k feature dimensions

	KNN	LR	SVM	RF	DT
Precision	0.308	0.751	0.986	0.601	0.801
Recall	0.328	0.761	0.994	0.555	0.816
F1-measure	0.358	0.749	0.986	0.554	0.807
FPR	0.801	0.284	0.016	0.399	0.216

As presented in the Table 5.8, the accuracy of all classifiers is slightly similar when k=30. Regarding the classifier's accuracy, the DT outperforms the other approaches by presenting a 94.9% detection rate when the top60 features were applied. Furthermore, when we increase the size of the k features to the 120, the accuracy of the classifiers improved compared to the lower features. On the other hand, the experimental results shown in Figure 5.11 reveals that the AUC values of the top30 features decreased as there are behaviors that have a high frequency in both malicious and benign logs. This indicates that fewer features do not characterize the real behavior of the ransomware as shown in Figure 5.11(a). Moreover, the ROC curve achieved the highest AUC rate of 98% as we increase the size of the k to the top90 features. Table 5.9 illustrates the Precision, Recall, F1-measure and FPR for the classifiers on the top90 k features. This Table clearly shows the SVM and DT achieved better performance not only for the precision and recall but also the F1-measure as well.

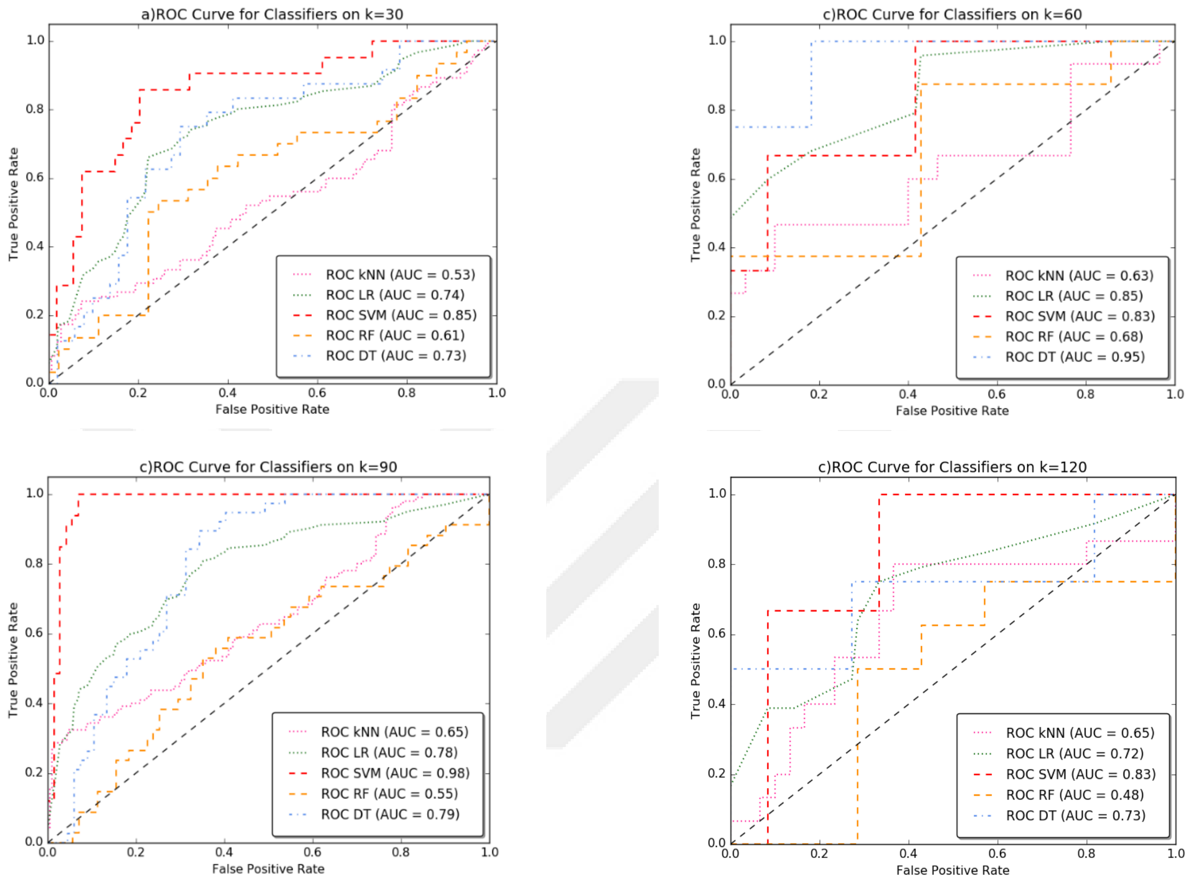


Figure 5.11: ROC Curve of the Classifiers with different on k features

5.2.3 Experiment Three

In Experiment one, the classifiers presented low accuracy of 0.901 when the whole dataset is used, this is due to some integrated features that are common both available in ransomware and benign executables. In addition, we used a large dataset, containing several major ransomware families and several hundreds of irrelevant and noisy features generated by the ransomware to hide its malicious behaviour. Unlike, the experiment one, in this section we applied system calls as they are effective for distinguishing between the behaviour of malicious and benign programs. We also implemented an automated deep-learning-based ransomware detection engine that can automatically detect whether an executable is a ransomware or not. With thousands of ransomware executables, we thoroughly test our model and perform an in-depth analysis on the features that deep learning essentially exploits to characterize the ransomware. So, deep learning models has the ability to perform automatic feature extraction from raw data, also called feature learning.

This section will represent the algorithm of our detection model presented in section 3, subsection 3.2.3. Here, we have considered the different arrangement of nodes described in section 4, subsection 4.6.3 on Table 4.4. We have varied the epoch number and do the train and test using 10-fold cross validation. The detection model is trained and tested using both global features and selected features obtained from Algorithm 5.1.

Algorithm 5.1: Varying the node arrangements using Deep Learning based Model

```
        $input \leftarrow Data(F_1 F_2, F_3 \dots \dots \dots \dots F_m)$ {data set with m number of global features}
        Output $Accuracy_1, Accuracy_2$
1.      Begin
2.      Set $arrangements \leftarrow$
        $(arrangement_1 arrangement_2, \dots \dots \dots \dots arrangement_n)$
3.      for each $arrangement$ in $arrangements$ do
4.          for $fold = 1$ to 10 do
5.              for $epoch = 1$ to 500 do
6.                  Train the model using global set of features
7.                  Test the model
8.                  Evaluate $Accuracy_1$
9.              end for
10.         end for
11.     end for
12.     Set $selected\ features \leftarrow features\ selected\ by\ FastICA$
13.     for each $arrangement$ in $arrangements$ do
14.         for $fold = 1$ to 10 do
15.             for $epoch = 1$ to 500 do
16.                 Train the model using $selected\ features$
17.                 Test the model
18.                 Evaluate the $Accuracy_2$
19.             end for
20.         end for
21.     end for
22.     END
```

Since we are considering the different number of nodes in the hidden layer, we have chosen three different architectures of node arrangements- equal number of nodes in hidden layers, more nodes in 1st layer and less nodes in 2nd layer and less nodes in first layer, more nodes in 2nd layer. In the first stage of our experiment, the deep learning-based detection model is developed using TensorFlow, and the performance of this model is evaluated using the global set of features. In our data set, there are total 15,972 features. This large number of features takes much more time even though we have GPU based operations to minimize the computational cost. The following Table 5.10(A), 5.10(B)) describes the performance details of our model using the global set of features.

Table 5.10 (A): Performance details using Global Set of features

With Global set of features and 500 epochs						
	L32_L32	L32_L64	L64_L32	L64_L64	L64_L128	L128_L64
Accuracy	95.227	94.9040	95.3092	94.90734	95.1505	95.067
Loss	0.1576	0.17443	0.1566	0.1596	0.1645	0.1584
Recall	0.9543	0.94413	0.9440	0.9481	0.9482	0.9420
Precession	0.9260	0.92800	0.9370	0.9245	0.9296	0.9330
f1 score	0.9398	0.9351	0.9402	0.9356	0.9385	0.9371
AUC	0.9926	0.9914	0.9927	0.9927	0.9921	0.9921
TPR	0.9702	0.9637	0.964	0.9662	0.9663	0.9626
TNR	0.9257	0.9268	0.9363	0.9233	0.9290	0.9323
FPR	0.0742	0.0731	0.0636	0.0766	0.0709	0.0676
FNR	0.0297	0.0362	0.036	0.0337	0.0336	0.0373

Table 5.10 (B): Performance details using Global Set of features

With Global set of features and 500 epochs						
	L128_L128	L128_L512	L512_L512	L512_L1024	L1024_L512	L1024_L1024
Accuracy	95.877	95.309	95.956	95.793	95.392	**95.962**
Loss	0.1548	0.1613	0.1572	0.1758	0.1666	0.1548
Recall	0.9545	0.9503	0.9627	0.9627	0.9524	0.9648
Precession	0.9415	0.9312	0.9361	0.9320	0.9314	0.9343
f1 score	0.9475	0.9405	0.9490	0.9469	0.9415	0.9492
AUC	0.9930	0.9926	0.9933	0.9932	0.9921	0.9935
TPR	0.9705	0.9677	0.9756	0.9756	0.9690	0.9769

TNR	0.9408	0.9310	0.9356	0.9318	0.9311	0.9338
FPR	0.0591	0.0689	0.0643	0.0681	0.0688	0.0661
FNR	0.0294	0.0322	0.0243	0.0243	0.0309	0.0230

Table 5.10(A, B) shows that, we have achieved 95.96% of accuracy with equal number of nodes 1024 in each layer (Architectute1). The area under ROC curve (AUC) is 0.9935, very close to 1. Moreover, we have used FastICA for the compression of the features and run the model with 40,50,80 and 100 features. The highest accuracy level is 94.837% using 50 selected features and architecture1 with 512 nodes in L1 and L2 is considered here. In this case, the AUC is 0.9871 which is also very satisfactory. The performance details are shown in the Table 5.11(A, B), Table 5.12(A, B), Table 5.13(A, B) and Table 5.14(A, B).

Table 5.11 (A): Performance details using FASTICA with 40 features

With 40 selected features using FastICA and 500 epochs						
	L32_L32	L32_L64	L64_L32	L64_L64	L64_L128	L128_L64
Accuracy	92.729	93.457	92.725	93.619	**94.185**	92.648
Loss	0.2359	0.2220	0.2355	0.2217	0.2115	0.2223
Recall	0.8944	0.9007	0.8861	0.9131	0.9069	0.8988
Precession	0.9216	0.9318	0.9295	0.9268	0.9433	0.9155
f1 score	0.9060	0.9148	0.9054	0.9181	0.9239	0.9054
AUC	0.9850	0.9857	0.9851	0.9856	0.9860	0.9855
TPR	0.9334	0.9375	0.9289	0.9446	0.9417	0.9356
TNR	0.9171	0.9294	0.9244	0.9225	0.9419	0.9117
FPR	0.0828	0.0705	0.0755	0.0774	0.0580	0.0882
FNR	0.0665	0.0624	0.0710	0.0553	0.0582	0.0643

Table 5.11 (B): Performance details using FASTICA with 40 features

With 40 selected features using FastICA and 500 epochs						
	L128_L128	L128_L512	L512_L512	L512_L1024	L1024_L512	L1024_L1024
Accuracy	94.109	92.888	92.807	93.461	93.700	93.380
Loss	0.2117	0.1999	0.2013	0.1990	0.2066	0.1976
Recall	0.9129	0.9110	0.9193	0.9068	0.9192	0.9049

Precession	0.9377	0.9125	0.9057	0.9290	0.9230	0.9268
f1 score	0.9240	0.9102	0.9101	0.9158	0.9196	0.9147
AUC	0.9856	0.9864	0.9871	0.9868	0.9861	0.9871
TPR	0.945	0.9428	0.9475	0.9410	0.9482	0.9397
TNR	0.9343	0.9072	0.8987	0.9240	0.9192	0.9238
FPR	0.0656	0.0927	0.1012	0.0759	0.0807	0.0761
FNR	0.0549	0.0571	0.0524	0.0589	0.0517	0.0602

Table 5.12 (A): Performance details using FastICA with 50 features

With 50 selected features using FastICA and 500 epochs						
	L32_L32	L32_L64	L64_L32	L64_L64	L64_L128	L128_L64
Accuracy	93.702	94.271	94.514	94.028	93.706	93.703
Loss	0.2252	0.2132	0.2234	0.2124	0.2065	0.2148
Recall	0.9069	0.8945	0.9027	0.9090	0.8883	0.9130
Precession	0.9319	0.9572	0.9554	0.9369	0.9480	0.9264
f1 score	0.9185	0.9238	0.9278	0.9224	0.9164	0.9190
AUC	0.9859	0.9860	0.9859	0.9859	0.9860	0.9859
TPR	0.9412	0.93503	0.9397	0.9427	0.9311	0.9447
TNR	0.9299	0.9557	0.9540	0.9360	0.9470	0.9245
FPR	0.0700	0.0442	0.0459	0.0639	0.0529	0.0754
FNR	0.0587	0.0649	0.0602	0.0572	0.0688	0.0552

Table 5.12 (B): Performance details using FastICA with 50 features

With 50 selected features using FastICA and 500 epochs						
	L128_L128	L128_L512	L512_L512	L512_L1024	L1024_L512	L1024_L1024
Accuracy	94.188	94.596	**94.837**	94.275	93.946	93.704
Loss	0.2032	0.1897	0.1911	0.1870	0.1936	0.1893
Recall	0.9048	0.8986	0.9153	0.8863	0.9027	0.9071
Precession	0.9447	0.9607	0.9509	0.9637	0.9418	0.9310
f1 score	0.9240	0.9281	0.9326	0.9231	0.9209	0.9181
AUC	0.9863	0.9867	0.9871	0.9875	0.9874	0.9876
TPR	0.9405	0.9375	0.9468	0.9306	0.9391	0.9412
TNR	0.9438	0.9601	0.9505	0.9639	0.9396	0.9299
FPR	0.0561	0.0398	0.0494	0.0360	0.0603	0.0700
FNR	0.0594	0.0624	0.0531	0.0693	0.0608	0.0587

Table 5.13 (A): Performance details using FASTICA with 80 features

With 80 selected features using FastICA and 500 epochs						
	L32_L32	L32_L64	L64_L32	L64_L64	L64_L128	L128_L64
Accuracy	**94.347**	94.187	93.945	94.108	93.947	93.784
Loss	0.2248	0.2145	0.2256	0.2138	0.2077	0.2167
Recall	0.9006	0.8903	0.8945	0.8883	0.8862	0.8965
Precession	0.9536	0.9590	0.9482	0.9588	0.9567	0.9421
f1 score	0.9254	0.9225	0.9199	0.9214	0.9193	0.9182
AUC	0.9860	0.9858	0.9853	0.9854	0.9855	0.9844
TPR	0.9384	0.9327	0.9346	0.9315	0.9302	0.9356
TNR	0.9518	0.9576	0.9473	0.9575	0.9553	0.9413
FPR	0.0481	0.0423	0.0526	0.0424	0.0446	0.0586
FNR	0.0615	0.0672	0.0653	0.0684	0.0697	0.0643

Table 5.13 (B): Performance details using FASTICA with 80 features

With 80 selected features using FastICA and 500 epochs						
	L128_L128	L128_L512	L512_L512	L512_L1024	L1024_L512	L1024_L1024
Accuracy	93.7854	93.866	93.866	94.026	93.783	94.028
Loss	0.2052	0.1961	0.1975	0.1936	0.1982	0.1888
Recall	0.8925	0.8904	0.8925	0.8987	0.9068	0.8987
Precession	0.9464	0.9502	0.9493	0.9470	0.9338	0.9466
f1 score	0.9177	0.9186	0.9188	0.9213	0.9190	0.9212
AUC	0.9862	0.9862	0.9874	0.9874	0.9871	0.9883
TPR	0.9334	0.9323	0.9335	0.9370	0.9413	0.9370
TNR	0.9451	0.9492	0.9472	0.9455	0.9319	0.9455
FPR	0.0548	0.0507	0.0527	0.0544	0.0680	0.0544
FNR	0.0665	0.0676	0.0664	0.0629	0.0586	0.0629

Table 5.14 (A): Performance details using FASTICA with 100 features

With 100 selected features using FastICA and 500 epochs						
	L32_L32	L32_L64	L64_L32	L64_L64	L64_L128	L128_L64
Accuracy	94.029	93.867	94.027	93.786	93.704	94.189
Loss	0.2229	0.2193	0.2254	0.2159	0.2101	0.2154
Recall	0.8862	0.8862	0.8903	0.8924	0.8903	0.8883

Precession	0.9592	0.9554	0.9552	0.9471	0.9478	0.9618
f1 score	0.9204	0.9184	0.9207	0.9179	0.9168	0.9225
AUC	0.9838	0.9841	0.9838	0.9838	0.9836	0.9835
TPR	0.9303	0.9302	0.9325	0.9334	0.9322	0.9316
TNR	0.9574	0.9532	0.9534	0.9451	0.9450	0.9597
FPR	0.0425	0.0467	0.0465	0.0548	0.0549	0.0402
FNR	0.0696	0.0697	0.0674	0.0665	0.0677	0.0683

Table 5.14 (B): Performance details using FASTICA with 100 features

With 100 selected features using FastICA and 500 epochs						
	L128_L128	L128_L512	L512_L512	L512_L1024	L1024_L512	L1024_L1024
Accuracy	93.948	93.944	94.105	94.186	94.348	**94.434**
Loss	0.2103	0.1976	0.1993	0.1942	0.2013	0.1958
Recall	0.8883	0.8965	0.9047	0.9048	0.9088	0.9088
Precession	0.9552	0.9469	0.9435	0.9456	0.9468	0.9494
f1 score	0.9196	0.9203	0.9229	0.9238	0.9262	0.9274
AUC	0.9841	0.9858	0.9853	0.9858	0.9866	0.9866
TPR	0.9313	0.9358	0.9404	0.9405	0.9430	0.9430
TNR	0.9533	0.9454	0.9418	0.9438	0.9440	0.9461
FPR	0.0466	0.0545	0.0581	0.0561	0.0559	0.0538
FNR	0.0686	0.0641	0.0595	0.0594	0.0569	0.0569

Moreover, we have considered the accuracy Vs epoch and Loss vs epoch curves to represent the learning process of our model. We have also considered the Adam optimizer to optimize the learning process. The following Figures (Figure 5.12 and 5.13) shows the learning process of our model where the loss minimizes, and accuracy maximizes with the increase of epoch numbers.

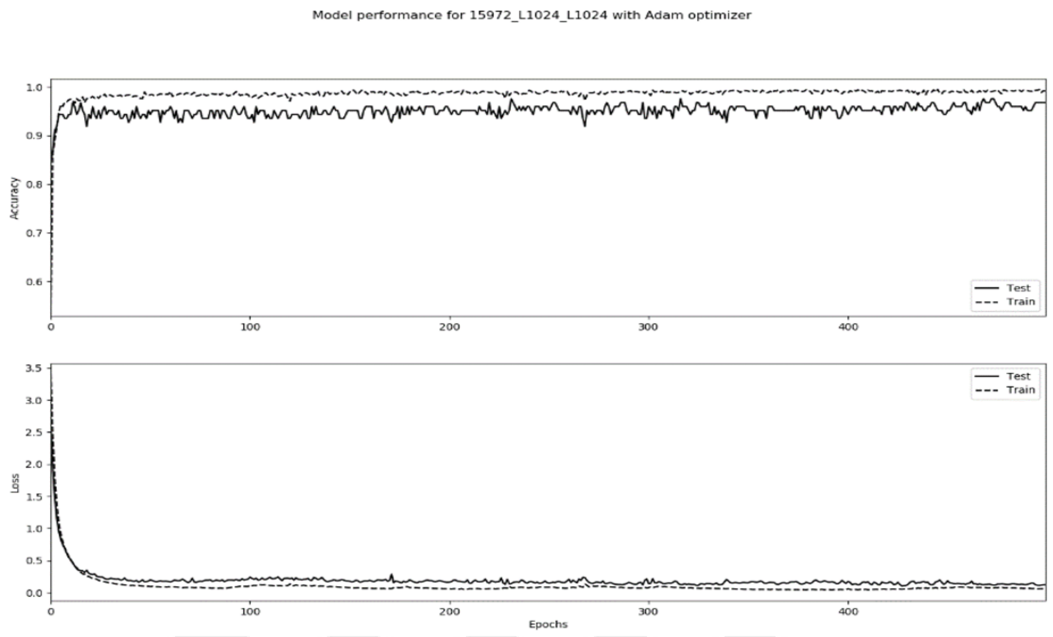


Figure 5.12: Accuracy VS epoch number and Loss VS epoch number for L1024_L1024 with global set of features

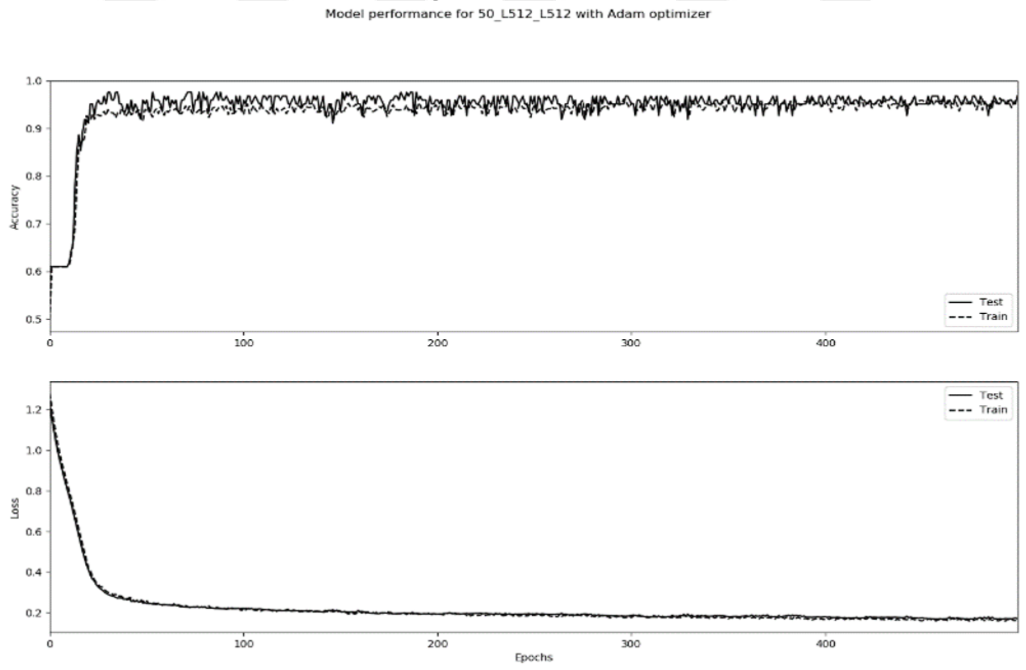


Figure 5.13: Accuracy VS epoch number and Loss VS epoch number for L512_L512 with 50 selected features using FastICA

5.3 Comparisons

Although our proposed methods show a better result, we also need to examine the capabilities of detecting the variants of known and unknown ransomware. We measure our model through effectiveness and accuracy criteria mentioned in the section 3, subsection 3.3. Therefore, we compare the performance of the proposed methods with other previous works, anti-virus engines and the similar methods like mRmR. The following subsection will explain in details.

5.3.1 Comparing with the other classifiers

We compare the accuracy of our proposed method with other classifiers in terms of the detection capabilities of the variants of known and unknown ransomware. For the purpose of the performance comparisons, the performance of the classifiers is determined through the evaluation criteria. For comparison purposes, we divide into two different scenarios.

5.3.1.1 Scenario one

In this scenario, we evaluate the performance of the Experiment One in section 5.3.1 method with the other three classifiers such as K-Nearest Neighbor (KNN), Decision Tree and Random Forest (RF). We tested the most informative features selected by the term frequency-inverse document frequency (TF-IDF) feature selection algorithm presented in Section 4.6.1. We explored the highest top features set with a testing 10-Fold cross validation techniques.

Table 5.15: Comparison of the Experiment One Method with other Classifiers

	FP Rate	**TP Rate**	**Precision**	**Recall**	**AUC**	**Detection Rate**
kNN	0.246	0.812	0.812	0.802	0.823	0.834
ANN	0.0261	0.986	0.981	0.979	0.983	0.986
SVM	0.016	0.986	0.976	0.971	0.973	0.979
RF	0.284	0.747	0.751	0.747	0.786	0.798
DT	0.061	0.945	0.945	0.944	0.946	0.952

Table 5.15 compares the accuracy of each classifier trained and tested with the selected top features. The ANN has performed the highest accuracy among all classifiers by presenting 0.986 of accuracy and less false positive rate. The RF classifier presented the lowest accuracy among the classifiers and shows 0.798 of accuracy, this poor performance is probably due to the overfitting problem during the training that the model is unable to generalize the new features in the testing phase.

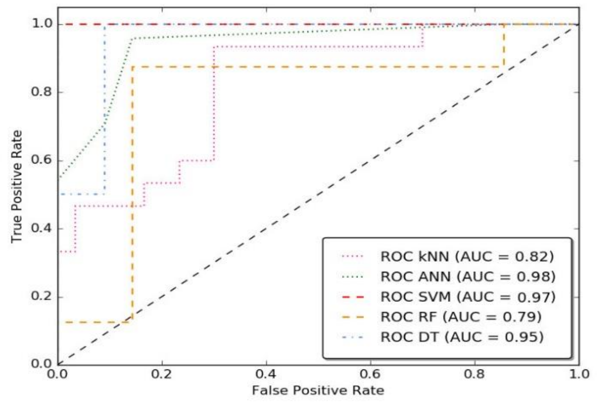


Figure 5.14. ROC Comparison of the Experiment One Method with other classifiers

On other hands, we evaluated the performance of the classifiers in terms of values of the Area under Curve (AUC) as evaluation metrics in this experiment. Figure 5.13 illustrates the AUC variations for the five classifiers. Three classifiers (ANN, SVM, and DT) have a high AUC rate while other two classifiers (kNN and RF) have a slightly lower AUC rate.

5.3.1.2 Scenario two

In this scenario, we compare the performance of the deep learning-based model presented in Experiment Three, in section 5.3.3 with other classifiers, we have chosen SVM, Random Forest, and Multiclass Classifier. The performance details are represented in Table 5.16. The comparison is illustrated in the following Table 5.17.

Table 5.16: The performance of SVM, Random Forest and Multi-Class classifiers

	SVM	**Random Forest**	**Multi Class**
Accuracy	89.9757	90.9458	88.1164
Recall	0.900	0.909	0.881
Precession	0.909	0.919	0.891
f1 score	0.901	0.910	0.882
AUC	0.909	0.971	0.955
TPR	0.900	0.909	0.881
FPR	0.081	0.070	0.101

From the comparison Table 5.17, we have observed that our designed model using deep learning and FastICA show better performance than the other classifiers. The proposed detection model has achieved 95.96% and 94.837% of accuracy using global and FastICA selected features respectively whereas the SVM achieved 89.97% and random Forest achieved 90.94% of accuracy.

Table 5.10,5.11,5.12,5.13 and 5.14 have shown that the equal number of nodes in L1 and L2 (Architecture 1) shows the best accuracy and lower false positive rate. Additionally, a higher epoch number shows better performance than lower epoch number.

Table 5.17: Comparison with other Classifiers in Experiment Three

	Accuracy Level	Area Under ROC Curve (AUC)
SVM	89.9757 %	0.909
Random Forest	90.9458 %	0.971
Multi Class Classifier	88.1164 %	0.955
Our Detection Model using Deep learning-based approach with features using FastICA [Architecture1 with 512 nodes in each layer]	94.837%	0.9871
Our Detection Model using Deep learning-based approach with global set of features [Architecture1 with 1024 nodes in each layer]	95.9629%	0.9935

5.3.2 Comparing with the previous work

In this section, we compare the performance of the proposed methods with the previous similar works based on the capabilities of detecting the variants of known and unknown ransomware. For the purpose of the performance comparisons, the effectiveness and the accuracy of the classifiers are determined via the evaluation criteria. For comparison benchmarks, we divide into two different scenarios as we highlighted in the following sections.

5.3.2.1 Scenario one

The purpose of this section is to compare the performance of the Experiment One, in section 5.3.1 of the proposed method with the previous similar works based on feature type and the classifier used. Sgandurra et al. proposed a machine learning-based framework with integrated features to identify the characteristics of the ransomware in the earlier phases, authors analyzed and extracted seven different features dynamically, they applied Logistic Regression classifier that achieved 96.3% detection rate with an area under the ROC curve of 0.995% (Sgandurra et al., 2016).

Table 5.18: Comparing the Experiment One proposed approach with earlier work

Works	**Classifier(s) Used**	**Det.Rate**
Experiment One of the Proposed Method	SVM and ANN	0.986
Sgandurra et al, EldeRan Framework	Logistic Regression	0.963
Mahbub et al, RansHunt framework	SVM	0.971
Ahmadian et al., 2entFOX Framework	Bayesian belief network	0.985
Alhawi at al., NetConverse scheme	BN, J48, kNN, MLP, RF and LMT	0.971
Zhang et al.	DT, RF, KNN, NB and GBDT	0.914
Poudyal at al.	BN, LR, SVM, DT, RF and ADA	0.965

Similar work was presented by Mahbub and Mahbubur using integrated features from static and dynamic analysis with a machine learning algorithm. Authors proposed

RansHunt framework to detect ransomware using support vector machine (SVM) with unique features. They claimed the hybrid analysis method can early detect the new ransomware variants (Hasan & Rahman, 2017). The RansHunt framework achieved an accuracy of 97.10 with less positive rate. A work of Zhang et al. employed the opcode sequences features with five machine-learning algorithms for the detection of ransomware. The classifier showed an accuracy of 91.43%. The Table 5.18. summarizes the comparison result of our proposed method against with similar works (Zhang et al., 2019).

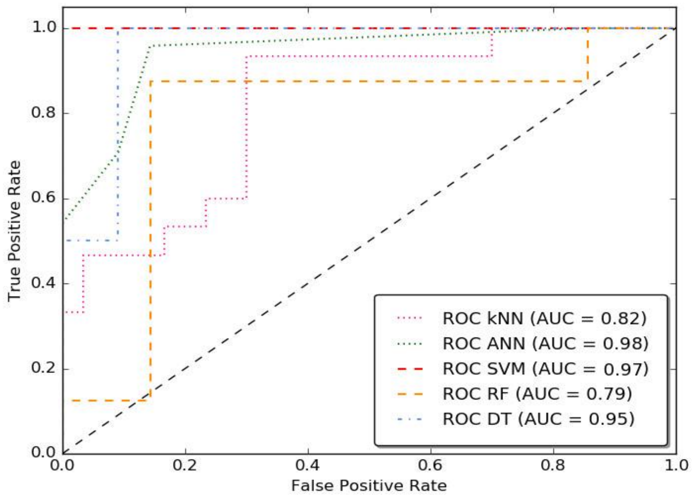


Figure 5.15: Comparison of the Experiment One method with other classifiers

5.3.2.2 Scenario two

In this section, we compare the performance of Experiment Two, in section 5.3.2 of the proposed method with the previous similar works based on the capabilities of detecting the variants of known and unknown ransomware. Non-signature-based approach for the detecting of the obfuscated malware samples has proposed by the Vinod et al et al. This method employed Minimum Redundancy and Maximum Relevance (mRmR) method and the Principal Component Analysis (PCA) with different mnemonic n–grams to extract predominant features (Vinod et al., 2012).

Table 5.19: Comparing the Experiment Two proposed approach with earlier work

Works	Classifier(s) Used	Detection Rate
Proposed	kNN, LR, SVM, RF and DT	0.981
Vinod et al et *al*.	NB, SMO, IBK, J48, ADA and RF	0.941
Iglesias and Zseby	DT, kNN, NB, LASSO-LAR, ANN and SVM	0.954
Sgandurra et al	Logistic Regression	0.963
Ye et *al*.	OOA_Apriori, OOA_FP-Growth and OOA_Fast_FP-Growth	0.931
Mahbub et *al*.	SVM	0.971

For classification purposes, several supervised machine learning algorithms are used that obtained detection accuracy of 94.1% with mRmR generated features. Similar work was presented by the Iglesias and Zseby proposing a feature reduction method for the network traffic using combined feature selection techniques such as SAM, LASSO, WMR, and mRmR (Iglesias & Zseby, 2015). In their work, commonly used 41 traffic features have been reduced into 16 features based on their contribution to the anomaly detection. To evaluate the proposed combined feature selection approach, the authors utilized six classification algorithms with fivefold cross-validation. The experimental results reveal a detection accuracy ranging from 0.27 to 95.48 with mRmR generated features.

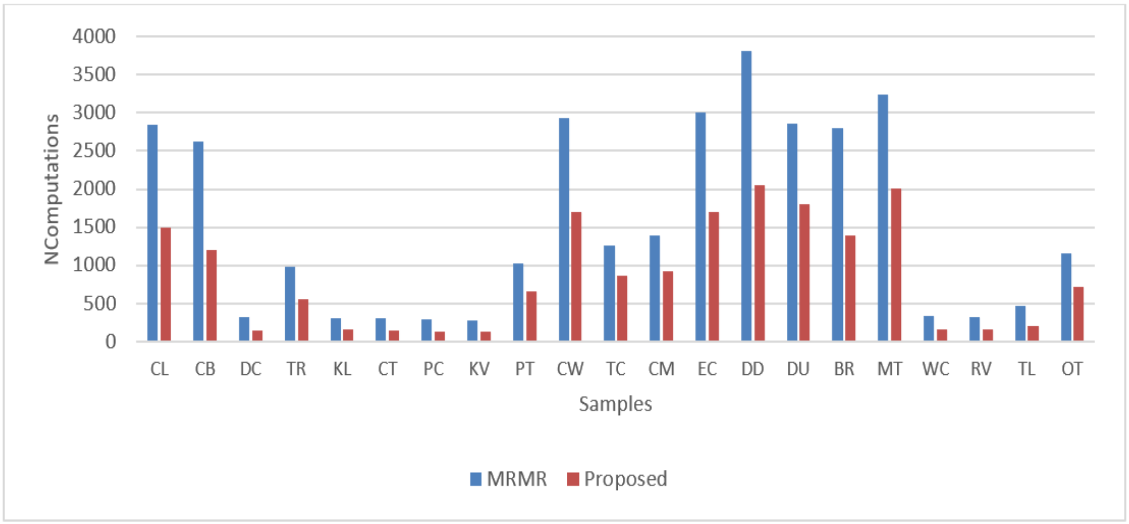


Figure 5.16: Compares the number of evaluations both mRmR and the EmRmR method

Sgandurra et al. have proposed a Logistic Regression classifier-based framework for the detection of the ransomware with integrated features. The authors dynamically analyzed 11 different ransomware classes and extracted seven features to characterize the behavior of the ransomware in the earlier phases of the attack. In their work, the LR has achieved a 96.3% detection rate with an area under the ROC curve of 0.995% and a less positive rate (Sgandurra et al., 2016). Ye et al. have discussed the detection of the polymorphic and the malicious metamorphic executables. Authors proposed Intelligent Malware Detection System (IMDS) using Objective Oriented Association (OOA) mining. They applied the Max-Relevance algorithm to select the informative API calls with regard to the class labels. To generate association rules among API calls, they adapted OOA mining techniques for classification (Ye, Wang, Li, Ye, & Jiang, 2008). A work of Mahbub and Mahbubur presented an integrated feature from static and dynamic analysis with a machine learning algorithm. Authors proposed RansHunt framework to detect ransomware using a support vector machine (SVM) with unique features. They claimed that the hybrid analysis method could early detect the new ransomware variants with an accuracy of 0.971 and a less positive rate. Table 5.19 shows the comparison of the proposed work with the previous work (Hasan & Rahman, 2017).

5.3.3 Comparing mRmR with the proposed EmRmR method

In this section, we evaluate the performance of the proposed method in section 4, subsection 4.6.2 with the original mRmR in terms of running time and the number of calculations on different feature sets. For the experiments, we employed a dataset consists of 14 different ransomware and 7 benign classes as presented in section 3, subsection 3.2.1.2, and tested each separately. We computed the runtime by counting the total time that the mRmR and the EmRmR method taking to select specific features from the collected logs. We adjusted the number of features to be selected as fifty features for both mRmR and the EmRmR method. The average runtime analysis and the number of computations of the mRmR and the mRmR method with mutual information (MI) on the various datasets are depicted in Figures 5.15-5.16.

Overall, the proposed method significantly outperformed the original mRmR method by reducing the computational time to an average of 9.6 minutes for the entire dataset as shown in Figure 5.17. The proposed method needs less than one minute to finish execution for the datasets with a large number of features like DD, CB, DU, CL and MT, whereas mRmR requires two minutes and 85 seconds. Regarding the datasets where the number of samples is small like DC, PC and KV, the runtime demanded by the EmRmR method reaches 91 up to the 46 seconds. This indicates that the proposed method is more than three times faster than the original mRmR.

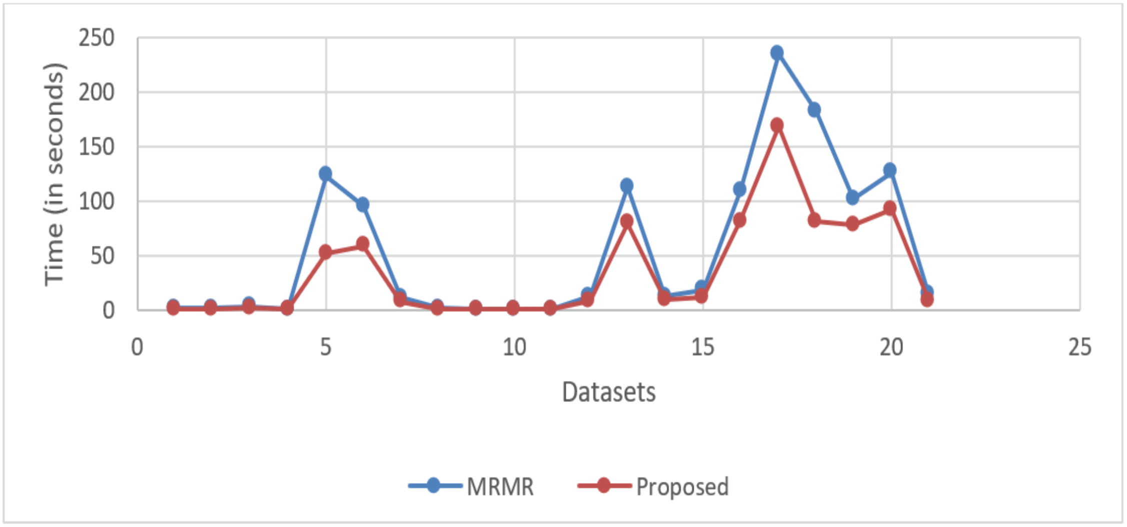


Figure 5.17: Time-complexity for mRmR and the Proposed EmRmR Method

In addition, we explored the number of computations executed by both mRmR and the Enhanced mRmR method on each of the aforementioned datasets. Similar to the previous running time analysis, again we set the number of features to select as 50 and calculated the number of evaluations required by the mRmR and the unnecessary evaluations of MI avoided by the EmRmR method. It is interesting to note that the proposed method has achieved important enhancement concerning the amount of the evaluations performed by the mRmR, particularly using samples with a large number of noisy features like CW, EC, DD, MT, EC, and DU. Generally, the EmRmR method has a lower margin of 18336 number of evaluations than the mRmR which is 43.65% lower on the whole datasets.

Table 5.20: Comparison of mRmR and the proposed EmRmR method

	mRmR	Proposed EmRmR	Difference
Runtime (s)	1169.42	742.83	426.59
Evaluations	32542	18336	14206

The reason for this reduced number of evaluations is due to the accumulating of mutual information computations achieved by the EmRmR. The run time and the number of evaluations for the mRmR and the EmRmR method on different datasets are presented in Table 5.20-5.21.

5.3.4 Comparison with AV scanners

Although our proposed method of Experiment One, in section 5.3.1, shows a better result, we also need to examine the capabilities of detecting the variants of known and unknown ransomware. We measure our classifiers through effectiveness and accuracy criteria and compare the performance of our proposed method with anti-virus engines. For comparative benchmarks, we selected the five Anti-Virus (AV) scanners with the highest detection rate available at VIRUSTOTAL service. VirusTotal (VT) is a web service that allows the analysis of a given malware sample by the signature-based engines of different AntiVirus vendors. All the engines are always kept up-to-date with the latest version of the signatures. A submission of a malware sample to VirusTotal at a given point in time thus provides a snapshot on the ability of the different signature-based engines to correctly identify a threat in such samples.

The detection performance of each AV scanners is evaluated using standard accuracy measurements such as True Positive Rate (TPR), False Positive Rate (FPR), ROC curve and the detection rate. From the experimental results, we observe that ANN significantly outperformed the other approaches and showed 0.986 of AUC, it also presented TPR of 0.988 and FBR of 0.036, despite ANN's false positive rate is higher than 3 of the AVs as shown in Table 5.22.

Table 5.21: Comparing mRmR and EmRmR based on the number of calculations and run Time on different datasets

Dataset	Acronym	Class	Sample	Features	mRmR		EmRmR	
					#Calc	#RTime (s)	# Calc	# RTime (s)
WannaCry	WC	M	74	8263	329	1.81	165	0.56
Reveton	RV	M	50	5021	316	1.75	163	0.53
Torrent Locker	TL	M	108	1102	460	2.84	205	1.43
Dirty Decrypt	DD	M	51	7539	314	1.03	153	0.46
CryptLocker	CL	M	173	44869	2850	103	1493	0.62
Cerber	CB	M	171	44235	2630	96	1210	0.59
Trojan-Ransom	TR	M	82	18593	986	11.2	560	7.82
Kollah	KL	M	73	7159	304	1.72	159	0.52
Citroni	CT	M	67	8036	303	1.01	138	0.41
Pgpcoder	PC	M	46	8172	283	0.95	134	0.43
Kovter	KV	M	23	7985	271	0.91	128	0.39
Petya	PT	M	89	17652	1030	12.3	654	8.62
CryptoWall	CW	M	151	43281	2937	113	1702	79.5
TeslaCrypt	TC	M	96	20352	1262	13.2	865	9.36
Compression	CM	B	225	26350	1390	18.4	920	11.5
Encryption	EC	B	172	44856	3001	110	1700	82.1
Data Destruction	DD	B	401	54385	3807	225	2053	139
Drivers Updater	DU	B	230	46523	2862	183	1802	110
Browsers	BR	B	152	45383	2804	102	1398	78
Multimedia tools	MT	B	182	40359	3240	126	2014	92
Others	OT	B	96	22986	1163	14.3	720	8.2

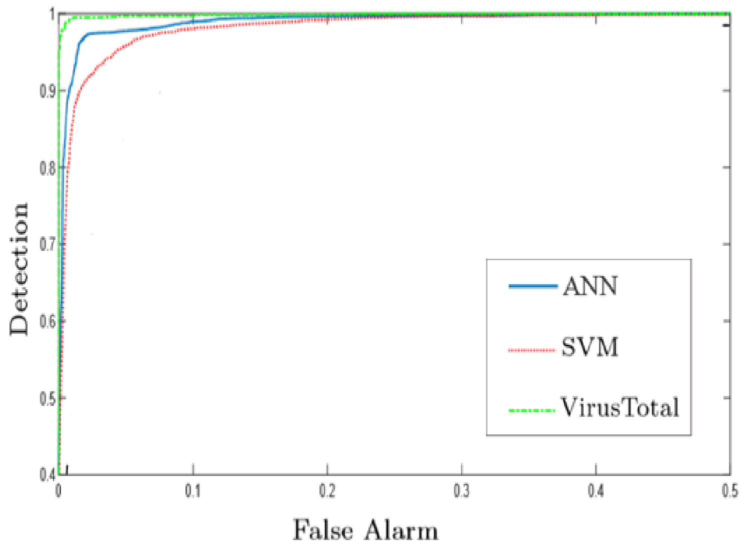


Figure 5.18: Comparison of the proposed method to VirusTotal

The other hands, the VirusTotal achieved the highest accuracy over the SVN and ANN algorithms with a detection rate of 0.989 and 0.986, though ANN classifier outperforms 3 out of 5 top selected AV scanners. This is due to the most common detection method used by the antivirus is signature-based detection; which implies that VirusTotal AV engines already have a matched signature of these datasets. In addition, some of our data set are publicly available for a longer period.

Table 5.22: Comparison of our proposed approach with AVs

	TP Rate	**FP Rate**	**AUC**	**Detection Accuracy**
SVM	0.0993 ± 0.0837	0.0371 ± 0.006	0.977 ± 0.238	0.9760±0.8910
ANN	0.0988 ± 0.0841	0.0360 ± 0.007	0.986 ± 0.186	0.9870±0.9010
AV1	0.0203 ± 0.0079	0.0186 ± 0.0080	0.977 ± 0.238	0.989 ± 0.0273
AV2	0.0159 ± 0.0060	0.0166 ± 0.0048	0.986 ± 0.186	0.986 ± 0.0262
AV3	0.0274 ± 0.0082	0.0396 ± 0.0080	0.977 ± 0.238	0.9569 ± 0.0173
AV4	0.0205 ± 0.0079	0.0000 ± 0.0000	0.986 ± 0.186	0.9369 ± 0.0173
AV5	0.0101 ± 0.0079	0.0496 ± 0.0080	0.977 ± 0.238	0.9160 ± 0.0273

The Figure 5.18 shows the comparison of the detection rate and false alarm of the VIRUSTOTAL and our method using both ANN and SVM. From the result of Figure 5.18, it is obvious that the AUC of VIRUSTOTAL outperforms ANN and SVM algorithms, however, according to the false alarm criteria, the VirusTotal shows an average of 5.4% and is worse than ANN classifier that presents an average error rate of 2.3%. ANN provided a better accuracy regarding SVM with 4.4% error rate.

5.4 Discussions

In this section, we present the discussions on the experimental results obtained from the previous sections. We discuss three different experiments such as an automated dynamic behavioural detection framework. Another important detection model which is a system call refinement-based enhanced minimum redundancy maximum relevance method for ransomware early detection and the last model which is the avoiding future digital extortion through robust protection against ransomware threats using deep learning-based adaptive approaches, the following subsection will explain detail.

5.4.1 Automated dynamic behavioral detection framework

The avoidance techniques that ransomware employs such as obfuscation and/or packing makes it difficult to analyse such programs statically. Although many ransomware detections studies have been conducted, they are limited to a small portion of the attack's characteristics. To this end, this research proposed a framework for the behavioural-based dynamic analysis of ransomware with integrated valuable feature sets.

This method analysed 673 real-world ransomware samples that infect Windows platforms. We collected 1,254 ransomware samples of 14 newly emerged different ransomware families from several sources such as VirusShare and VirusTotal. We focused these malicious behaviours on their suspicious intentions of 14 families. The activities performed by the malicious program is recorded in the sandbox in a controlled environment and obtained generated report of the samples as JSON format. The size of the report generated by the sandbox occupies hundreds of MBs, analysing and examining each report manually is experimentally infeasible, therefore, we build our own parsing algorithm to convert JSON formatted string representations to key-pair objects. The feature parsing reads the JSON files from all sandbox output reports and then parsed to get the required features to reduce the search space.

Then, the extracted a set of features that forms an integrated set of features that could indicate what the ransomware strain is actually doing on the system. Term Frequency-Inverse document frequency (TF-IDF) was employed to select the most useful features from the analysed samples. Support Vector Machine (SVM) and Artificial Neural Network (ANN) were utilized to develop and implement a machine learning-based detection model able to recognize certain behavioural traits of high survivable ransomware attacks. Creating SVM and ANN models require to set various parameters and test individually to select the best model among them. In this study, SVM kernel functions such as the linear kernel, polynomial kernel and Radial basis function (RBF) are used.

In this study we applied three different experiments along with a variety of outputs. Experimental evaluation indicates that the proposed framework achieved an area under the ROC curve of 0.987 and a few false positive rates 0.007. The experimental results indicate that the proposed framework can detect high survivable ransomware in the early stage accurately. To empirically evaluate the proposed approach, we compared the experimental results with previous work, other classifiers and the VirusTotal service.

5.4.2 A System Call Refinement-Based EmRmR Method for Ransomware Detection

Ransomware is a special type of malicious software that encrypts the user's assets and makes it unavailable to the users until a ransom is paid to the ransomware author. Distribution of the crypto-ransomware can happen through a very large infection vector including application and browser vulnerabilities, extraction of ZIP files, malicious payload e.g. cryptoWall, JRE vulnerability e.g. DMA locker, exploit kits such as neutrino, eternal blue, eternal romance, etc. Such attacks have become one of the most widespread malwares that poses serious threat to both individuals and business organizations. Against this destructive malicious program, the dynamic analysis approach is the most popular approach for detecting such an attack. The majority of dynamic analysis relies on the system calls as these provide an interface for programs to request service from the operating system. However, to hide its malicious behaviour, ransomware invokes a huge amount of system calls that contain the redundancy and the irrelevant system calls that the ransomware authors inject in the actual execution flow of suspicious binaries generate a high noisy behavioural sequence that adversely impacts in the induction of the classifiers.

To this end, we proposed a non-signature-based detection approach on the effective windows API call sequences using supervised machine learning techniques. We also introduced a refinement process to reduce the size of the system call traces gathered from the dynamic analysis and filter the system call that do not describe the behaviour of the malware. Then, five machine-learning classifiers (Decision Tree (DT), K-Nearest Neighbour (KNN), Logistic Regression (LR), Random Forest (RF) and Support Vector Machine (SVM)) are trained on the refined system call sequences. We tested the classifiers on the various performance metrics with an extensive experimental evaluation to determine the effectiveness of the proposed method.

To achieve this objective, we proposed an Enhanced Maximum-Relevance and Minimum-Redundancy (EmRmR) filter method to remove the noisy features and select the most relevant subset of features to characterize the real behaviour of the ransomware. Unlike the original mRmR, the EmRmR avoids unnecessary computations intrinsic in the original mRmR algorithms with small number of evaluations. To prove this, we experimented with the EmRmR algorithm on datasets with different feature-sets and compared with mRmR. The EmRmR method was three times faster than the original mRmR.

In addition, this work has introduced a refinement process to reduce the size of the program's call traces by removing those windows API calls that do not have a strong indication for describing the critical behaviour of the ransomware. To do this, Windows API calls performed by the ransomware are deeply studied through the descriptions provided by Microsoft's website using a customized python script. After accomplishing extensive experimental evaluations, and comparing with existing behavioural-based detection approaches, the proposed method has shown to be effective for discriminating the behavior of ransomware, and indicates a high detection accuracy with few false-positive rates. For comparative benchmarks, we evaluated the performance of the proposed method with the previous similar works based on the capabilities of detecting the variants of known and unknown ransomware.

5.4.3 Avoiding future digital extortion through robust protection against ransomware threats using deep learning based adaptive approaches

Digital extortion has become a major cyber risk for many organizations; small-medium enterprises (SME) to large enterprises business and individual entrepreneurs. In recent years, mass ransomware attacks not only targeted to the individuals but also have been proliferated into the large business organizations such as courier companies FedEx and TNT, Maerx, WPP (the world's largest advertising agency), pharmaceutical company Reckitt Benckiser and Kingdom's National Health Service.

Ransomware is a kind of malware which is the main threat for digital extortion and has caused many organizations to lose huge revenue by paying much bigger ransom demands to the cyber criminals in recent years. Explosive growth of ransomware is due to the existing large infection vector such as social engineering, email attachment, zip file download, browsing malicious site, infected search engine which are boosted dramatically by easily available cryptographic tools, Ransomware As a Service (RaaS), increased cloud storage and off-the-self ransomware toolkits. The large infection vector and available toolkits not only grew ransomware extremely, but also made them more obfuscated, encrypted and varying patterns in the new variants. This, in turn, caused the conventional supervised analysis and detection engine to fail to detect the new variants of ransomware.

This research addresses the limitations of a conventional supervised detection engine and proposes semi-supervised framework to compute the inherent latent sources of the varying patterns in the new variants in an unsupervised way using deep learning approaches. We Proposed a framework that analysis ransomware based on the runtime behavior of program and deep learning based semi-supervised technique, and then extracts the inherent characteristics in the varying patterns from the unlabeled ransomware obtained from the wild which is scalable to accommodate upcoming malicious executables. Then the unsupervised learned model is combined with supervised classification, thus constructing an adaptive detection model.

The novelty of our proposed approach is that deep learning based semi-supervised technique can extract dynamics of behavioral patterns from the new variants of ransomware obtained from the wild and can integrate the latent sources to the supervised classifier, making the detection engine independent of manual signature generation and robust to the changes. The proposed framework has been verified using real ransomware data with a dynamic analysis testbed. Our extensive experimental results and discussion demonstrate that the proposed adaptive framework can successfully identify different variants of ransomware and achieve higher performance than existing supervised approaches.

5.5 Summary

This section discusses the experimental results of the proposed a behavioural ransomware detection framework using a machine-learning approach. Several experiments are implemented including an automated dynamic behavioural analysis approach for the detection of the ransomware. In this experiment, the key observation of this research is to investigate an integrated set of features that could indicate what the ransomware strain is actually doing on the system. The TF-IDF algorithm has shown to be an effective approach for ranking and weighting behavioural features. We developed a detection model for HSR by utilizing Support Vector Machine and Artificial Neural Network algorithms using integrated valuable features that generated. Three different experimental evaluation was conducted to measure the performance of the proposed method. Through our experimental results, the proposed approach has shown to be easy to train and test. Another important experiment was carried out, a non-signature-based method for the detection of the ransomware on windows API call sequences using a supervised learning approach has been proposed. In this experiment, we proposed an enhanced maximum relevance and minimum redundancy (EmRmR) method that selects the same features as the original mRmR but significantly faster. To carry out this objective, we avoided the unnecessary computations of the mutual information by accumulating the duplications of each MI iterations. To prove this, we experimented with the EmRmR algorithm on datasets with different feature-sets and compared with mRmR. For detection purposes, we employed five machine learning classifiers on $n-grams$ of length 2,3,4 and 5 from each system call sequence made by the malicious and benign samples. Finally, we experimented an adaptive approach that can extract inherent nature of exploitation and encryption of new variants of ransomware. The extracted features are integrated into the supervised detection engine to build an adaptive model. The proposed methods have been tested in a real ransomware dynamic analysis engine with real ransomware data. Our experimental results demonstrate that the proposed model achieves significant performance improvement over supervised detection approaches and achieves high accuracies.

REFERENCES

Acid, S., De Campos, L. M., & Fernández, M. (2011). *Minimum redundancy maximum relevancy versus score-based methods for learning Markov boundaries.* Paper presented at the 2011 11th International Conference on Intelligent Systems Design and Applications.

Adamov, A., & Carlsson, A. (2017). *The state of ransomware. Trends and mitigation techniques.* Paper presented at the 2017 IEEE East-West Design & Test Symposium (EWDTS).

Ahmadian, M. M., & Shahriari, H. R. (2016). *2entFOX: A framework for high survivable ransomwares detection.* Paper presented at the 2016 13th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC).

Ahmadian, M. M., Shahriari, H. R., & Ghaffarian, S. M. (2015). *Connection-monitor & connection-breaker: A novel approach for prevention and detection of high survivable ransomwares.* Paper presented at the 2015 12th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC).

Aidan, J. S., Verma, H. K., & Awasthi, L. K. (2017). *Comprehensive Survey on Petya Ransomware Attack.* Paper presented at the 2017 International Conference on Next Generation Computing and Information Systems (ICNGCIS).

Al-rimy, B. A. S., Maarof, M. A., & Shaid, S. Z. M. (2017). *A 0-day aware crypto-ransomware early behavioral detection framework.* Paper presented at the International Conference of Reliable Information and Communication Technology.

Al-rimy, B. A. S., Maarof, M. A., & Shaid, S. Z. M. (2018). Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions. *Computers & Security, 74*, 144-166.

Al-rimy, B. A. S., Maarof, M. A., & Shaid, S. Z. M. (2019). Crypto-ransomware early detection model using novel incremental bagging with enhanced semi-random subspace selection. *Future Generation Computer Systems, 101*, 476-491.

Alhawi, O. M., Baldwin, J., & Dehghantanha, A. (2018). Leveraging machine learning techniques for windows ransomware network traffic detection. In *Cyber Threat Intelligence* (pp. 93-106): Springer.

Almashhadani, A. O., Kaiiali, M., Sezer, S., & O'Kane, P. (2019). A multi-classifier network-based crypto ransomware detection system: A case study of Locky ransomware. *IEEE Access, 7*, 47053-47067.

Alzarooni, K. (2012). *Malware variant detection.* UCL (University College London),

Amamra, A., Robert, J. M., & Talhi, C. (2015). Enhancing malware detection for Android systems using a system call filtering and abstraction process. *Security and Communication Networks, 8*(7), 1179-1192.

Anderson, H. S., Kharkar, A., Filar, B., & Roth, P. (2017). Evading machine learning malware detection. *black Hat*.

Andronio, N. (2015). *Heldroid: Fast and efficient linguistic-based ransomware detection.*

Andronio, N., Zanero, S., & Maggi, F. (2015). *Heldroid: Dissecting and detecting mobile ransomware.* Paper presented at the International Symposium on Recent Advances in Intrusion Detection.

Aurangzeb, S., Aleem, M., Iqbal, M. A., & Islam, M. A. (2017). Ransomware: A Survey and Trends. *Journal of Information Assurance & Security, 6*(2).

Bhardwaj, A., Avasthi, V., Sastry, H., & Subrahmanyam, G. (2016). Ransomware digital extortion: a rising new age threat. *Indian Journal of Science and Technology, 9*(14), 1-5.

Bishop, C. M. (2006). *Pattern recognition and machine learning*: springer.

Breiman, L. (2001). Random forests. *Machine learning, 45*(1), 5-32.

Brewer, R. (2016). Ransomware attacks: detection, prevention and cure. *Network Security, 2016*(9), 5-9.

Cabaj, K., Gawkowski, P., Grochowski, K., & Osojca, D. (2015). Network activity analysis of CryptoWall ransomware. *Przeglad Elektrotechniczny, 91*(11), 201-204.

Cabaj, K., & Mazurczyk, W. (2016). Using software-defined networking for ransomware mitigation: the case of cryptowall. *Ieee Network, 30*(6), 14-20.

Carrasquilla, J., & Melko, R. G. (2017). Machine learning phases of matter. *Nature Physics, 13*(5), 431-434.

Chen, Q., & Bridges, R. A. (2017). *Automated behavioral analysis of malware: A case study of wannacry ransomware.* Paper presented at the 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA).

Chen, Q., Islam, S. R., Haswell, H., & Bridges, R. A. (2019). *Automated Ransomware Behavior Analysis: Pattern Extraction and Early Detection.* Paper presented at the International Conference on Science of Cyber Security.

Chou, T.-S., Yen, K. K., & Luo, J. (2008). Network intrusion detection design using feature selection of soft computing paradigms. *International journal of computational intelligence, 4*(3), 196-208.

Collberg, C. S., Thomborson, C. D., & Low, D. W. K. (2003). Obfuscation techniques for enhancing software security. In: Google Patents.

Conti, M., Gangwal, A., & Ruj, S. (2018). On the economic significance of ransomware campaigns: A Bitcoin transactions perspective. *Computers & Security, 79*, 162-189.

Corrigan, K. (2017). *Ransomware: a growing epidemic for business.* Utica College,

Da-Yu, K., HSIAO, S.-C., & Raylin, T. (2019). *Analyzing WannaCry Ransomware Considering the Weapons and Exploits.* Paper presented at the 2019 21st International Conference on Advanced Communication Technology (ICACT).

Dada, E. G., Bassi, J. S., Chiroma, H., Adetunmbi, A. O., & Ajibuwa, O. E. (2019). Machine learning for email spam filtering: review, approaches and open research problems. *Heliyon, 5*(6), e01802.

Darshan, S. S., & Jaidhar, C. (2018). Performance evaluation of filter-based feature selection techniques in classifying portable executable files. *Procedia Computer Science, 125*, 346-356.

Das, S. (2001). *Filters, wrappers and a boosting-based hybrid for feature selection.* Paper presented at the Icml.

Dietterich, T. G. (2000). *Ensemble methods in machine learning.* Paper presented at the International workshop on multiple classifier systems.

Dietterich, T. G., & Kong, E. B. (1995). *Machine learning bias, statistical bias, and statistical variance of decision tree algorithms*. Retrieved from

Doguet, J. J. (2012). The nature of the form: Legal ad regulatory issues surrounding the bitcoin digital currency system. *La. L. Rev., 73,* 1119.

Faruki, P., Laxmi, V., Gaur, M. S., & Vinod, P. (2012). *Mining control flow graph as api call-grams to detect portable executable malware.* Paper presented at the Proceedings of the Fifth International Conference on Security of Information and Networks.

Francillon, A., & Castelluccia, C. (2008). *Code injection attacks on harvard-architecture devices.* Paper presented at the Proceedings of the 15th ACM conference on Computer and communications security.

Fukushima, Y., Sakai, A., Hori, Y., & Sakurai, K. (2010). *A behavior based malware detection scheme for avoiding false positive.* Paper presented at the 2010 6th IEEE workshop on secure network protocols.

Gallegos-Segovia, P. L., Bravo-Torres, J. F., Larios-Rosillo, V. M., Vintimilla-Tapia, P. E., Yuquilima-Albarado, I. F., & Jara-Saltos, J. D. (2017). *Social engineering as an attack vector for ransomware.* Paper presented at the 2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON).

Gandotra, E., Bansal, D., & Sofat, S. (2014). Malware analysis and classification: A survey. *Journal of Information Security, 2014*.

Gazet, A. (2010). Comparative analysis of various ransomware virii. *Journal in computer virology, 6*(1), 77-90.

Goel, D., & Jain, A. K. (2018). Mobile phishing attacks and defence mechanisms: State of art and open research challenges. *Computers & Security, 73,* 519-544.

Gostev, A., Unuchek, R., Garnaeva, M., Makrushin, D., & Ivanov, A. (2016). IT Threat Evolution in Q1 2016. *Kapersky 2015 Report, Kapersky L.*

Greamo, C., & Ghosh, A. (2011). Sandboxing and virtualization: Modern tools for combating malware. *IEEE Security & Privacy, 9*(2), 79-82.

Grimes, G. A., Hough, M. G., & Signorella, M. L. (2007). Email end users and spam: relations of gender and age group to attitudes and actions. *Computers in Human Behavior, 23*(1), 318-332.

Gu, Q., Li, Z., & Han, J. (2012). Generalized fisher score for feature selection. *arXiv preprint arXiv:1202.3725*.

Gupta, S., & Kumar, P. (2015). An immediate system call sequence based approach for detecting malicious program executions in cloud environment. *Wireless Personal Communications, 81*(1), 405-425.

Hampton, N., Baig, Z., & Zeadally, S. (2018). Ransomware behavioural analysis on windows platforms. *Journal of information security and applications, 40,* 44-51.

Hampton, N., & Baig, Z. A. (2015). Ransomware: Emergence of the cyber-extortion menace.

Han, S., Huang, H., & Qin, H. (2017). Automatically Redundant Features Removal for Unsupervised Feature Selection via Sparse Feature Graph. *arXiv preprint arXiv:1705.04804*.

Hasan, M. M., & Rahman, M. M. (2017). *RansHunt: A support vector machines based ransomware analysis framework with integrated feature set.* Paper presented at the 2017 20th International Conference of Computer and Information Technology (ICCIT).

Heckerman, D. (2008). A tutorial on learning with Bayesian networks. In *Innovations in Bayesian networks* (pp. 33-82): Springer.

Hoopes, J. (2009). *Virtualization for security: including sandboxing, disaster recovery, high availability, forensic analysis, and honeypotting*: Syngress.

Huda, S., Islam, R., Abawajy, J., Yearwood, J., Hassan, M. M., & Fortino, G. (2018). A hybrid-multi filter-wrapper framework to identify run-time behaviour for fast malware detection. *Future Generation Computer Systems, 83*, 193-207.

Hyvärinen, A., & Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural networks, 13*(4-5), 411-430.

Iglesias, F., & Zseby, T. (2015). Analysis of network traffic features for anomaly detection. *Machine learning, 101*(1-3), 59-84.

Jiang, X., Wangz, H. J., Xu, D., & Wang, Y.-M. (2007). *Randsys: Thwarting code injection attacks with system service interface randomization.* Paper presented at the 2007 26th IEEE International Symposium on Reliable Distributed Systems (SRDS 2007).

Jones Jr, C., & Muhammad, J. Ransomware and Its Impact on Modern Society.

Kalaimannan, E., John, S. K., DuBose, T., & Pinto, A. (2017). Influences on ransomware's evolution and predictions for the future challenges. *Journal of Cyber Security Technology, 1*(1), 23-31.

Kara, I., & Aydos, M. (2018). *Static and dynamic analysis of third generation cerber ransomware.* Paper presented at the 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT).

Kendzierskyj, S., & Jahankhani, H. (2019). *The Role of Blockchain in Supporting Critical National Infrastructure.* Paper presented at the 2019 IEEE 12th International Conference on Global Security, Safety and Sustainability (ICGS3).

Kharraz, A., & Kirda, E. (2017). *Redemption: Real-time protection against ransomware at end-hosts.* Paper presented at the International Symposium on Research in Attacks, Intrusions, and Defenses.

Kharraz, A., Robertson, W., Balzarotti, D., Bilge, L., & Kirda, E. (2015). *Cutting the gordian knot: A look under the hood of ransomware attacks.* Paper presented at the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment.

Kirda, E. (2017). *UNVEIL: a large-scale, automated approach to detecting ransomware (keynote).* Paper presented at the 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER).

Kok, S., Abdullah, A., Jhanjhi, N., & Supramaniam, M. (2019). Prevention of Crypto-Ransomware Using a Pre-Encryption Detection Algorithm. *Computers, 8*(4), 79.

Kotov, V., & Massacci, F. (2013). *Anatomy of exploit kits.* Paper presented at the International symposium on engineering secure software and systems.

Kunwar, R. S., & Sharma, P. (2016). *Malware Analysis: Tools and Techniques.* Paper presented at the Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies.

Kurniawan, A., & Riadi, I. (2018). Detection and Analysis Cerber Ransomware Based on Network Forensics Behavior. *International Journal of Network Security, 20*(5), 836-843.

Kyurkchiev, N., Iliev, A., Rahnev, A., & Terzieva, T. (2019). A New Analysis of Cryptolocker Ransomware and Welchia Worm Propagation Behavior. Some Applications. III. *Communications in Applied Analysis, 23*(2), 359-382.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature, 521*(7553), 436-444.

Lee, J., Lee, J., & Hong, J. (2017). *How to Make Efficient Decoy Files for Ransomware Detection?* Paper presented at the Proceedings of the International Conference on Research in Adaptive and Convergent Systems.

Liao, K., Zhao, Z., Doupé, A., & Ahn, G.-J. (2016). *Behind closed doors: measurement and analysis of CryptoLocker ransoms in Bitcoin.* Paper presented at the 2016 APWG Symposium on Electronic Crime Research (eCrime).

Liu, W., Ren, P., Liu, K., & Duan, H.-x. (2011). *Behavior-based malware analysis and detection.* Paper presented at the 2011 first international workshop on complexity and data mining.

Lynn, B., Prabhakaran, M., & Sahai, A. (2004). *Positive results and techniques for obfuscation.* Paper presented at the International conference on the theory and applications of cryptographic techniques.

Mansfield-Devine, S. (2013). Security review: the past year. *Computer Fraud & Security, 2013*(1), 5-11.

Mansfield-Devine, S. (2017). Ransomware: the most popular form of attack. *Computer Fraud & Security, 2017*(10), 15-20.

McIntosh, T. R., Jang-Jaccard, J., & Watters, P. A. (2018). *Large scale behavioral analysis of ransomware attacks.* Paper presented at the International Conference on Neural Information Processing.

Menahem, E., Shabtai, A., Rokach, L., & Elovici, Y. (2009). Improving malware detection by applying multi-inducer ensemble. *Computational Statistics & Data Analysis, 53*(4), 1483-1494.

Misini, L. (2018). *Etude des Ransomware.* Haute école de gestion de Genève,

Moore, C. (2016). *Detecting ransomware with honeypot techniques.* Paper presented at the 2016 Cybersecurity and Cyberforensics Conference (CCC).

Morato, D., Berrueta, E., Magaña, E., & Izal, M. (2018). Ransomware early detection by the analysis of file sharing traffic. *Journal of Network and Computer Applications, 124*, 14-32.

Mouton, F., Malan, M. M., Leenen, L., & Venter, H. S. (2014). *Social engineering attack framework.* Paper presented at the 2014 Information Security for South Africa.

O'Kane, P., Sezer, S., & Carlin, D. (2018). Evolution of ransomware. *IET Networks, 7*(5), 321-327.

Oktavianto, D., & Muhardianto, I. (2013). *Cuckoo malware analysis*: Packt Publishing Ltd.

Palisse, A., Le Bouder, H., Lanet, J.-L., Le Guernic, C., & Legay, A. (2016). *Ransomware and the legacy crypto API.* Paper presented at the International Conference on Risks and Security of Internet and Systems.

Paquet-Clouston, M., Haslhofer, B., & Dupont, B. (2019). Ransomware payments in the bitcoin ecosystem. *Journal of Cybersecurity, 5*(1), tyz003.

Pascariu, C., & Barbu, I.-D. (2015). Ransomware–an emerging threat. *International Journal of Information Security and Cybercrime, 4*(2), 27-32.

Pathak, P., & Nanded, Y. M. (2016). A dangerous trend of cybercrime: ransomware growing challenge. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), 5*(2), 371-373.

Player, V. (2010). 7.1. VMware.

Pletinckx, S., Trap, C., & Doerr, C. (2018). *Malware coordination using the blockchain: An analysis of the cerber ransomware.* Paper presented at the 2018 IEEE Conference on Communications and Network Security (CNS).

Poudyal, S., Subedi, K. P., & Dasgupta, D. (2018). *A framework for analyzing ransomware using machine learning.* Paper presented at the 2018 IEEE Symposium Series on Computational Intelligence (SSCI).

Prakash, K. P., Nafis, T., & Biswas, D. S. (2017). Preventive measures and incident response for locky Ransomware. *International Journal of Advanced Research in Computer Science, 8*(5), 392-395.

Rabek, J. C., Khazan, R. I., Lewandowski, S. M., & Cunningham, R. K. (2003). *Detection of injected, dynamically generated, and obfuscated malicious code.* Paper presented at the Proceedings of the 2003 ACM workshop on Rapid malcode.

Rajab, M. A., Ballard, L., Marvrommatis, P., Provos, N., & Zhao, X. (2010). The nocebo effect on the web: an analysis of fake anti-virus distribution.

Ramírez-Gallego, S., Lastra, I., Martínez-Rego, D., Bolón-Canedo, V., Benítez, J. M., Herrera, F., & Alonso-Betanzos, A. (2017). Fast-mRMR: Fast minimum redundancy maximum relevance algorithm for high-dimensional big data. *International Journal of Intelligent Systems, 32*(2), 134-152.

Raunak, P., & Krishnan, P. (2017). Network detection of ransomware delivered by exploit kit. *ARPN Journal of Engineering and Applied Sciences, 12*, 3885-3889.

Reddy, D. K. S., & Pujari, A. K. (2006). N-gram analysis for computer virus detection. *Journal in computer virology, 2*(3), 231-239.

Richardson, R., & North, M. M. (2017). Ransomware: Evolution, mitigation and prevention. *International Management Review, 13*(1), 10.

Richet, J.-L. (2016). Extortion on the internet: the rise of crypto-ransomware. *Harvard*.

Roberts, N. (2018). *Ransomware: an evolving threat.* Utica College,

Rogojanu, A., & Badea, L. (2014). The issue of competing currencies.

Saeed, I. A., Selamat, A., & Abuagoub, A. M. (2013). A survey on malware and malware detection systems. *International Journal of Computer Applications, 67*(16).

Scaife, N., Carter, H., Traynor, P., & Butler, K. R. (2016). *Cryptolock (and drop it): stopping ransomware attacks on user data.* Paper presented at the 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS).

Sedano, J., Chira, C., González, S., Herrero, Á., Corchado, E., & Villar, J. R. (2015). *On the selection of key features for android malware characterization.* Paper presented at the Computational Intelligence in Security for Information Systems Conference.

Segal, M. R. (2004). Machine learning benchmarks and random forest regression.

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

Shabtai, A., Moskovitch, R., Elovici, Y., & Glezer, C. (2009). Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey. *information security technical report, 14*(1), 16-29.

Shabtai, A., Moskovitch, R., Feher, C., Dolev, S., & Elovici, Y. (2012). Detecting unknown malicious code by applying classification techniques on opcode patterns. *Security Informatics, 1*(1), 1.

Shukla, M., Mondal, S., & Lodha, S. (2016). *Poster: Locally virtualized environment for mitigating ransomware threat.* Paper presented at the Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security.

Sood, A. K., & Enbody, R. J. (2011). Malvertising–exploiting web advertising. *Computer Fraud & Security, 2011*(4), 11-16.

Tailor, J. P., & Patel, A. D. (2017). A comprehensive survey: ransomware attacks prevention, monitoring and damage control. *Int. J. Res. Sci. Innov, 4,* 2321-2705.

Takeuchi, Y., Sakai, K., & Fukumoto, S. (2018). *Detecting ransomware using support vector machines.* Paper presented at the Proceedings of the 47th International Conference on Parallel Processing Companion.

Tang, A., Sethumadhavan, S., & Stolfo, S. J. (2014). *Unsupervised anomaly-based malware detection using hardware features.* Paper presented at the International Workshop on Recent Advances in Intrusion Detection.

ur Rehman, H., Yafi, E., Nazir, M., & Mustafa, K. (2018). *Security assurance against cybercrime ransomware.* Paper presented at the International conference on intelligent computing & optimization.

Vinod, P., Laxmi, V., & Gaur, M. S. (2012). *Reform: Relevant features for malware analysis.* Paper presented at the 2012 26th International Conference on Advanced Information Networking and Applications Workshops.

Vinod, P., & Viswalakshmi, P. (2018). Empirical Evaluation of a System Call-Based Android Malware Detector. *Arabian Journal for Science and Engineering, 43*(12), 6751-6770.

Wang, T.-Y., Wu, C.-H., & Hsieh, C.-C. (2009). *Detecting unknown malicious executables using portable executable headers.* Paper presented at the 2009 Fifth International Joint Conference on INC, IMS and IDC.

Wang, X., & Karri, R. (2013). *Numchecker: Detecting kernel control-flow modifying rootkits by using hardware performance counters.* Paper presented at the 2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC).

Wyke, J., & Ajjan, A. (2015). The current state of ransomware. *SOPHOS. A SophosLabs Technical Paper*.

Xiao, T., Xia, T., Yang, Y., Huang, C., & Wang, X. (2015). *Learning from massive noisy labeled data for image classification.* Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

Xing, X., Meng, W., Lee, B., Weinsberg, U., Sheth, A., Perdisci, R., & Lee, W. (2015). *Understanding malvertising through ad-injecting browser extensions.* Paper presented at the Proceedings of the 24th international conference on world wide web.

Yang, Y., & Pedersen, J. O. (1997). *A comparative study on feature selection in text categorization.* Paper presented at the Icml.

Yao, H., & Wang, Z.-Y. (2013). Construction of virtualized resource pool based on KVM-QEMU with Libvirt. *Comput. Modernization, 1*(7), 26-29.

Yaqoob, I., Ahmed, E., ur Rehman, M. H., Ahmed, A. I. A., Al-garadi, M. A., Imran, M., & Guizani, M. (2017). The rise of ransomware and emerging security challenges in the Internet of Things. *Computer Networks, 129*, 444-458.

Ye, Y., Wang, D., Li, T., Ye, D., & Jiang, Q. (2008). An intelligent PE-malware detection system based on association mining. *Journal in computer virology, 4*(4), 323-334.

You, I., & Yim, K. (2010). *Malware obfuscation techniques: A brief survey.* Paper presented at the 2010 International conference on broadband, wireless computing, communication and applications.

Young, A., & Yung, M. (1996). *Cryptovirology: Extortion-based security threats and countermeasures.* Paper presented at the Proceedings 1996 IEEE Symposium on Security and Privacy.

Young, A. L. (2005). *Building a cryptovirus using Microsoft's cryptographic API.* Paper presented at the International Conference on Information Security.

Zakaria, W. Z. A., Abdollah, M. F., Mohd, O., & Ariffin, A. F. M. (2017). *The rise of ransomware.* Paper presented at the Proceedings of the 2017 International Conference on Software and e-Business.

Zavarsky, P., & Lindskog, D. (2016). Experimental analysis of ransomware on windows and android platforms: Evolution and characterization. *Procedia Computer Science, 94*, 465-472.

Zhang, H., Xiao, X., Mercaldo, F., Ni, S., Martinelli, F., & Sangaiah, A. K. (2019). Classification of ransomware families with machine learning based on N-gram of opcodes. *Future Generation Computer Systems, 90*, 211-221.

Zhao, J. Y., Kessler, E. G., Yu, J., Jalal, K., Cooper, C. A., Brewer, J. J., . . . Guo, W. A. (2018). Impact of trauma hospital ransomware attack on surgical residency training. *journal of surgical research, 232*, 389-397.

APPENDIX A

The most important ransomware API Calls with category and attack phases

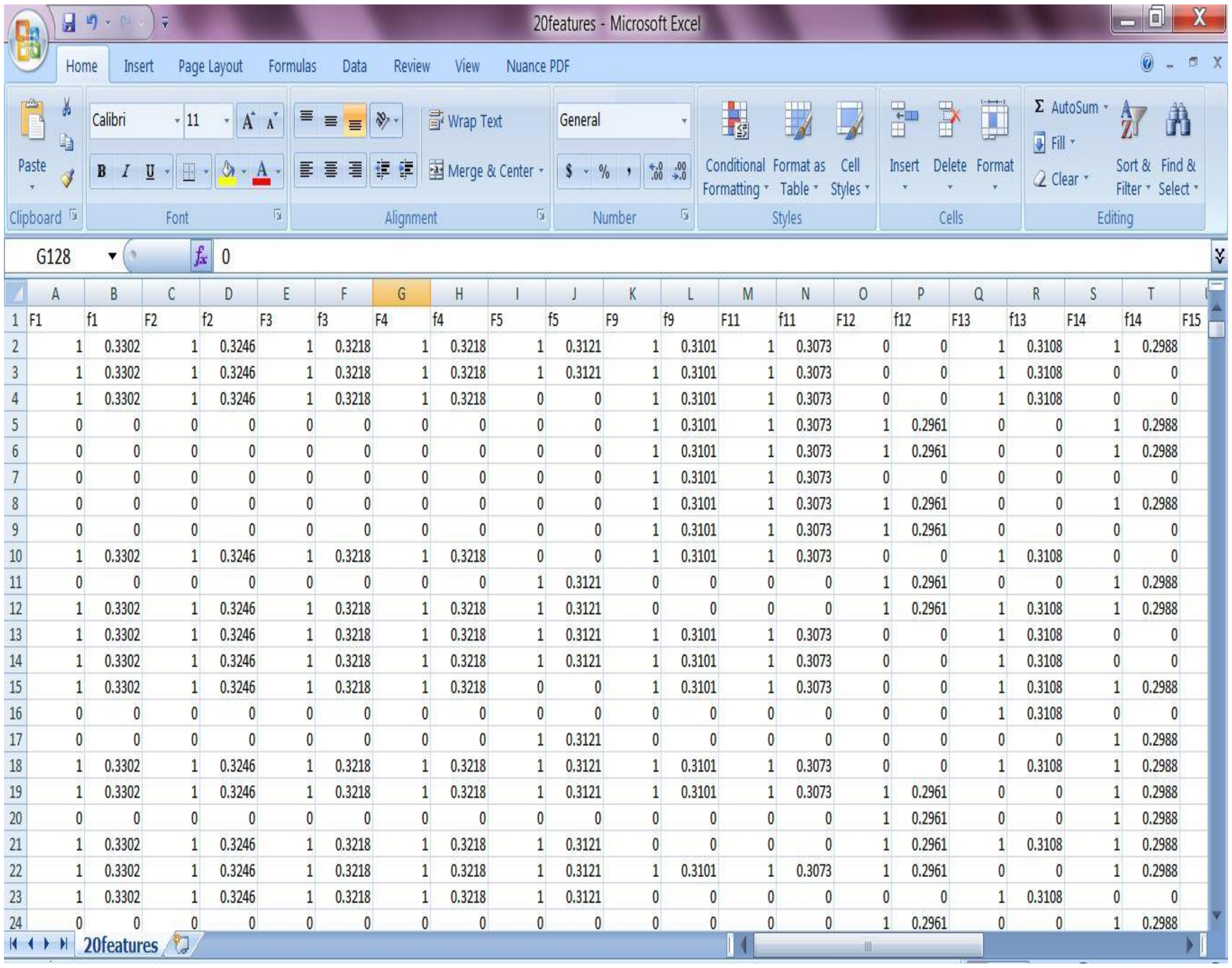
	API Calls	API Category	Attack Stages	Functions
Pre-Encryption	GetComputerNameA	System Information	Exploitation	Retrieves the NetBIOS name of the local computer
	GetNativeSystemInfo	System Information	Exploitation	Gathering the system information like system time and memory status
	GetSystemDirectoryA	System Information	Exploitation	Retrieves the path of the system directory
	GetSystemInfo	System Information	Exploitation	Retrieves information about the current system
	GetSystemWindowsDirectoryW	System Information	Exploitation	Retrieves the path of the shared Windows directory on a multi-user system
	GetUserNameA	System Information	Exploitation	Retrieves the name of the user associated with the current thread
	GetSystemTimeAsFileTime	Time Functions	Exploitation	Retrieves the current system date and time
	GetTimeZoneInformation	Time Functions	Exploitation	Retrieves the current time zone settings
	SetFileTime	Time Functions	Exploitation	Sets the date and time that the specified file or directory was created
	HttpOpenRequestW	WinINet Functions	Contacting	Creates an HTTP request handle
	InternetConnectA	WinINet Functions	Contacting	Opens File Transfer Protocol (FTP) or HTTP session for the attacker's site.
	InternetOpenUrlA	WinINet Functions	Contacting	Opens a resource specified by a complete FTP or HTTP URL
	HttpSendRequestA	WinINet Functions	Contacting	Sends the specified request to the attacker's HTTP server
	InternetReadFile	WinINet Functions	Contacting	Reads data from a handle opened by the InternetOpenUrl, FtpOpenFile, HttpOpenRequest
	WSARecv	Windows Network	Contacting	Receives data from a connected socket or a bound connectionless socket
	WriteProcessMemory	Process and Thread	Execution	To write data to a remote process. Ransomware uses this function as part of process injection.

	CreateRemoteThread	Process and Thread	Execution	Creates a thread that runs in the virtual address space of another process
	NtResumeThread	Process and Thread	Execution	To resume a previously suspended thread
	QueueUserAPC	Process and Thread	Execution	To execute, ransomware uses this function to inject code into another process
	SetThreadContext	Process and Thread	Execution	To modify the context of a given thread.
	GetFileSize	File Management	Execution	The ransomware retrieves the size gathered of the specified file.
	GetFileType	File Management	Execution	Retrieves the file type of the targeted file
	RegNotifyChangeKeyValue	Registry Function	Execution	Runs in loop and monitors if the key is changed
	RegCreateKeyExA	Registry Function	Execution	Creates the specified registry key for encryption
	NtAllocateVirtualMemory	Memory Management	Execution	Creates new region of memory within the virtual address space of the specified process.
	NtCreateSection	Memory Management	Execution	Creates or opens a named or unnamed file mapping object for a specified file
Encryption	CryptEncrypt	Cryptography	Encryption	The CryptEncrypt function encrypts data.
	CryptCreateHash	Cryptography	Encryption	Initiates the hashing of a stream of data
	CryptGenKey	Cryptography	Encryption	Generates a random cryptographic session key or a public/private key pair.
	CryptHashData	Cryptography	Encryption	Adds data to a specified hash object
	CryptAcquireContextW	Cryptography	Encryption	Acquires a handle to a particular key container within a particular cryptographic service provider.
	CryptReleaseContext	Cryptography	Encryption	Releases the handle of a cryptographic service provider (CSP) and a key container.
	CryptGetHashParam	Cryptography	Encryption	Retrieves data that governs the operations of a hash object.
	CryptDecodeObjectEx	Cryptography	Encryption	Decodes a structure of the type indicated by the lpszStructType parameter
	CryptExportKey	Cryptography	Encryption	Exports the victim's public RSA key to new file

	EncryptMessage	Cryptography	Encryption	Encrypts a message to provide privacy
	CertControlStore	Cryptography	Encryption	Notifies when there is a difference between the contents of a cached store and the contents of that store as it is persisted to storage
	FindFirstFileExW	File Operation	Encryption	Searches a directory for a file or subdirectory with a name and attributes that match those specified
	FindNextFileW	File Operation	Encryption	Continues a file search to encrypt
	CreateFileW	File Operation	Encryption	Create opens encrypted files.
	WriteFile	File Operation	Encryption	Encrypts and Writes data to the targeted file.
Post-Encryption	DrawTextExW	Font and Text	Extortion	Draws formatted text in the specified rectangle
	SendNotifyMessageW	Font and Text	Extortion	Sends threatening message to the victim via windows
	LoadStringA	String Functions	Extortion	Loads a string resource from the executable file associated with a specified module.
	WriteConsoleA	Console Functions	Extortion	Writes a threatening character string to a console
	DeleteFileW	File Management	Backup spoliation	Deletes the backup files
	CopyFileA	File Management	Backup spoliation	Copies an existing encrypted back up file to a new file
	MoveFileWithProgressW	File Management	Backup spoliation	Replaces the original file with ones that is encrypted
	SetFilePointer	File Management	Backup spoliation	Moves the file pointer of the specified file
	CryptDecrypt	Cryptography	Release	Decrypts data previously encrypted by the malware

APPENDIX B

SNIPPET OF FEUTURE SELECTION DATASET



G128 0

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	F1	f1	F2	f2	F3	f3	F4	f4	F5	f5	F9	f9	F11	f11	F12	f12	F13	f13	F14	f14	F15
2	1	0.3302	1	0.3246	1	0.3218	1	0.3218	1	0.3121	1	0.3101	1	0.3073	0	0	1	0.3108	1	0.2988	
3	1	0.3302	1	0.3246	1	0.3218	1	0.3218	1	0.3121	1	0.3101	1	0.3073	0	0	1	0.3108	0	0	
4	1	0.3302	1	0.3246	1	0.3218	1	0.3218	0	0	1	0.3101	1	0.3073	0	0	1	0.3108	0	0	
5	0	0	0	0	0	0	0	0	0	0	1	0.3101	1	0.3073	1	0.2961	0	0	1	0.2988	
6	0	0	0	0	0	0	0	0	0	0	1	0.3101	1	0.3073	1	0.2961	0	0	1	0.2988	
7	0	0	0	0	0	0	0	0	0	0	1	0.3101	1	0.3073	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	0	0	0	1	0.3101	1	0.3073	1	0.2961	0	0	1	0.2988	
9	0	0	0	0	0	0	0	0	0	0	1	0.3101	1	0.3073	1	0.2961	0	0	0	0	
10	1	0.3302	1	0.3246	1	0.3218	1	0.3218	0	0	1	0.3101	1	0.3073	0	0	1	0.3108	0	0	
11	0	0	0	0	0	0	0	0	1	0.3121	0	0	0	0	1	0.2961	0	0	1	0.2988	
12	1	0.3302	1	0.3246	1	0.3218	1	0.3218	1	0.3121	0	0	0	0	1	0.2961	1	0.3108	1	0.2988	
13	1	0.3302	1	0.3246	1	0.3218	1	0.3218	1	0.3121	1	0.3101	1	0.3073	0	0	1	0.3108	0	0	
14	1	0.3302	1	0.3246	1	0.3218	1	0.3218	1	0.3121	1	0.3101	1	0.3073	0	0	1	0.3108	0	0	
15	1	0.3302	1	0.3246	1	0.3218	1	0.3218	0	0	1	0.3101	1	0.3073	0	0	1	0.3108	1	0.2988	
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.3108	0	0	
17	0	0	0	0	0	0	0	0	1	0.3121	0	0	0	0	0	0	0	0	1	0.2988	
18	1	0.3302	1	0.3246	1	0.3218	1	0.3218	1	0.3121	1	0.3101	1	0.3073	0	0	1	0.3108	1	0.2988	
19	1	0.3302	1	0.3246	1	0.3218	1	0.3218	1	0.3121	1	0.3101	1	0.3073	1	0.2961	0	0	1	0.2988	
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.2961	0	0	1	0.2988	
21	1	0.3302	1	0.3246	1	0.3218	1	0.3218	1	0.3121	0	0	0	0	1	0.2961	1	0.3108	1	0.2988	
22	1	0.3302	1	0.3246	1	0.3218	1	0.3218	1	0.3121	1	0.3101	1	0.3073	1	0.2961	0	0	1	0.2988	
23	1	0.3302	1	0.3246	1	0.3218	1	0.3218	1	0.3121	0	0	0	0	0	0	1	0.3108	0	0	
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.2961	0	0	1	0.2988	

APPENDIX C

ELIMINATING THE REDUNDANCY OF THE FUNCTION CALLING, C++ COD

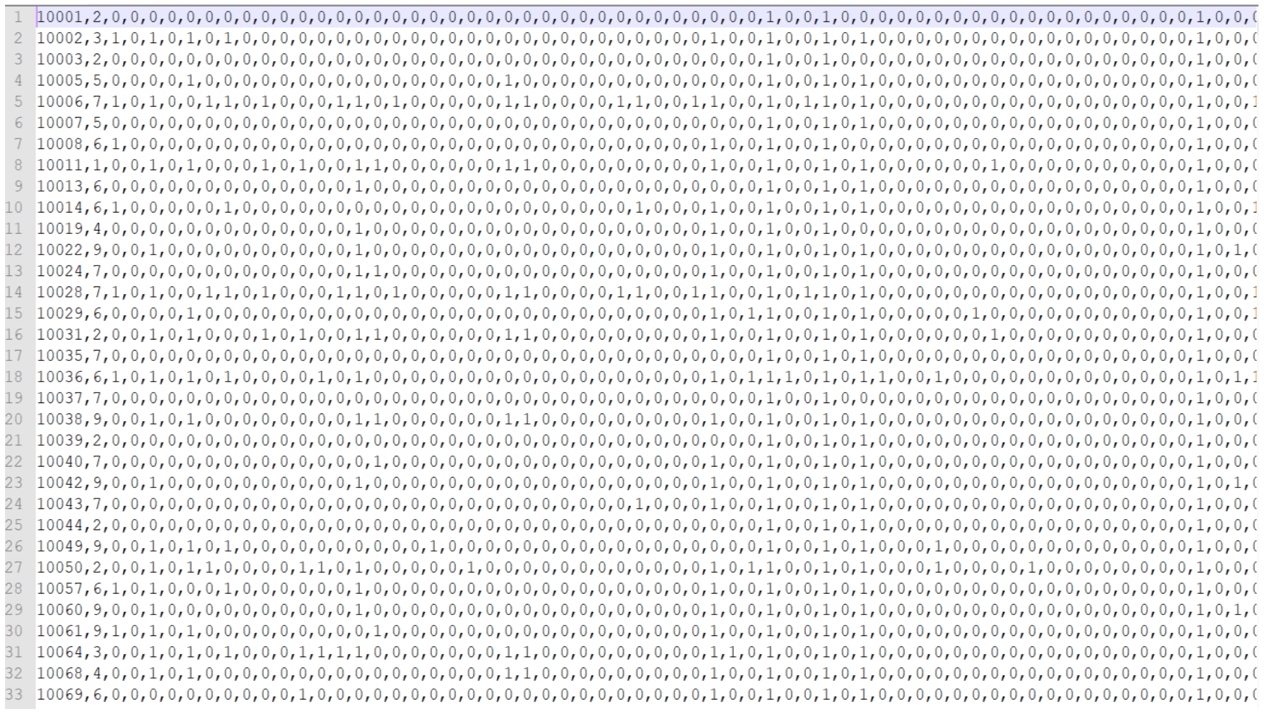
```
#include <stdio.h>   /* required for file operations */
#include <stdlib.h>
#include <string.h>
#include <conio.h>  /* for clrscr */
#include <dos.h>  /* for delay */
FILE *fr,*fw;            /* declare the file pointer */
Main ()
{
    clrscr();
    printf ("\n \n \t\t Program for Eliminating Redundancy has Started," );
    printf ("\n \n \t\t Open Files ," );
   fr = fopen ("All_Benign Data set.txt", "rt");  /* open the file for reading */
   fw = fopen ("All_Benign Data set_output.txt", "wt");
             int i=0;
              int j=0,k=0;
              char line;//,ch;
              char cmd[25];
              char cmd1[25];
              printf ("\n \n \t\t Read file ," );
              // c = fgetc (pFile);   (c != EOF);
               Line =NULL;
             while(line != EOF)
   {
                   Line = fgetc(fr);
                    if(line==',')
                      {
                        cmd[k++]=',';
                       fputs  (cmd,fw);
                     //   printf (" found ," );
                       for (int w=0;w<=25;w++)
                          cmd[w]=NULL;
                          k=0;
               Continue;
             }
        //******************************
             if(line==' ')
             {
                        If (strcmp(cmd,cmd1)== 0)
                             {
```

```
                       For (int w=0;w<=25;w++)
                          cmd[w]=NULL;
                         k=0;
                        Continue;
                }
              Else
                {
                        for (int w=0;w<=25;w++)
                          cmd1[w]=NULL;
                        for (j=0; j<=k; j++)
                         cmd1[j]=cmd[j];

                        cmd[k++]=' ';
                        fputs  (cmd,fw);
                        for (int w=0;w<=25;w++)
                          cmd[w]=NULL;
                         k=0;
                         Continue;
               }
           }
       //*****************************
      cmd[k]=line;
      k++;
  }
fclose(fr);  /* close the file prior to exiting the routine */
fclose(fw);
  printf ("\n \n \t\t Finish copy , %d line has been copyed",i );
getche();  } /*of main*/
```

APPENDIX D

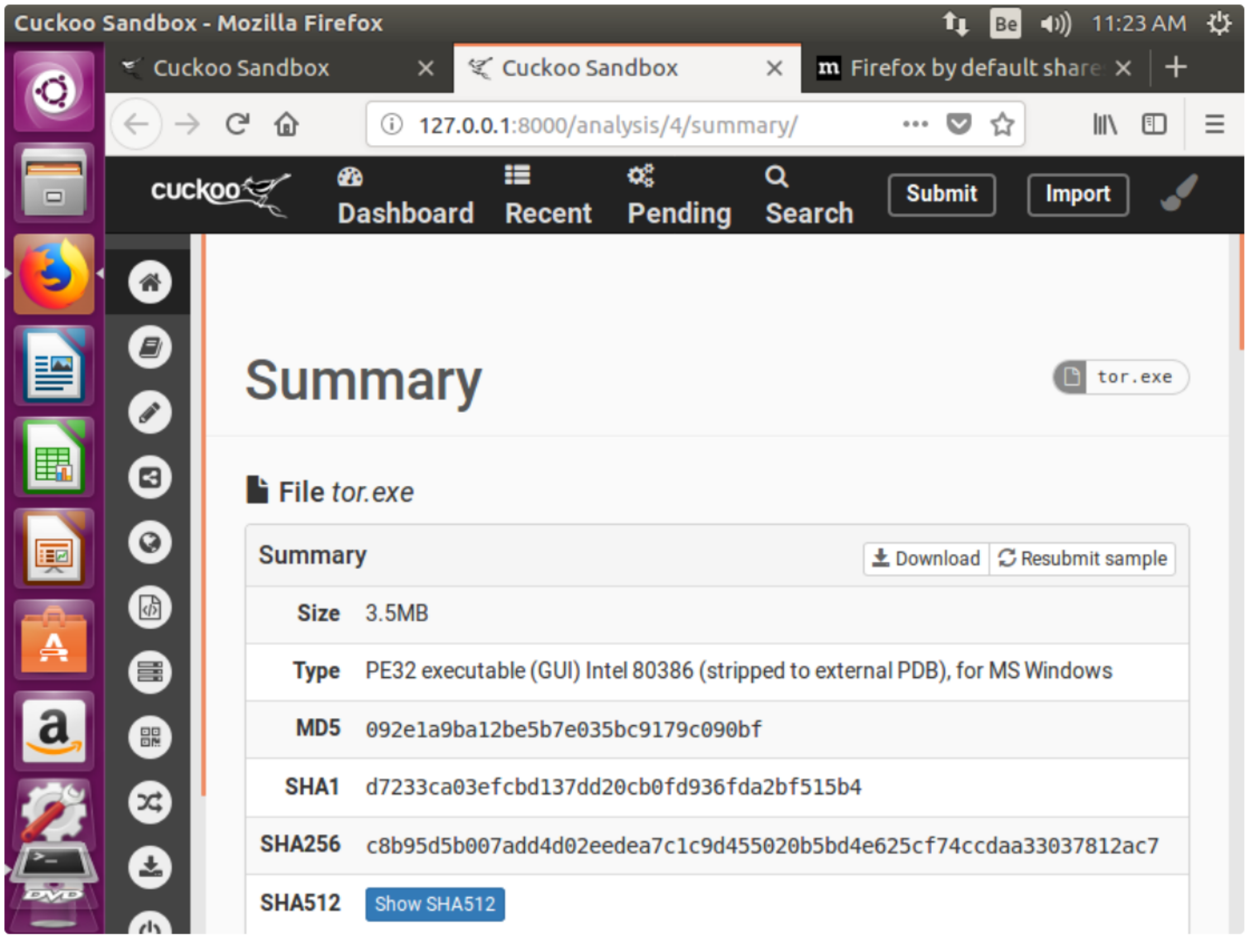
A SNIPPET OF BINARY BENIGN AND RANSOMWARE DATA SET



```
10001,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10002,3,1,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10003,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10005,5,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10006,7,1,0,1,0,0,1,1,0,1,0,0,0,1,1,0,1,0,0,0,0,0,1,1,0,0,0,0,1,1,0,0,1,1,0,0,1,0,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10007,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10008,6,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10011,1,0,0,1,0,1,0,0,0,1,0,1,0,0,1,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10013,6,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10014,6,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10019,4,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10022,9,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1
10024,7,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10028,7,1,0,1,0,0,1,1,0,1,0,0,0,1,1,0,1,0,0,0,0,0,1,1,0,0,0,0,1,1,0,0,1,1,0,0,1,0,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10029,6,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10031,2,0,0,1,0,1,0,0,0,1,0,1,0,0,1,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10035,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10036,6,1,0,1,0,1,0,1,0,0,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,1,0,1,0,1,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1
10037,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10038,9,0,0,1,0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10039,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10040,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10042,9,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1
10043,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10044,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10049,9,0,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10050,2,0,0,1,0,1,1,0,0,0,0,1,1,0,1,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,1,0,1,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0
10057,6,1,0,1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10060,9,0,0,1,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1
10061,9,1,0,1,0,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10064,3,0,0,1,0,1,0,1,0,0,0,1,1,1,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,1,1,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10068,4,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
10069,6,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,1,0,0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0
```

APPENDIX E

CUCKOO SANDBOX HTML REPORT



Summary

File *tor.exe*

Summary

Size	3.5MB
Type	PE32 executable (GUI) Intel 80386 (stripped to external PDB), for MS Windows
MD5	092e1a9ba12be5b7e035bc9179c090bf
SHA1	d7233ca03efcbd137dd20cb0fd936fda2bf515b4
SHA256	c8b95d5b007add4d02eedea7c1c9d455020b5bd4e625cf74ccdaa33037812ac7
SHA512	Show SHA512

APPENDIX F

REMOVING NOISY DATASET USING C++ COD

```
#include <stdio.h>   /* required for file operations */
#include <stdlib.h>
#include <string.h>
#include <conio.h>  /* for clrscr */
#include <dos.h>  /* for delay */
FILE *fr,*fw;           /* declare the file pointer */
 const int size=100;
main()
{
    char line[250],ch;
   clrscr();
   fr = fopen ("Xburstcopy.txt", "rt");  /* open the file for reading */
   fw = fopen ("Xburstcopy.txt", "wt");
   int i=0;
    int j=0,k=0;
   while(fgets(line, 250, fr) != NULL)
   {
       k=0;
    do
    {
          ch= line[k];
          k++;
    } while (ch !='*');
     do
    {
          ch= line[k];
          k++;
    } while (ch !='*');
    int l=0;
    do
    {
          char cmd[25];
          ch= line[k];
          if(ch=='*')
         {
            if ( (strcmp(cmd, "HeapAlloc")== 0)
                || (strcmp(cmd, "HeapFree")== 0)
```

```
              || (strcmp(cmd, "IsBadReadPtr")== 0)
              || (strcmp(cmd, "IsBadWritePtr")== 0)
              || (strcmp(cmd, "LocalAlloc")== 0)
              || (strcmp(cmd, "LocalFree")== 0)
              || (strcmp(cmd, "IsBadStringPtrW")== 0) )
              {
                  ;
              }
             else {
                               i++;
                        fputs  (line,fw);
                              }
         for (int w=0;w<=25;w++)
        cmd[w]=NULL;
          break;
         }
         cmd[l]= line[k];
         l++;
         k++;
    } while (ch !='*');
   }
   fclose(fr);  /* close the file prior to exiting the routine */
   fclose(fw);
     printf ("\n \n \t\t Finish copy , %d line has been copyed",i );
   getche();
 } /*of main*/
```

Curriculum Vitae

Personal Information

Name:	**Yahye Abukar Ahmed**
Mother name:	**Anab Ali Mohamed**
Date of Birth:	**1986**
Place of Birth:	**Mogadishu, Somalia**
Citizenship:	**Somali**
Marital Status:	**Married**
Contact Address:	**E-mail: yahye@simad.edu.so** **Tel: 00905367473020**

Educational Background

2014- Current	**Selçuk Üniversitesi, PhD Candidate in Computer Engineering (Intrusion Detection system)**
2014-2015	**Turkish Language Certificate, GBA:3.58 (Necmetttin Erbakan Üniversitesi)**
2011-2013	**Universiti Teknologi Malaysia (UTM). Master of Computer Science (Information Security) CGPA: 3.71**
2010 – 2011	**SIMAD University. Bachelor Degree: in Information Technology. CGPA: 3.67**
2007 – 2010	**SIMAD University. Post Secondary Diploma: In Information Technology : CGPA: 3.81**
2008	**SIMAD University. CISCO IT Essential I**
2007	**Horyal Computer institute: Computer Training.**
2006-2007	**Immamu shafi'i School Secondary School**

Honors and Awards

Won first class award for recognition of high performing student in SIMAD University.

Publications

A) Papers

- **Sh.Sharmeen, Y. Ahmed, Sh.Huda, B.Kocer and M.Hassan, "Avoiding future digital extortion through robust protection against ransomware threats using deep learning based adaptive approaches,"** ***IEEE Access*****, vol.**
- **Y. Ahmed, B.Kocer and B.Al-irmy , "Automated Analysis Approach for the Detection of High Survivable Ransomware,"** ***KSII TRANSACTIONS ON INTERNET AND INFORMATION SYSTEMS,*** **vol. 14, Nov. 2019.**
- **Y. Ahmed, M. Maarof, F.Hassan and M. Abshir, "Survey of keyloggers Technologies,"** ***International Journal of Computer Science and Telecommunications,*** **vol. 5, Feb. 2014.**
- **Y. Ahmed and A.Abdullahi, " Mitigating of Malicious Insider Keylogger Threats,"** ***JISRI. Journal of Research and Innovation in Information Systems,*** **vol. 3, Jan. 2013.**
- **Y. Ahmed and B.Kocer, " Supervised Machine learning Approach for Detection of malicious executables,"** ***International Conference on Mathematics, Science and Engineering Education***
- **Y. Ahmed, B.Kocer, Sh.Huda, and M.Hassan, "A System Call Refinement-Based Enhanced Minimum Redundancy Maximum Relevance Method for Ransomware Early Detection" Journal of Network and Computer Applications" June 15, 2020**

B) Articles

- **Security Issues in Mobile Banking Service: (Deeqtoon), published in Somali Business Review (SBR)**
- **The Impact of Emerging ICT Industries on Small-Business Performance In Somalia, published in Somali Business Review (SBR)**
- **The Challenges of Social Network on Small Business Performance in Somalia, published in Somali Business Review (SBR)**
- **The Effect of Emerging EPOS Technology on Customer Service in Retailer Shops in Somalia, published in Somali Business Review (SBR)**

Work Experiences

19/12/017-05/09/2018	**EMFA Software and Consultancy**
2/11/2016-15/3/2017	**Asylum-Seekers and Immigrants and Solidarity Association (Volunteer)**
2015-2016	**Research assistance at Mevlana University**
2013-2014	**Former head of computer Science Department and lecturer at SIMAD UNIVERSITY**
2013-2014	**Sofa software developer (ICT)**
2013-2014	**Lecturer at UNIVERSITY OF SOMALIA (UNISO)**
2009– 2010	**Lecturer at SOMALI C0MPUTER INSTTTUTE (SOCOMIN).**
2008– 2009	**Teacher in Hilaal Primary and Secondary school.**

Seminars and Workshops attended

4-6 January 2013	**UTM ISC International Student Activity Management Workshop, Mersing, JB Malaysia**
7-8 October 2012	**Matlab workshops Held in Faculty of Computer Science and Information System, UTM**
December, 2011	**Navigate Your Career Held in UTM**
12-13, May 2012	**Adobe Flash Training Held in Faculty Of Computer Science and Information System, UTM**

9 March 2011	**The Role of knowledge in building Society held by UTM-ISC-Somalia**
Aug – Oct, 2007	**Web Developing Training by Sofa Software Technology, Held in Mogadisho.**
December, 2014	**How to Write an Effective Academic Report as Facilitator in UNIVERSITY OF SOMALIA (UNISO)**
Aug, 28, 2015	**Ethical Hacking Seminar as Facilitator in SIMAD UNIVERSITY**
December 25, 2013	**workshop on research methods in SIMAD UNIVERSITY**
July 23, 2013	**Training on Pedagogy, Curriculum Review and Examination quality Assurance by Makerere University Business School in Uganda**

Skills

Security	**information security, security policy**
IT Professional	**Web Developing, Communications, Networking and Programming.**
Computer	**All Office applications**

Languages

Somali	**Mother Tongue**
English	**Fluently**
Turkish	**very good (with certificate)**
Arabic	**Good**

Referee

NAME	**JOB**	**ADDRESS**
Prof. Dr. Bekir KARLIK	**Selçuk Üniversitesi Bilgisayar Müh. Bölümü**	
Fuad Mirre	**Dean, Faculty of Computer Science and Technology in SIMAD**	