Detection and Prediction of Locker Ransomware

By

Ayesha Mansha

00000317970

MS-IS - 2019

Supervisor

Dr. Sana Qadir

Department of Computing

School of Electrical Engineering and Computer Science (SEECS)

National University of Sciences and Technology (NUST)

Islamabad, Pakistan

October 2022

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS/MPhil thesis entitled "Detection and Prediction of Locker Ransomware" written by AYESHA MANSHA, (Registration No 00000317970), of SEECS has been vetted by the undersigned, found complete in all respects as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS/M Phil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

Signature: Sana Qadir

Name of Advisor: Dr. Sana Qadir

Date: 10-Oct-2022

HoD/Associate Dean: ______________________

Date: ______________________

Signature (Dean/Principal): ______________________

Date: ______________________

Approval

It is certified that the contents and form of the thesis entitled "Detection and Prediction of Locker Ransomware" submitted by AYESHA MANSHA have been found satisfactory for the requirement of the degree

Advisor : Dr. Sana Qadir

Signature: Sana Qadir

Date: 10-Oct-2022

Committee Member 1:Dr. Mehdi Hussain

Signature: [illegible]

11-Oct-2022

Committee Member 2:Dr. Dr Hasan Tahir

Signature: [illegible]

Date: 11-Oct-2022

Signature: ______________________

Date: ________________________

Dedication

This thesis is dedicated to my *beloved Mama Jaan & Papa Jaan*, who believed in me even when I did not. And to my younger self, who would be super surprised to know I chose the computing field.

Last but not least, I'd like to dedicate it to my best friends, without whom this thesis would have been completed a year earlier.

Certificate of Originality

I hereby declare that this submission titled "Detection and Prediction of Locker Ransomware" is my own work. To the best of my knowledge it contains no materials previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any degree or diploma at NUST SEECS or at any other educational institute, except where due acknowledgement has been made in the thesis. Any contribution made to the research by others, with whom I have worked at NUST SEECS or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except for the assistance from others in the project's design and conception or in style, presentation and linguistics, which has been acknowledged. I also verified the originality of contents through plagiarism software.

Student Name: AYESHA MANSHA

Student Signature: [illegible]

Acknowledgment

First of all, I would like to express my sincere gratitude to my supervisor, Dr. Sana Qadir, for believing in me and her continuous guidance and support from inception of this idea to its completion. This work would not have been possible without her.

I am also thankful to my Guidance and Evaluation Committee, Dr. Hasan Tahir and Dr. Mehdi Hassan for providing valuable insights and constructive feedback during this research work.

Contents

List of Figures

List of Tables

List of Abbreviations

ACC	Accuracy
API	Application Programming Interface
ANN	Artificial Neural Network
AV	AntiVirus
BERT	Bidirectional Encoder Representations from Transformers
C&C Server	Control and Command Server
CTB Locker	Curve Tor Bitcoin Locker
DeepAMD	Detection and identification of Android malware using high-efficient Deep Artificial Neural Network
FDR	Fasle Discovery Rate
FN	False Negative
FNR	False Negative Rate
FP	False Positive
FPR	Fasle Positive Rate
FSOWA	Fibonacci-based Super-increasing Ordered Weighted Averaging
HIPS	Host Intrusion Prevention System
LRDPA	Locker Ransomware Detection and Prediction Algorithm
MBR	Master Boot Record

MCC	Matthews Correlation Coefficient
MI	Mutual Information
ML	Machine Learning
NPV	Negative Predictive Value
PEELER	Profiling Kernel Level Events to Detect Ransomware
RAAS	Ransomware as a Service
RAMD	Registry Based Anomaly Malware Detection
RDP	Remote Desktop Protocol
RegKey	Registry Key
RF	Random Forest
SHA256	Secure Hashing Algorithm 256
SMB	Server Message Blocks
TN	True Negative
TNR	True Negative Rate
TP	True Positive
TPR	True Positive Rate
VPN	Virtual Private Network
VS	Virus Share
VT	Virus Total

Abstract

The number of incidents involving ransomware has reached an alarming level. Organizations worldwide have suffered financial loss as a result of having their data encrypted by this type of malware. Some organizations have had no choice but to pay exorbitant sums to obtain the decryption key and restore access to their data. Others have not been so fortunate and have had their private data published online, deleted, or left permanently inaccessible. One type of ransomware, called the locker ransomware, has a slightly different mode of operation. Instead of encrypting the victim's data, it locks the victim's system or files. A ransom is demand in return for restoration of access. In order to address the threat posed by locker ransomware, we propose a simple and automated approach for their detection and prediction. We collected and analysed behaviour of locker ransomware and benign software in a sandbox environment. The APIs called and the registry keys triggered were recorded. The data was then pre-processed, refined, and compiled into a dataset. The Locker Ransomware Detection and Prediction Algorithm (LRDPA) is then implemented. This algorithm contained two tiers. First tier implemented static detection by comparing the hash digest of a suspect application with those stored in the signature database. This enabled quick and accurate detection of known locker ransomwares. The second tier implemented prediction and comprised of a Machine Learning (ML) model trained using dynamic behavioural data contained in the dataset. This data consisted of 275 APIs called and the 21,780 Registry keys triggered. The data was then fed to the RF algorithm with 10 fold cross validation. The resulting LRDPA model was evaluated using several metrics. To the best of our knowledge, its accuracy of 99.44% is higher than any existing single ML model-based study. In future, the performance of LRDPA can be improved with the expansion of the dataset and implementation of additional feature selection. **Keywords:** *malware, ransomware, locker, api calls, registry keys*

Chapter 1

Introduction

1.1 Ransomware

In the recent past, there has been an explosion in the prevalence of a kind of malicious software known as ransomware. In its most basic form, ransomware works by encrypting the data on a target machine and then demanding money in exchange for the key to decode the data. In 1989, the first ransomware that became well recognised was called Cyborg. Since 2012, there has been a meteoric rise in the amount of ransomware attacks carried out against businesses. [4] [5]. According to *Fortinet's 2021 Ransomware Report*, ransomware attacks increased by 1,070% from July 2020 to June 2021 [6]. In the year 2021, approximately 37% of the organisations around the globe were targeted by a ransomware attack [7]. A free ransomware identification tool estimates that there are 1,070 distinct types of ransomware [8]. There are a few factors behind this escalation of ransomware attacks. The first is the *pseudo-anonymity* provided by the use of crypto-currencies for payment of ransom. The second factor is the use of *strong encryption* (e.g. elliptic curve cryptography). Both these factors combine to make the source of the ransomware attack extremely difficult to locate and the payment impossible to trace. Furthermore, the rapidly evolving techniques and targets used by adversaries exacerbate the situation [9].

The most obvious impact of a ransomware attack is the unavailability of data to legitimate users. This can be temporary or permanent. For organisations, even the temporary loss of access to valuable or critical data leads to financial loss and reputational damage. In order to recover access to their data, many people who have been hit by ransomware attacks are forced to pay the demanded sum. This is particularly true for victims who do not make use of cloud storage or other means of off-site data backup. When victims of ransomware attacks do not pay the demanded amount of money, the ransomware's creators may disclose the victims' private data (including trade secrets) on the internet or the dark web.

1.2 Types of Ransomware

On the basis of their methods of operation and its intended victims, ransomware may be divided into a number of distinct groups. In a report by Deloitte, ransomware is classified it into two main types: *Lockers* (ransomwares that lock systems) and *Cryptos* (ransomwares that encrypt system data) [10]. In the case of Lockers, a ransom is demanded to unlock the system while in the case of Cryptos, a ransom is demanded to decrypt the data.

However, in the malware community, the general consensus is that there are mainly three major categories of ransomware as shown in figure 1.1.

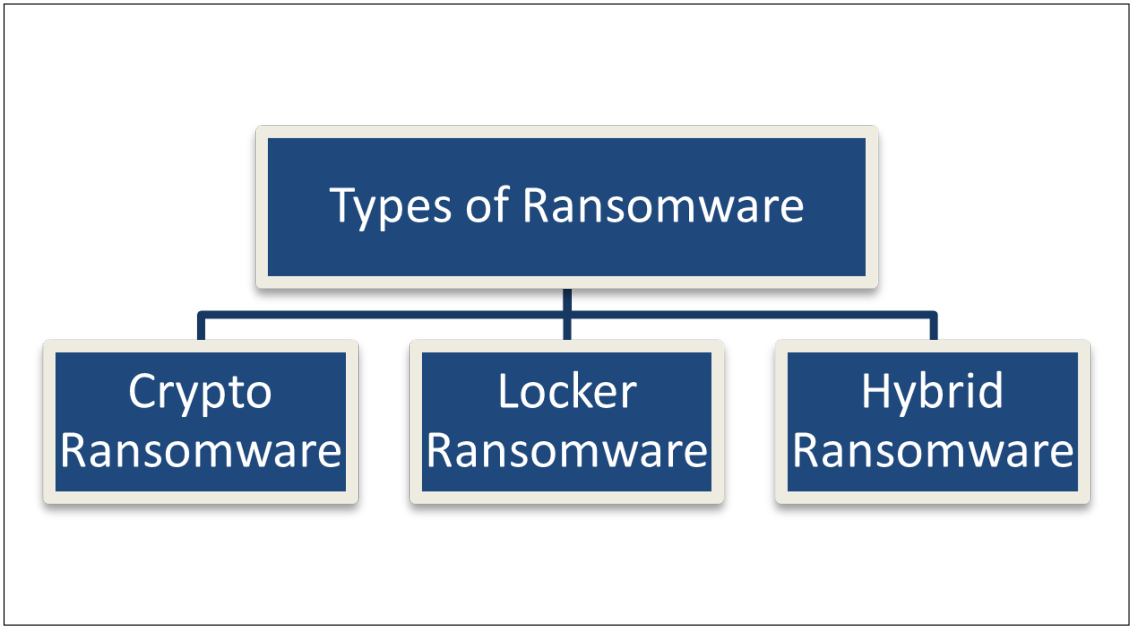


Figure 1.1: Types of Ransomware

1.2.1 Crypto Ransomware

The modus operandi of crypto ransomware is to encrypt sensitive data (such as documents, images, and videos). Generally, this type of malware does not tamper with the essential functionality of a system [11]. In other words, users are able to interact with their systems and are usually able to see their files in a directory listing. However, they are not able to access them. Also, in most cases a countdown timer with a demand for ransom is displayed to the victim.

Examples of popular crypto ransomwares include:

- *TeslaCrypt* (2015) - files on the local computer are encrypted, and a ransom is demanded in return for the key to decode the data. This key is required to restore access to the encrypted files.
- *Ryuk* (2018) - is a ransomware that targeted large, publicly accessible Microsoft Windows computers. Typically, Ryuk encrypts data on a victim machine and demands payment in Bitcoins.
- *Egregor* (2020) - this ransomware is known for its cruel, but very effective, double-extortion tactics. The adversary gets into sensitive data and encrypts it so that the victim is not able to access it.

1.2.2 Locker Ransomware

Locker ransomwares are designed to lock a victim's system or files. Once locked, the system is usually not able to provide basic functionality to a user [12]. A lock screen appears demanding ransom for the password to unlock the system or files. Additionally, the locker ransomware may partially disable the mouse and keyboard.

One interesting sub-type of locker ransomware is the Master Boot Record (MBR) locker ransomware. This ransomware can permanently remove the original MBR file, replace the original MBR with a fake MBR file, or encrypt the original MBR file [13].

Examples of popular locker ransomwares include:

- *Petya* (2016) - encrypts files and overwrites the MBR. This means the operating system is not able to start.

- *CryptoLocker* (2013) - encrypts operating system files (with a specific extension) and makes them unavailable.
- *GoldenEye* (2017) - encrypts files and then appends an 8-character extension to each encrypted file. It then alters the MBR as well.

1.2.3 Hybrid Ransomware

This category of ransomwares have features of locker and crypto ransomwares. Encrypting files and locking users out of their system or files makes this a category notoriously dangerous. The appearance of Ransomware as a Service (RAAS), often known as ransomware assaults, has led to a rise in the frequency of this specific sort of ransomware attack [14].

The emphasis of this thesis is locker ransomware. This choice was taken because locker ransomwares have received relatively less attention and wreak more harm than crypto ransomware. There is generally no option for recovery if the MBR is replaced, removed, or encrypted.

1.3 Top Risks

The consequences of a ransomware attack can range from moderate (e.g. financial loss, data loss) to extensive (e.g. complete shut down of a company). In a survey conducted by Fortinet, 62% of the organisations reported *data loss* as the top concern in a ransomware attack (see figure 1.2) [6]. A significant loss of data can take months or years to recover from and is more serious than *loss of productivity* (38%) or *disruption of operations* (36%).

1.4 Infection Vector

It is generally agreed that a lack of basic user-level cyber-hygiene is the most common enabler of ransomware attacks. Specifically, *phishing* is the frequently used infection vector (see figure 1.3). Generally, this phishing email appears to be from legitimate source and contains an attachment or a link to a website designed to obtain the credentials of

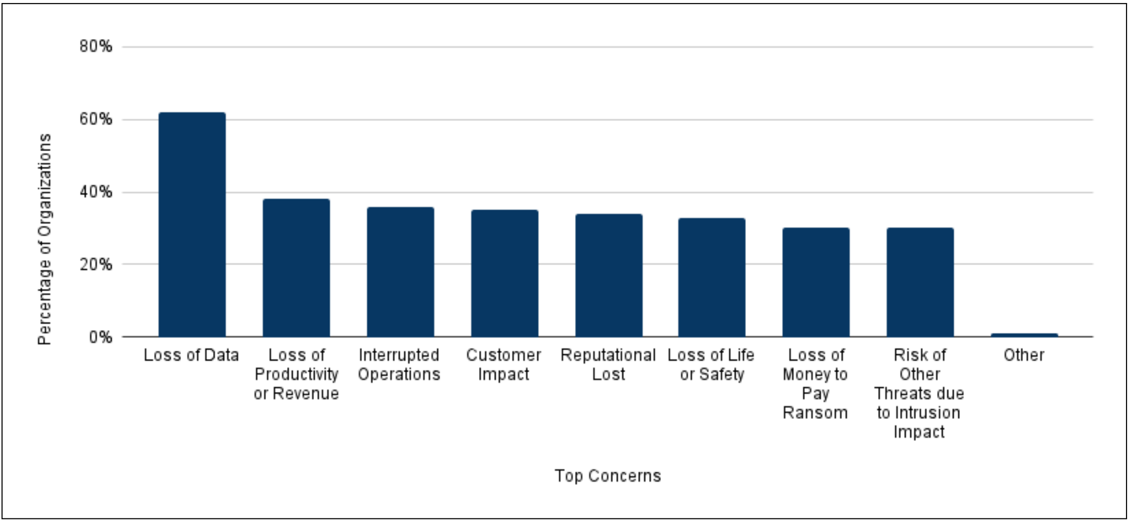


Figure 1.2: Ransomware Attacks: Top Concerns [1]

a legitimate user [15].

The second most common infection vector is through the *Remote Desktop Protocol* (RDP). This protocol is used access a remote system over a network. With the correct credentials, a user is able to obtain full access. This makes RDP is a prime target for malicious actors [16]. A study on the *WannaCry* ransomware found that it exploited vulnerabilities in Microsoft's RDPv3 and SMB (Server Message Blocks) and spread through phishing emails and pharming websites [17]. If the Microsoft patches and updates for Windows platform had been properly tested, it would prevented the significant outbreak of *WannaCry* .

The *exploitation of vulnerabilities* is the third most common infection vector and includes attacking systems (e.g. Web servers and VPN servers) that have not been patched properly or are missing the latest patches. This is important for Websites that integrate plugins and libraries because contemporary software supply chains allow processes to link to many services and functionalities - any of which might be used as an infection vector [18].

Finally, some attacks occur through *account takeover*. This can be done in several ways including downloading the ransomware as a payload by another malware or a drive-by download [2][19].

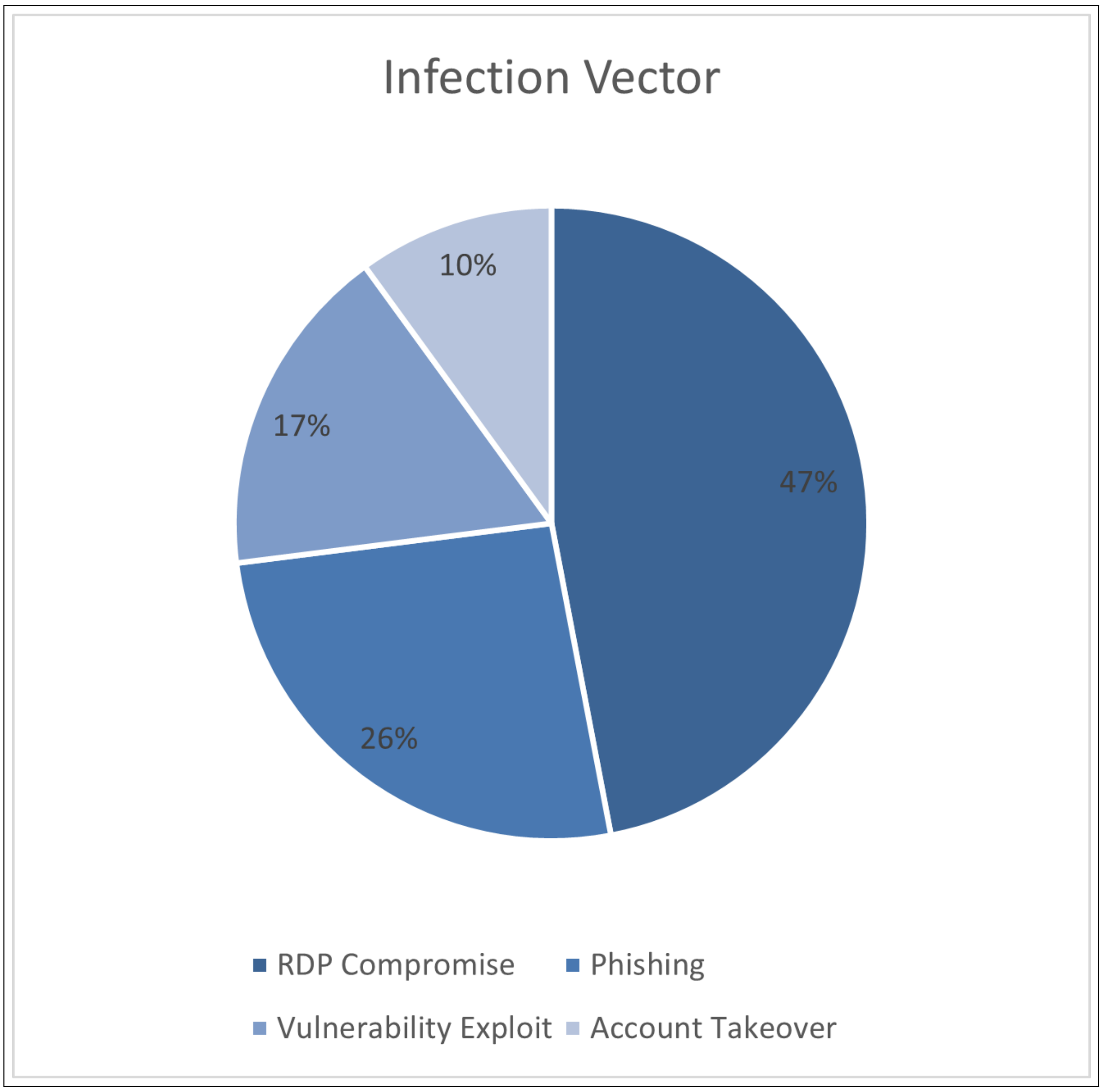


Figure 1.3: Infection Vectors [2]

1.5 Payment Method

As mentioned before, the purpose of almost all ransomware is to obtain monetary benefit from the victim in return for restoring access to their data or system. In older ransomware attacks, bank accounts, prepaid cards, money transfer, or e-wallets were used as payment methods. However, law enforcement authorities are able to track down individuals using these tactics. With the rise of crypto-currency, the ability to conduct anonymous transactions became possible. Ransomwares developers quickly adopted crypto-currency for payment because it provides instant transfer without involving a central banking system or authority [20]. For crypto ransomware, the preferred method of payment is Bitcoin although other crypto-currencies like Ethereum and Monero are also used. Additionally, Bitcoin makes it difficult to differentiate between legitimate transactions and fraudulent ones in the history of Bitcoin transactions [21]. Figure 1.4 shows how the victim of a ransomware attack sends Bitcoins to the attacker in return for the decryption key.

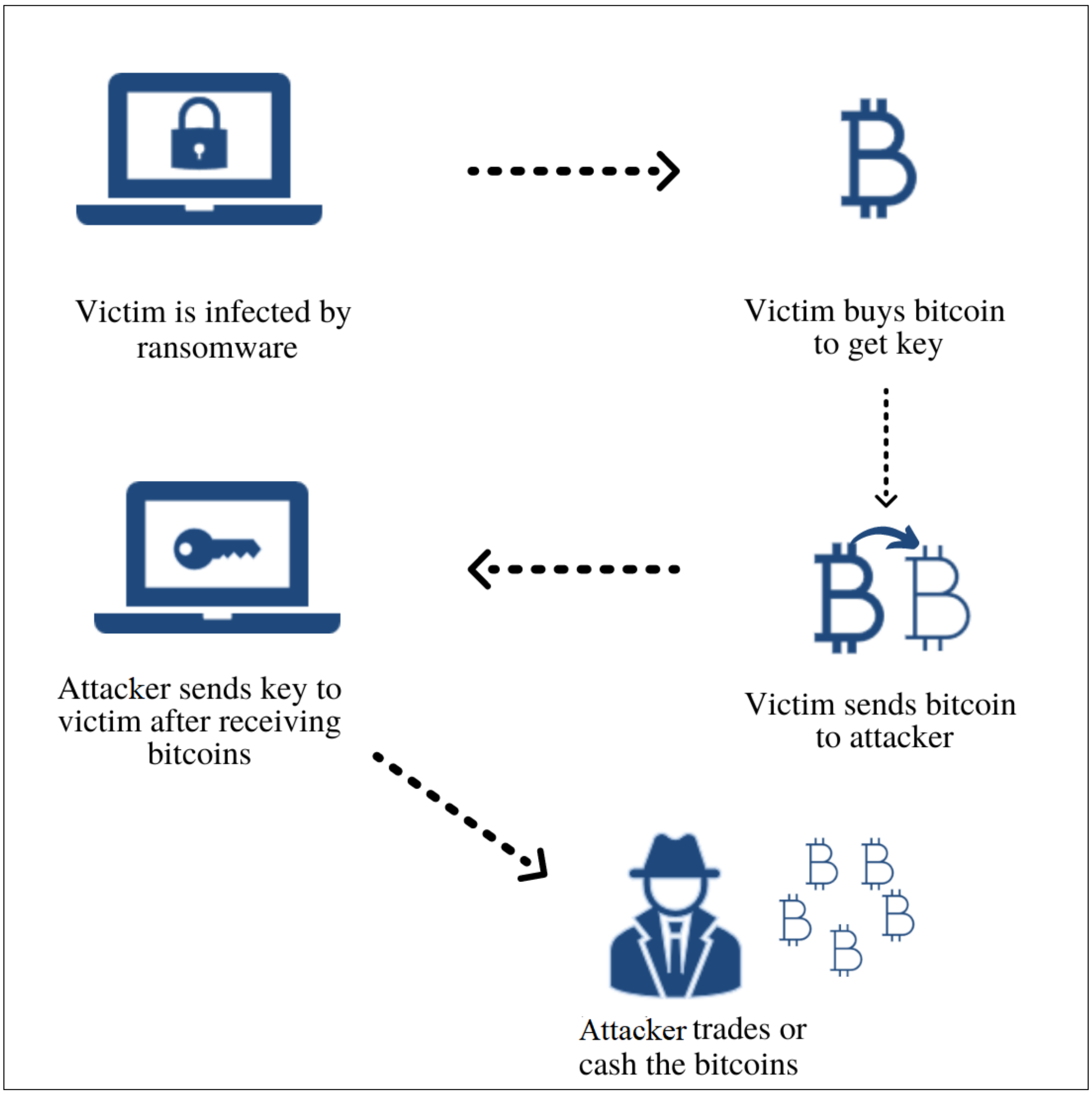

Victim is infected by ransomware
Victim buys bitcoin to get key
Attacker sends key to victim after receiving bitcoins
Victim sends bitcoin to attacker
Attacker trades or cash the bitcoins

Figure 1.4: Ransom Payment Process using Bitcoins

1.6 Life Cycle

Every malware has a different life cycle based on its infection vector, target, and behavior [22]. In order to analyse ransomware, it is essential to understand the phases of its life cycle.

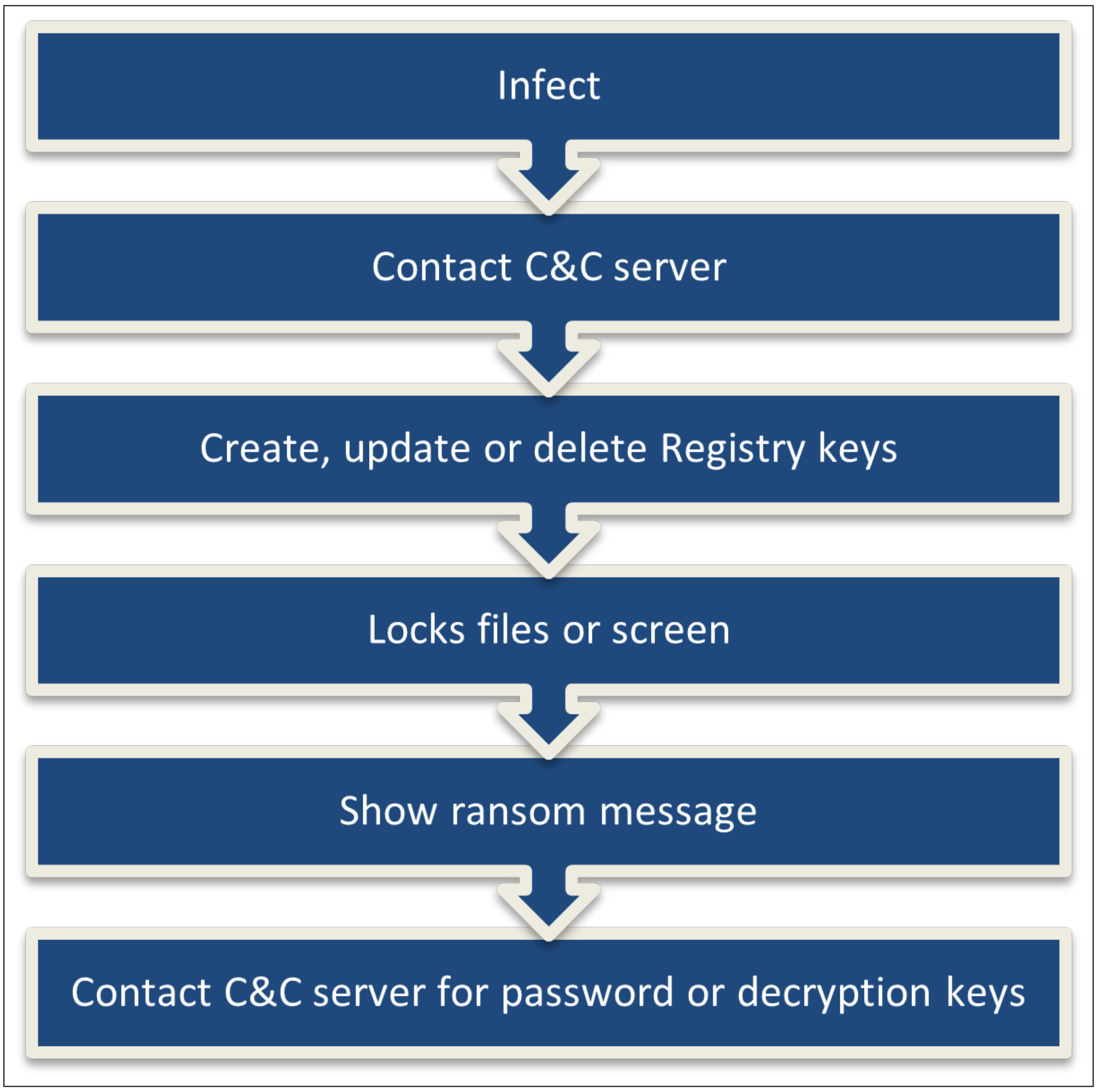

Infect
Contact C&C server
Create, update or delete Registry keys
Locks files or screen
Show ransom message
Contact C&C server for password or decryption keys

Figure 1.5: Life Cycle of Locker Ransomware

Ransomware life cycle commences with the infection of target system and concludes when the victim is asked to pay the ransom. The phases of the *locker* ransomware are slightly different as illustrated in figure 1.5. A target is often infected with locker ransomware through a code dropper, by opening a malicious email attachment, or by downloading it from an infected Website [23]. Once it has entered a target system, the locker ransomware performs operations such as generating a unique computer identity, blocking certain applications, enabling certain programs to run at boot time, and contacting the Command and Control (C&C) server. C&C servers are servers that malware contacts in order to save stolen information or to download additional code or instructions. Adversaries setup C&C servers to help them make their way further inside a system and to strengthen their foothold in networks. After this step, the locker ransomware generally creates, modifies, or deletes files and Registry keys. In the next step, the user is locked out of the system. Then the ransom message is displayed. The final step, which takes place only after the victim pays the ransom, is to download the decryption key or password from the C&C server.

1.7 Problem Statement

Due to its architecture and widespread usage, the Windows platform is a prime target for malware infections. Windows executable files and dynamic link libraries (DLLs) account for 95% of all discovered ransomware. In line with these observations, both academia and industry have developed a variety of methods to detect and predict ransomware. These techniques are based on features gleaned from static and behavior analyses of ransomware samples. This set of features represent a profile of the ransomware that is different from that of benign software.

It is essential to be aware that the *locker ransomware* strain of ransomware has garnered a very little amount of attention from cybersecurity professionals. This thesis intends to address this vacuum by investigating how both static and behavior analysis may be used to construct a profile that can be used to detect and forecast locker ransomware. Specifically, this thesis will look into how these analyses can be employed.

1.8 Research Objective

The objectives of this research are:

1. To analyse the behavior of locker ransomware samples on a Windows system. This is carried out by running samples of locker ransomware and examining the APIs called and the Registry keys triggered.
2. To create a dataset consisting of:
    (a) hash digests of benign and locker ransomware samples. This is to aid in the static detection of locker ransomware.

   (b) APIs called and Registry keys triggered by benign and locker ransomware samples. This is to aid in the dynamic detection of locker ransomware.

3. To design a machine learning algorithm based model for the locker ransomware detection using the dataset that was developed in step (2), which will be used as a starting point.

1.9 Thesis Organisation

This thesis starts with the introduction in chapter 1. It then presents the literature review in chapter 2. This chapter also covers methods for malware detection and analyses existing work on detection of crypto ransomware and locker ransomware. Chapter 3 provides an outline of the methodology and includes details of the methods and tools used to analyse samples and compile the dataset. Chapter 4 presents the developed machine learning-based model and chapter 5 presents its evaluation. The thesis concludes in chapter 6 by summarising the findings and highlighting potential future work.

Chapter 2

Literature Review

The numerous incidents involving ransomware has motivated several studies on individual samples of ransomware and also on families of ransomware. The long-term goal of these studies is to aid the design of effective detection and prevention solutions. Two useful examples of such studies are [24] and [25]. The former study focuses on the behaviour of *CTB Locker* ransomware while the latter study thoroughly analyses four prominent ransomware variants — *CryptoWall*, *TorrentLocker*, *CTB Locker*, and *TeslaCrypt*. Such studies, though useful, are not sufficient. A general approach, applicable to all ransomware is required. For the most effective method, it is widely agreed that the use of Machine Learning (ML) techniques should be utilised. This is done in order to identify malicious software.

2.1 Machine Learning (ML)

Machine learning (ML) is a sub-field of artificial intelligence (AI) that allows computers to *learn* (without being explicitly programmed) from data using statistical models and algorithms. These algorithms identify patterns in the data and then learn from those patterns in order to make predictions. Essentially, the ML models acquire knowledge through experience and improve over time.

There are essentially two main decision-making principles used by machines learning algorithms and they are used to classify ML algorithms into either *Unsupervised* or *Supervised* ML.

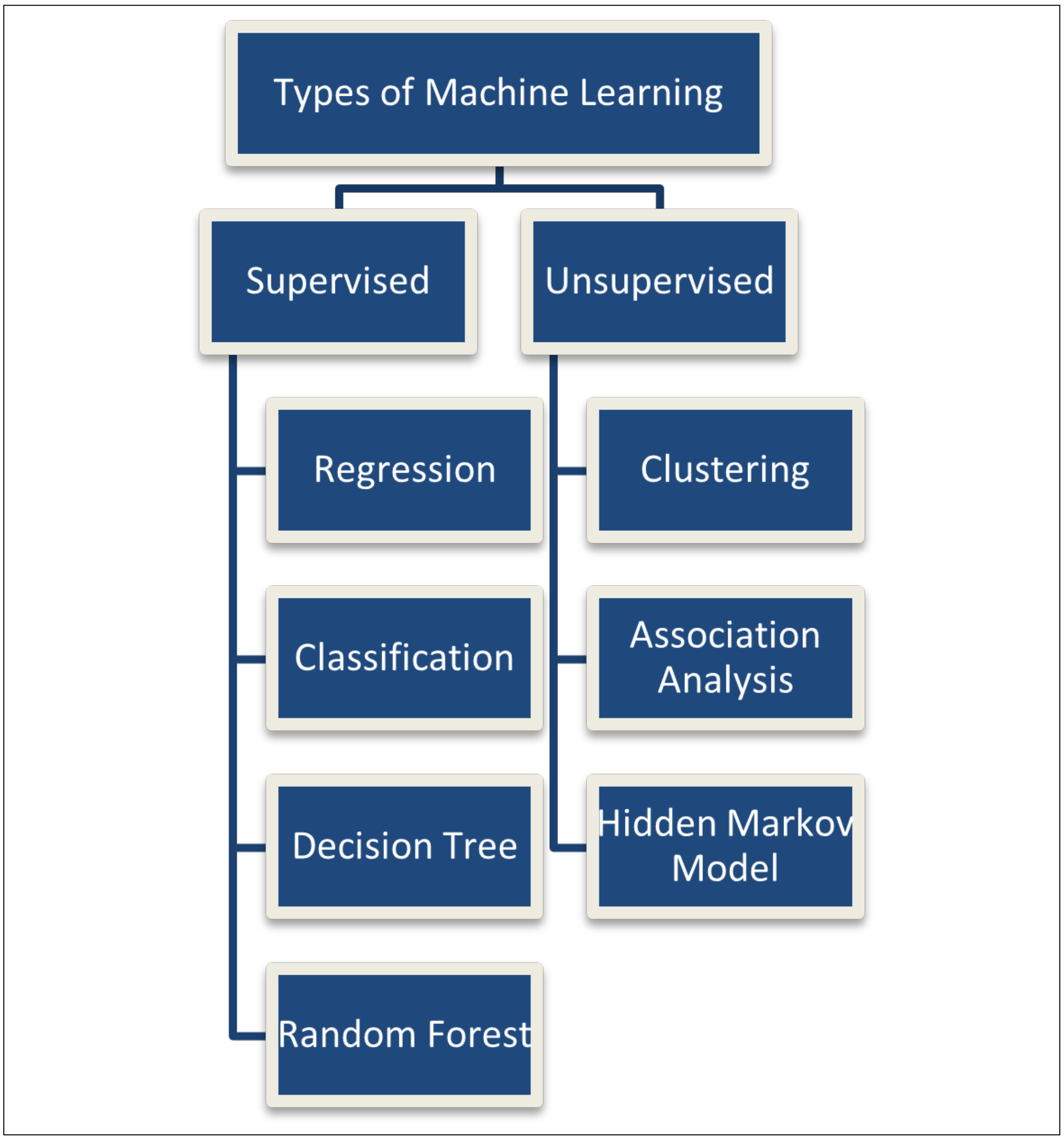


Figure 2.1: Types of Machine Learning

2.1.1 Unsupervised Machine Learning

Unsupervised machine learning algorithms look for insights and connections in data that has not been tagged. In this scenario, models are provided with input data, but the anticipated consequences are not known; hence, they are required to form inferences based on circumstantial evidence without being given any instruction or training. Because the models are not provided with instruction on the *correct answer*, they are required to independently identify patterns.

Clustering, which entails grouping related data, is one of the most used forms of unsupervised learning. This technique is typically used for exploratory research and may assist in uncovering hidden trends or patterns.

2.1.2 Supervised Machine Learning

When making predictions using labelled training data, supervised machine learning techniques and models are used. Every training sample has a corresponding input to go along with the predicted result. A supervised learning algorithm analyses the sample data and develops an inference by doing so. This inference is simply an educated estimate about the labels that should be applied to data that has not yet been observed.

This is the way of machine learning that is used the most often and extensively. It is supervised due to the fact that these models have to be trained using example data that has been manually annotated. The data are labelled in order to specify what patterns (similar words and pictures, data categories, etc.) it should identify and which associations it should recognise.

Supervised machine learning is further divided into two types: *classification* and *regression.*

- Regression - The result that may be anticipated from using regression is a continuous figure. Because this model is utilized for predicting, i.e., the possibility of an event happening, the output might be any number within a given range of values.
- Classification - Support Vector Machines and Naive Bayes are the two most used classification algorithms used in supervised learning. When a classification job is completed, the result is a discrete category from which a choice has to be chosen.

For instance, a machine learning model designed to do sentiment analysis needs to determine if the incoming data can be categorized as positive, negative, or neutral.

2.1.3 Random Forest (RF)

Random forest is a supervised learning method applicable to both classification and regression applications. It is an ensemble learning approach that integrates various classifiers in order to make predictions. This approach increases the performance of the model and is selected for the purpose of this thesis.

RF constructs a decision tree for each subset of the provided data and calculates their average (see Figure 2.2). Essentially, the procedure used by RF comprises of four stages:

1. Pick arbitrary subsets of data from a larger collection.
2. Make a decision tree for each sample and use each decision tree to make a prediction.
3. Conduct a vote for each expected outcome.
4. Select as final prediction the outcome that received the most votes.

A typical random-forest should include 64-128 trees. The accuracy of the method is improved by increasing the number of trees that are used. Random forest is a quick method, that can efficiently cope with the missing and erroneous data. This robustness of RF is another reason for its selection in this research.

2.2 ML-based Approach for Detection of Malware

This is an active field of study that makes use of characteristics collected from static and dynamic analyses of malware. This procedure is nearly usually automated and calls for a dataset to be provided. The findings that have been reported are likewise quite positive. Here are two instances that illustrate the point:

- In [26], a neural network called eXpose is suggested. This method use character-level embedding and neural network convolution to concurrently extract and categorize characteristics from potentially harmful URLs, file paths, named pipes,

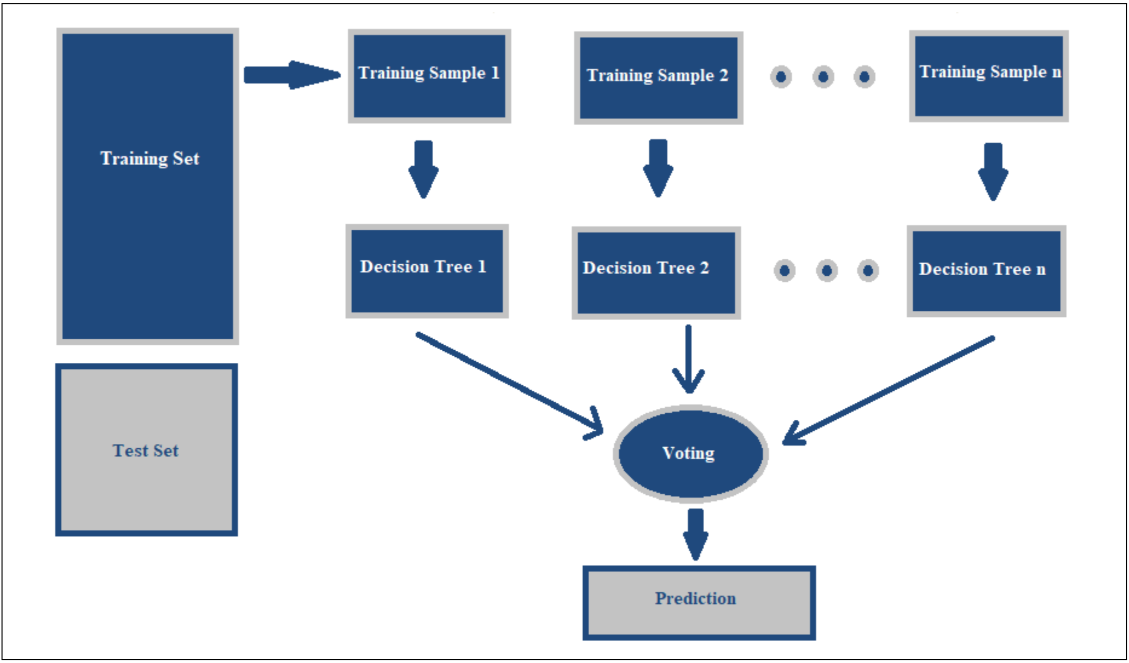


Figure 2.2: Random Forest Explained

named mutexes, and Registry entries. eXpose outperforms manual feature extraction-based baselines on all studied intrusion detection tasks.

- In [27] RAMD is a novel approach that leverages an ensemble classifier to detect both known and unknown malware. RAMD constructs a model of Registry behaviour shown by benign software and then searches for atypical Registry accesses to identify malware. RAMD combines the outputs of one-class classifiers using a particular aggregation operator called Fibonacci-based Super-increasing Ordered Weighted Averaging (FSOWA). RAMD can successfully detect 98.52% of threats and has an accuracy of 98.43%.

2.3 ML-based Approach for Detection of Ransomware

2.3.1 Detection of General Ransomware

The detection of ransomware has been attempted using a variety of features such as opcode, APIs, Registry keys, network data. This section presents seven such studies together with the ML models used and the results obtained.

In [28], an innovative mechanism called DeepAMD is proposed for the early identification of sophisticated ransomware that targets Android systems. DeepAMD extracts static and dynamic data from a suspect application, and processes it to obtain the feature set. It then uses Artificial Neural Network (ANN)s to determine the nature of the suspect application. Its reported accuracy is 95% for static detection and 55% for dynamic detection. DeepAMD is also able to identify scareware and adware.

In [29], a new ransomware detection method that uses hex codes is presented. The binaries are decoded to extract the hex codes and a Random Forest (RF) classifier is developed. This classifier is trained on files and benign software. The 10-fold cross-validation (with information gain-based feature selection) has an accuracy of 88.5%.

[30] relies on capturing and analysing the behaviour of ransomware in its early stages. The focus is on Windows APIs called, Registry key operations, and file system operations. The proposed framework, EldeRan [30], allows a user to watch how ransomware performs some unique operations. This enables the dynamic evaluation of features for ransomware detection. The Mutual Information (MI) criterion is used to choose the informative binary features. Then, a Logistic Regression (LR) classifier is developed and a detection rate of 96.3% is reported.

[31] provides a two-stage Mixed Markov and Random Forest (RF) ransomware detection model. First, a Markov model is developed to capture the properties of ransomware. Then, a RF model is applied to the remaining data. The combined model achieves 97.3% accuracy, a False Positive Rate (FPR) of 4.8%, and a False Negative Rate (FNR) of 1.5%.

[32] a strategy for detecting ransomware is proposed that is based on the sequence in which API calls are made and an SVM classifier. In order to see the sequence in which Windows APIs are called when executing samples, the environment is carefully managed. In order to create a standard vector representation of q-grams, the output log files are used. The SVM-based model had a success rate of 97.48% of the time.

In [33], a new ransomware detecting system called PEELER is presented. To identify the underlying behavioural traits of ransomware, PEELER first analyses large-scale OS-level provenance data gathered from a broad group of ransomware families. It then employs a pre-trained Bidirectional Encoder Representations from Transformers (BERT) model to fingerprint the contextual behaviour of the suspect application. An F1-score of 99.5% is reported for a large ransomware dataset with 67 malware types.

[34] was the first study to explore the effect of ransomware in clinical settings and focuses specifically on *WannaCry*. The authors develop a ML-based system to identify ransomware before it spreads. The first module of their system detects patterns and variations in network traffic when ransomware executes. From these patterns, a probabilistic supervised classifier extracts complicated characteristics. This module needs human oversight to provide an appropriate dataset for the ML algorithm to identify and classify ransomware.

[35] conducted malware analysis to obtain a wide variety of different features including APIs, Summary Information, DLL information, and changed Registry Keys. They used Cuckoo sandbox to dynamically analyse malware and accurately obtained more than 2300 features. Another 92 features were obtained after statically analysis of the malware. The accuracy of their model when trained on features obtained from static analysis was 99.36%. However, when trained on features obtained from dynamic analysis, the accuracy reduced to 94.64%.

2.3.2 Detection of Crypto Ransomware and Locker Ransomware

The following three studies focus specifically on either crypto ransomware or locker ransomware.

In [36] a detection model for the early detection of crypto ransomware is developed. An analysis of the behaviour of crypto ransomware and an estimate of anomalies are employed by the model. The fusion of both findings is used to determine whether a binary is malicious or benign. This method can identify zero-day attacks and sophisticated advanced persistent threats(APTs). The proposed model easily detects ransomware strands in 12,000 application. A low false positive rate means this model can be applied to various ecosystems.

[37] undertakes the static and dynamic analysis of *WannaCry*. After undergoing preliminary processing, the data are utilised to compile a dataset, which is then subdivided into testing data and training data (7:3 ratio). With these three different machine learning classifiers—Random Forest (RF), Gradient Boost (GB), and K-Nearest Neighbors (KNN)—this research obtained an accuracy of 99.99%. The Deep Neural Multilayer Perceptron algorithm is also employed to but it provides slightly lower accuracy of 98%.

In [38] a complete investigation of the transaction mechanism and behaviours of locker ransomware is carried out. After that, an efficient and lightweight detection system that makes use of six distinct sorts of behavioural traits is created. To reach the final findings, an ensemble of four different classifiers was used, and the outcomes were determined by majority voting. It has been stated that the accuracy overall is 99.98% of this method.

2.4 Summary

This chapter provided a comprehensive and up-to-date overview of the research published on the topic of detecting ransomware. In addition to this, it explored how successful an approach based on machine learning may be in the identification of malware. In specifically, it investigated the random forest technique and its potential benefits for the identification and forecasting of malicious software. A research gap has also been identified that relatively little effort has been done on the detection of certain forms of ransomware, such as locker ransomware and needs to be covered considering the increasing number of ransomware attacks everyday.

CHAPTER 3

Research Methodology

3.1 Locker Ransomware Detection and Prediction (LRDPA)

The Locker Ransomware Detection and Prediction Algorithm (LRDPA) developed in this thesis uses signature-based and behavior-based approaches [30]. Figure 3.1 illustrates the data it uses in each approach.

3.1.1 Signature-based Detection

In signature-based detection, the hash digest of a unknown sample is generated. The digest is then compared to a list of hash digests for known locker ransomware samples. Although, hash functions are generally used for integrity verification, the LRDPA uses the SHA-256 hash algorithm for quick and reliable identification [39]. This is possible because hash functions are one-way functions (which means it is not possible to obtain the input from the hash digest) and because any minor change to a sample of locker ransomware will result in a unique hash digest.

3.1.2 Behavior-based Detection

In behavior-based detection, a sample is run in a sandbox environment and a script is used to extract, refine, and input API calls and Registry key data into a machine learning algorithm. Basically, LRDPA's behaviour-based detection consists of two steps. In the first step, it executes a sample and extracts the data. The extracted data consists of *APIs calls* and *Registry keys* data which is then refined. In the second step, the refined

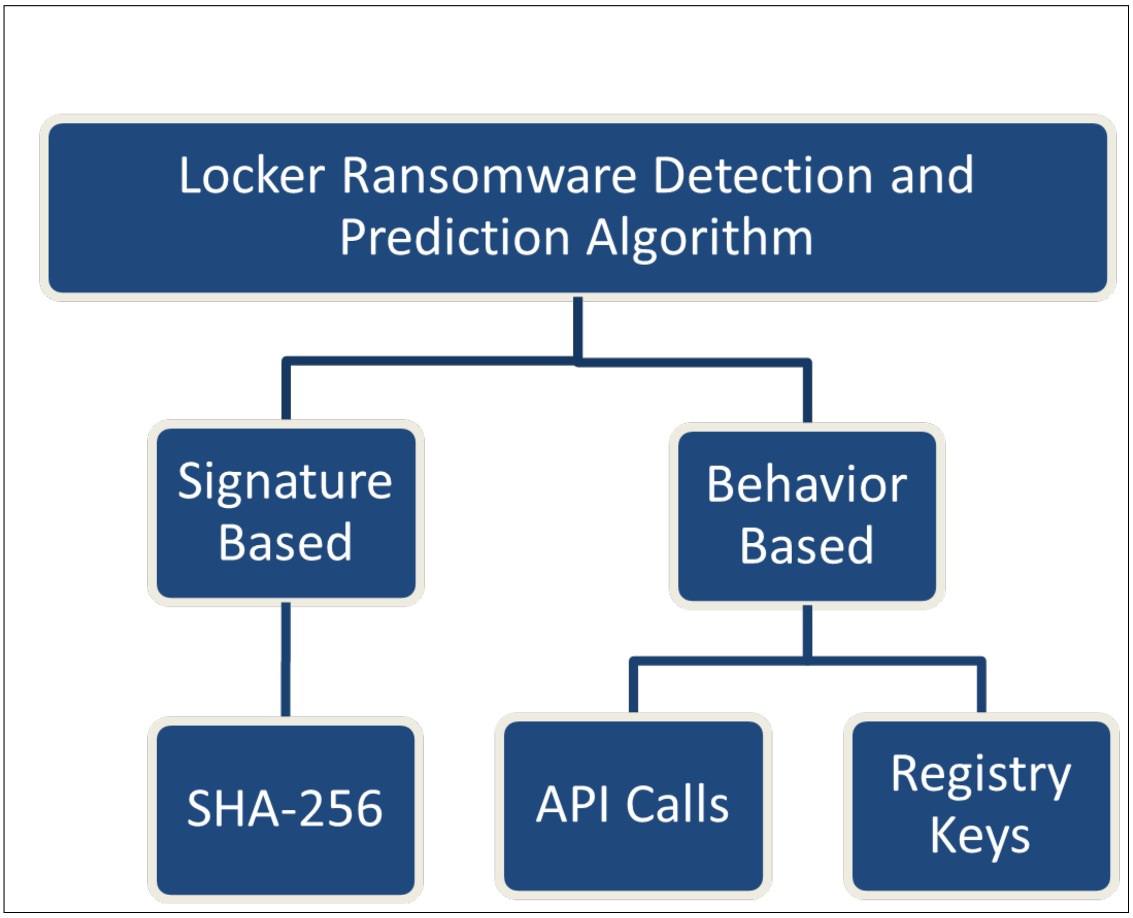


Figure 3.1: LRDPA: Approach and Data

data is used to train a machine learning model. This model is then used to predict if an unknown sample is malware or benign software.

The algorithm selected for LRDPA is *Random Forest (RF)* algorithm. RF is a supervised learning algorithm and is selected for two reasons. Firstly, it is capable of effectively handling huge datasets and secondly it provides higher accuracy than the second most popular malware detection algorithm (i.e. the decision tree algorithm). Another benefit of RF is its ability to handle datasets with continuous and categorical variables (e.g. in regression and classification problems). It also produces more accurate results when dealing with categorization tasks.

3.2 Methodology

The steps in the methodology are illustrated in Figure 3.2.

3.2.1 Collection of Samples

This phase consists of collecting samples of locker ransomware and benign software for the Windows platform. These samples are used to create the dataset.

A total of 505 samples of locker ransomware were collected ranging from *screen-lockers*, *MBR lockers*, and *files lockers*. The following five sources were used for collection:

- *VirusShare* - this renowned site has more than 45 million samples of malware. It is possible to register at this site and obtain samples of malware. A total of 302 samples of locker ransomware were downloaded from this site.
- *TheZoo* - this is an open repository of malware on GitHub. A total of 5 locker ransomware samples were obtained from it.
- *AnyRun* - 110 samples of locker ransomware were downloaded from this site.
- *Malware Bazaar* - 85 samples of locker ransomware were collected for this source.
- *DasMalware* - 3 samples were obtained from this site.

A total of 382 samples of benign software were collected from *Filehippo*, *Softonic*, and *CNET*. They include a variety of software ranging from simple PDF converters, video makers, editors, browsers, and remote desktop tools.

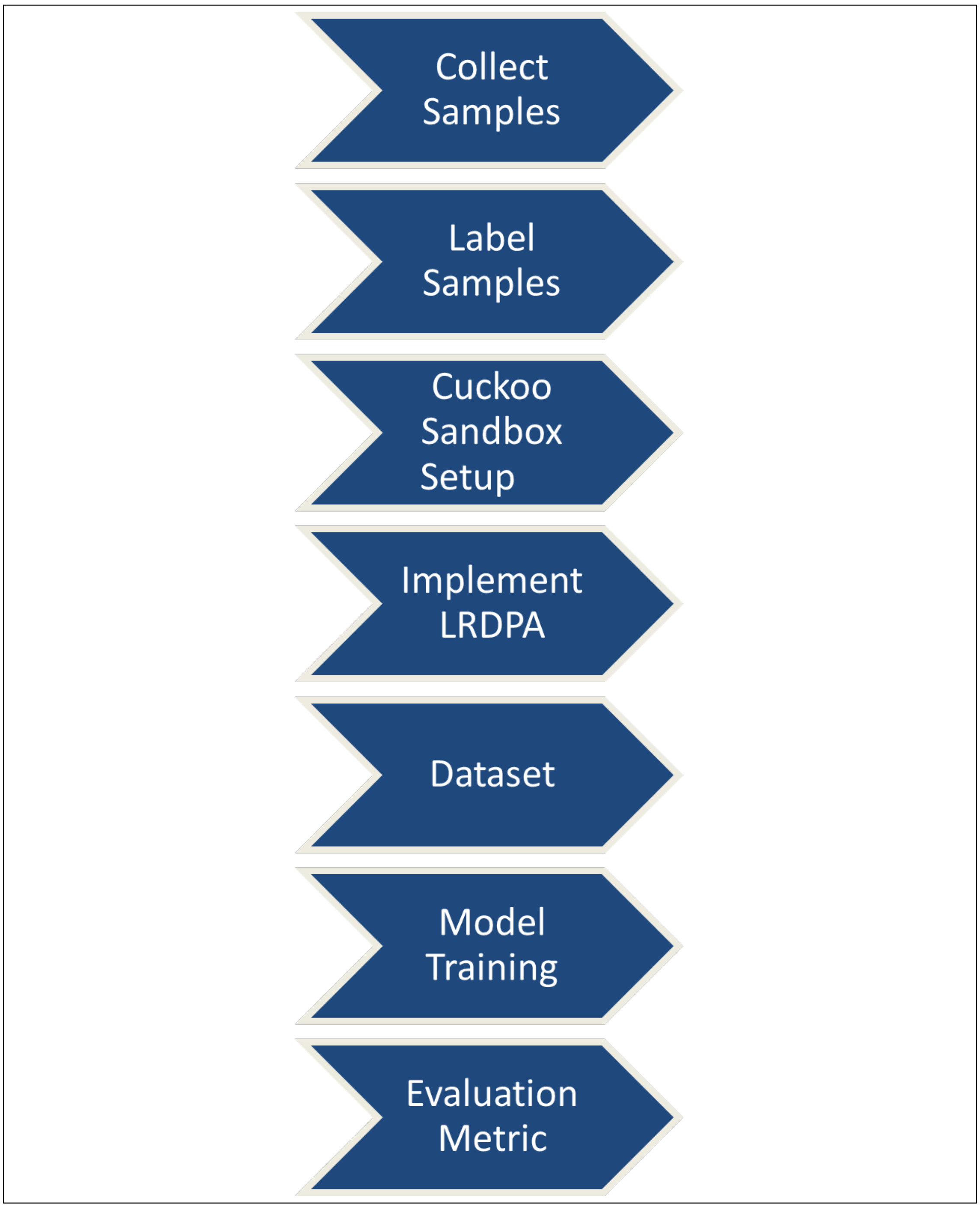


Figure 3.2: Methodology

3.2.2 Labeling of Samples

In this stage, the data is cleaned to guarantee removal of outliers and observations that might lead to misleading conclusions. This is done to ensure that the data is accurate. The procedure makes use of a number of methods but the most essential one is the deletion of records that are null, empty, or duplicates. After the data has been cleaned, the next step is labeling the data. Each sample collected in the previous phase (382 benign software and 505 locker ransomware) is labeled as either "Benign Software" or "Locker Ransomware" with the help of *VirusTotal*. VirusTotal is the world's largest anti-malware scanning service where a suspect file or URL can be submitted for analysis. The service uses more than 70 cutting-edge anti-malware engines to generate a report that includes an overall detection result (i.e. benign, virus, trojan, crypto ransomware, locker ransomware, etc.) [40]. When VirusTotal confirmed that a sample was benign, it was labeled "Benign Software" and added to the dataset. Similarly, when VirusTotal confirmed that a sample was a locker ransomware, it was labeled as "Locker Ransomware" and added to the dataset. If VirusTotal returned any other label, the sample file was not included in the dataset.

3.2.3 Setup of Cuckoo Sandbox

The Cuckoo sandbox is a tool used to execute malware in an isolated environment. It is designed to deceive the malware into believing that it has infected a legitimate host so that it behaves as normally as possible. The sandbox logs the activities of the malware and prepares a detailed report on what the malware tried to accomplish while running within the protected environment.

The Cuckoo sandbox was installed on a Dell Inspiron 15 (500GB SSD and 8GB RAM) dual boot system. The two OSes installed were Ubuntu 18.04 3 LTS (a requirement for Cuckoo sandbox) and Windows 10 Pro. The dependencies for Cuckoo sandbox e.g. Python 2.7, XEN API, MongoDB, TCPdump, and Postgresql, were also installed. Additionally, the MySQL database was setup to store the data extracted from the sample by Cuckoo sandbox.

In order to analyse each sample using Cuckoo sandbox, a virtual environment was needed. VirtualBox was used to create the required Windows 10 guest machine. Windows 10 was selected because it is the most popular OS and because most ransomware attacks target the Windows OS. The network was configured to enable communication between the host machine (running Cuckoo sandbox) and the guest machine (running Windows 10). After each sample was executed on the guest machine, a snapshot was taken, and the guest machine restored to its initial state. Figure 3.3 shows the final setup.

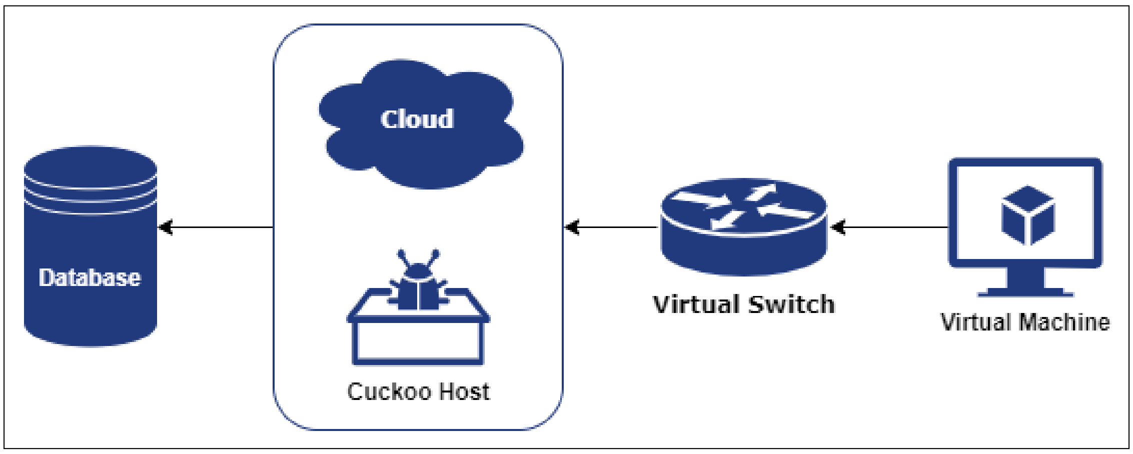


Figure 3.3: Cuckoo Sandbox and System Architecture

3.2.4 Implementation of LRDPA

In this phase, the LRDPA algorithm was implemented in Python. Essentially, this algorithm consists of the following steps:

1. Generate hash digest of the input sample and compare it to the hash digests stored in the signature database.
2. Execute the input sample in Cuckoo sandbox and record all APIs called and Registry keys triggered.
3. Train the RF algorithm using the recorded data and use it to predict if an unknown sample is benign software or locker ransomware.

The detailed LRDPA is shown in Algorithm 1.

3.2.5 Creation of Dataset

The two types of data stored in the MySQL database are explained in this section.

Algorithm 1 Locker Ransomware Detection Algorithm

```
1: INPUT sample
2: if sample is zipped then
3:     EXTRACT sample
4: end if
5: GENERATE hash digest of sample
6: if hash digest exists in Database then
7:     SHOW Label
8:     END
9: else
10:    RUN sample in Cuckoo sandbox
11:    SAVE Report from Cuckoo sandbox
12:    EXTRACT Data (APIs and Registry keys) from Report
13:    REFINE Data
14:    RUN RF Model
15:    SHOW Predicted Label
16:    STORE hash digest in Database
17:    STORE Predicted Label in Database
18:    STORE Data in Database
19:    UPDATE RF Model
20: end if
```

1. **Signatures** - The first table in MySQL database consists of the hash digest or signature of each input sample. This table is used for signature-based detection of locker ransomware. The detection, though inflexible, happens almost instantly (without the need for analysis using Cuckoo Sandbox). Running a sample in Cuckoo sandbox takes around 3 to 4 minutes while comparing hash digests with those stored in this signature table takes an average of 1.3 seconds. The fields of the signature table (Table 3.1) are:
   (a) ID of sample (alphanumeric ID assigned by Cuckoo Sandbox)
   (b) Label ("Benign Software"/"Locker Ransomware")
   (c) Hash Signature (SHA 256)

Table 3.1: Signature Table

Sample ID	Label	Hash Signature
230001	Locker Ransomware	7630b72ba032833fdbbc39b281014a7957d f57e06398d4e09497c0562f957215
...	...	...
...	...	...
...	...	...
2699330	Benign Software	b48d2e9338b96dd9a5511090bcbd14ec3c6 095445716a9cf76183d7d797ece4b
...	...	...
...	...	...
...	...	...

2. **API and Registry Keys** - The second table is created using a list of all APIs called and Registry keys triggered by each input sample.
   (a) API Calls - The Windows Application Programming Interface (API) enables applications to communicate with the Windows OS. For example, an application calls an API to display objects on the screen or receive input from the mouse / keyboard. Regardless of the language, all Windows applications, with the exception of console programs, must interface with the Windows

API. When an application executes, the Cuckoo sandbox records the APIs called and includes them in a summary report with the return codes for each called API.

(b) Registry Keys - A Registry key is similar to a folder, except that it only exists on a Windows system. Also, just as folders hold files, Registry keys hold Registry values [41]. Additionally, a Registry key may include nested Registry keys, which are referred to as Subkeys. Typically, when malware executes, it modifies Registry keys and tries to switch to the privileged mode of the OS. If successful, it gains the ability to modify the OS [42]. Malware may also try to change the Registry in order to relaunch itself after a reboot, to remain hidden, or to interact with an already running legitimate process [43]. Additionally, most malware also alter several Registry keys to circumvent Windows and firewall protection. Overall, the triggering of Registry keys is characteristic of malicious intent and is therefore utilised for identification of malware. Multiple changes to the Registry increase the likelihood that a program is malicious [42]. The report generated by Cuckoo sandbox for each input sample includes the four main types of Registry keys triggered:

   i. ***regkey-opened***: Registry keys that are marked `opened` in Cuckoo sandbox report.
   ii. ***regkey-deleted***: Registry keys that are marked `deleted` in Cuckoo sandbox report.
   iii. ***regkey-written***: Registry keys that are marked `written` in Cuckoo sandbox report.
   iv. ***regkey-read***: Registry keys that are marked `read` in Cuckoo sandbox report.

The data extracted from the report is in the form of a JSON file. From this file, the following fields are refined and stored in a CSV file:

(a) ID of sample (alphanumeric ID assigned by Cuckoo Sandbox),

(b) Label ("Benign Software"/"Locker Ransomware")

(c) API calls (total: 282)

(d) Registry keys (total: 21,780)

When an API is called or a Registry key is triggered by an input sample, a '1'is written in the column for that API or Registry key. If an API is not called or a Registry key is not triggered, a '0'is written to the corresponding cell. The contents of this file are stored in the second table (Table 5.1) of database.

Table 3.2: API and Registry Key Table

Sample ID	**Label**	**API1**	**API2**	**...**	**RegKey1**	**RegKey2**	**...**
230001	Locker Ran-somware	0	1	...	1	0	...
230002	Locker Ran-somware	1	1	...	1	0	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...
230734	Benign Software	1	0	...	1	0	...
230735	Benign Software	0	0	...	1	0	...
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...

Once the dataset is compiled, it is scanned for any fields that contain the NULL value. Such instances are removed with the help of the Python `Pandas` library in order to minimise bias in the LRDPA model. To split the dataset for training and testing purposes, the "train_test_split" function from `scikit-learn` library is used with the ratio set to 80% and 20% respectively.

3.2.6 Training the Model

Training is the most critical phase in the machine learning pipeline. During this stage, the prepared data is fed to the model. The model uses an algorithm to learn from the data, search for patterns, and eventually generate predictions. The more data the model is fed, the better it becomes at making accurate predictions.

When the same dataset is applied to different ML algorithms, the output is different. Consequently, choosing the appropriate algorithm is essential. For this thesis, the random forest classifier from the `sklearn ensemble` is used. This is a meta estimator that fits a variety of decision tree classifiers on different sub-samples of the dataset and utilises averaging to increase the prediction accuracy and to control over-fitting.

3.2.7 Evaluation of the Model

The performance of a ML model is best evaluated using a number of metrics [44]. The first and most basic metric is the *Confusion Matrix.* This matrix reports the following:

- *True Positive* (TP) - the number of instances where the model predicted positive and the actual outcome was positive as well.
- *True Negative* (TN) - the number of instances where the model predicted negative and the actual outcome was negative as well.
- *False Positive* (FP) - the number of instances where the model predicted positive but the the actual outcome was negative.
- *False Negative* (FN) - the number of instances where the model predicted negative but the the actual outcome was positive.

The second frequently-used metric is *Accuracy* (ACC). This metric represents the model's overall performance across all classes or categories. It is beneficial especially when all classes are equally important. It is determined by dividing the number of correct predictions by the total number of predictions.

$$= \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

The following nine metrics are also used in this thesis:

- *Precision* - this is the ratio of correctly predicted positives to the total expected positives in a class [45].

$$= \frac{TP}{(TP + FP)}$$

- *Sensitivity* - this metric is used to quantify the proportion of correctly categorized positives over the total actual positives. It is also known as the *Detection Rate* or the *True Positive Rate* (TPR).

$$= \frac{TP}{(TP + FN)}$$

- *False Positive Rate* (FPR) - quantifies the accuracy with which the proposed model predicts the positives incorrectly.

$$= \frac{FP}{(FP + TN)}$$

- *Specificity* - this metric is used to quantify the proportion of incorrectly categorised negatives. It is also known as the *True Negative Rate* (TNR).

$$= \frac{TN}{(TN + FP)}$$

- *False Negative Rate* (FNR) - This is the likelihood that a real positive will be overlooked by the model. It is also known as the *Miss Rate*.

$$= \frac{FN}{(FN + TP)}$$

- *Negative Predictive Value* (NPV) - This refers to the likelihood that a negative result is correct.

$$= \frac{TN}{(FN + TN)}$$

- *False Discovery Rate* (FDR) - this is the fraction of all predictions that are incorrect.

$$= \frac{FP}{(FP + TP)}$$

- *F1 Score* - this is another measure of the accuracy of a model on a dataset. It is defined as the harmonic mean of the accuracy and recall [46].

$$= \frac{2 \times Precision \times Recall}{(Precision + Recall)}$$

- *Matthews Correlation Coefficient* (MCC) - determines how well the predictions of a model coincides with the actual outcomes. It evaluates the disparity between predicted and actual values [47].

$$= \frac{TP \times TN - FP \times FN}{(\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)})}$$

3.3 Flow Diagram

The complete flow diagram of LRDPA is given in Figure 3.4. The important stages are:

1. Input a sample (executable file) for analysis.
2. If the sample is in a compressed format
   (a) then extract it
3. Generate hash digest (SHA-256) of the sample.
4. Compare the hash digest with the signatures stored in the database to find a match.
   (a) If the match found has the label "Benign Software", then inform the user and terminate.
   (b) If the match found has the label "Locker Ransomware", then quarantine the sample, generate an alert for the user, and terminate.
5. If no match is found, then store the hash digest in the database as a new entry.
6. Run the sample in Cuckoo sandbox.
7. Extract and refine the data (API calls and Registry Keys) from the Cuckoo sandbox report.
8. Use the RF model to predict label for the sample ("Locker Ransomware" or "Benign Software")
   (a) If the predicted label is "Locker Ransomware", then quarantine the sample and generate alert for user.
   (b) If the predicted label is "Benign Software", then inform the user.
9. Update database with the predicted label and store data (API calls and Registry Keys).

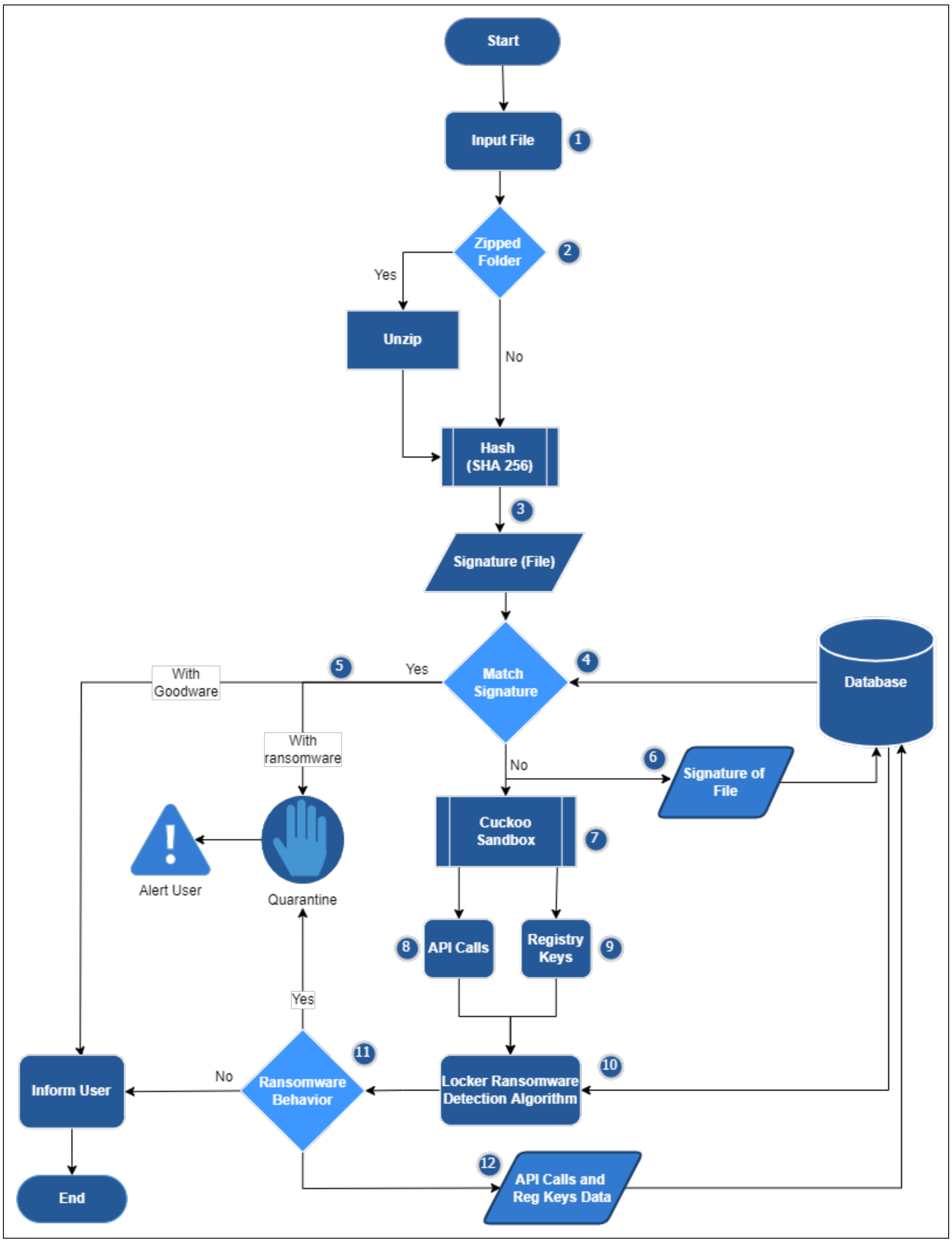


Figure 3.4: Complete Flow Diagram

3.4 Summary

This chapter presents the Locker Ransomware Detection and Prediction Algorithm (LRDPA). It also covered the seven steps of the methodology used in this research. In the beginning, samples of locker ransomware and benign software were collected. These samples were labelled and then analysed using the Cuckoo sandbox environment. The data collected during this analysis was used to create a dataset. Then LRDPA was implemented using the Random Forest (RF) algorithm. The dataset was split into training and test data (80% and 20% respectively). The former was used to train the model and the latter to evaluate it. Finally, the metrics selected for evaluation were presented. The complete flow diagram was included as Figure 3.4.

Chapter 4

Analysis

In this chapter the APIs called and the Registry keys triggered by locker ransomware and benign software are analysed.

4.1 Unique to Locker Ransomware

4.1.1 API Calls

A quick look at the list of APIs shows that:

- some APIs are *common* i.e. they are called by benign software as well as locker ransomware.
- some APIs are *unique* meaning they are only called by locker ransomware.

A total of 31 unique APIs were identified. The purpose of 22 of these unique APIs is shown in Table 4.1. Examples include `CryptEncrypt` for encrypting data, `NtWriteVirtualMemory` for writing to memory, and `MessageBoxTimeout` for showing a message box.

Table 4.1: 28/31 Unique APIs in Locker Ransomware [3]

API	**Purpose**
`NtSetContextThread`	Sets the usermode context of the specified thread.
`StartService (A & W)`	Starts a service.

Continued on next page

Table 4.1 – *Continued from previous page*

API	**Purpose**
`CopyFile`	Copies an existing file to a new file.
`RtlAddVectoredContinueHandler`	Adds vectored exception handler
`ExitWindowsEx`	Logs off the interactive user, shuts down the system, or shuts down and restarts the system.
`CryptDecrypt`	Decrypts data previously encrypted by using the `CryptEncrypt` function.
`CryptEncrypt`	Encrypts data
`NtWriteVirtualMemory`	Writes on the memory
`MessageBoxTimeout`	Shows timed messagebox
`RtlDecompressBuffer`	Decompresses an entire compressed buffer
`CreateService (A & W)`	Creates a service object and adds it to the specified service control manager database.
`NtTerminateThread`	Terminates a thread.
`CreateRemoteThread`	Creates a thread that runs in the virtual address space of another process.
`ControlService`	Sends a control code to a service.
`RtlRemoveVectoredExceptionHandler`	Removes vectored exception handler.
`InternetOpenUrlA`	Opens a resource specified by a complete FTP or HTTP URL.
`CopyFileExW`	Copies an existing file to a new file, notifying the application of its progress through a callback function.
`InternetCrackUrlA`	Cracks a URL into its component parts.
`CryptGenKey`	Generates a random cryptographic session key or a public/private key pair.
`OpenService (A & W)`	Opens an existing service.

Continued on next page

Table 4.1 – *Continued from previous page*

API	**Purpose**
`CryptUnprotectMemory`	Decrypts memory that was encrypted using the `CryptProtectMemory` function.
`NtShutdownSystem`	Shut down all drivers, flushing Registry hives and the disc cache, clearing the page file, etc and then shuts the window down.
`GetKeyboardState`	Copies the status of each of the 256 virtual keys to the buffer you specify.
`DeleteUrlCacheEntry`	Erases from memory a copy of any files whose name begins with the source name, if any.
`URLDownloadToFile`	Downloads bits from the Internet and saves them to a file.
`NtQueryFullAttributesFile`	It supplies network open information for the specified file.
`NtDeleteKey`	Deletes an open key from the registry.
`CryptProtectData`	Performs encryption on the data.

Figure 4.1 shows the percentage of locker ransomware samples in which these 31 unique APIs were called. The set of unique APIs triggered by more than 70% of locker ransomware samples provides a strong basis for prediction of zero-day locker ransomware.

4.1.2 Registry Keys

As with APIs, some Registry keys are only called by locker ransomware. Out of the 21,780 Registry keys recorded, 5,082 Registry keys were triggered only by locker ransomware (but not by benign software). The distribution of these unique Registry keys among the four categories mentioned earlier is shown in Figure 4.2:

- `regkey-read` category: from this category 3,562 unique Registry keys were triggered. This category contains Registry keys that are read by the system to match any pre-written condition or to find a specific Registry key that malware might want to alter or delete.

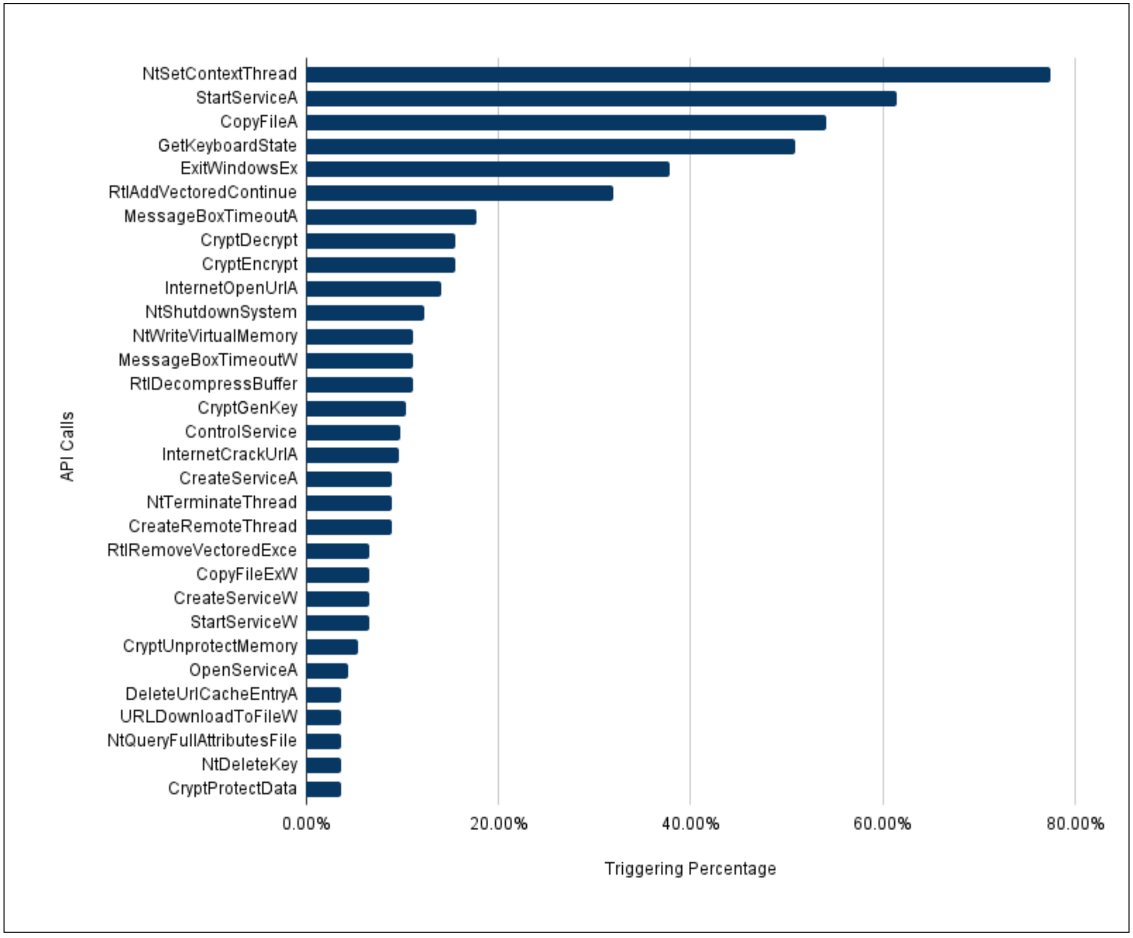


Figure 4.1: Triggering Percentage of 31/31 Unique APIs in Locker Ransomware

- `regkey-opened` category: from this category 1,314 unique Registry keys were triggered. A pre-written Registry key must be opened before it can be edited or updated.

- `regkey-deleted` category: from this category 168 unique Registry keys were triggered. Registry keys marked as written are the ones written to by the locker ransomware.

- `regkey-written` category: from this category 38 unique Registry keys were triggered. The deleted keys are the ones that were deleted from the system by the locker ransomware. Registry keys are typically deleted to hide activity or restrict system performance or functionality in some manner.

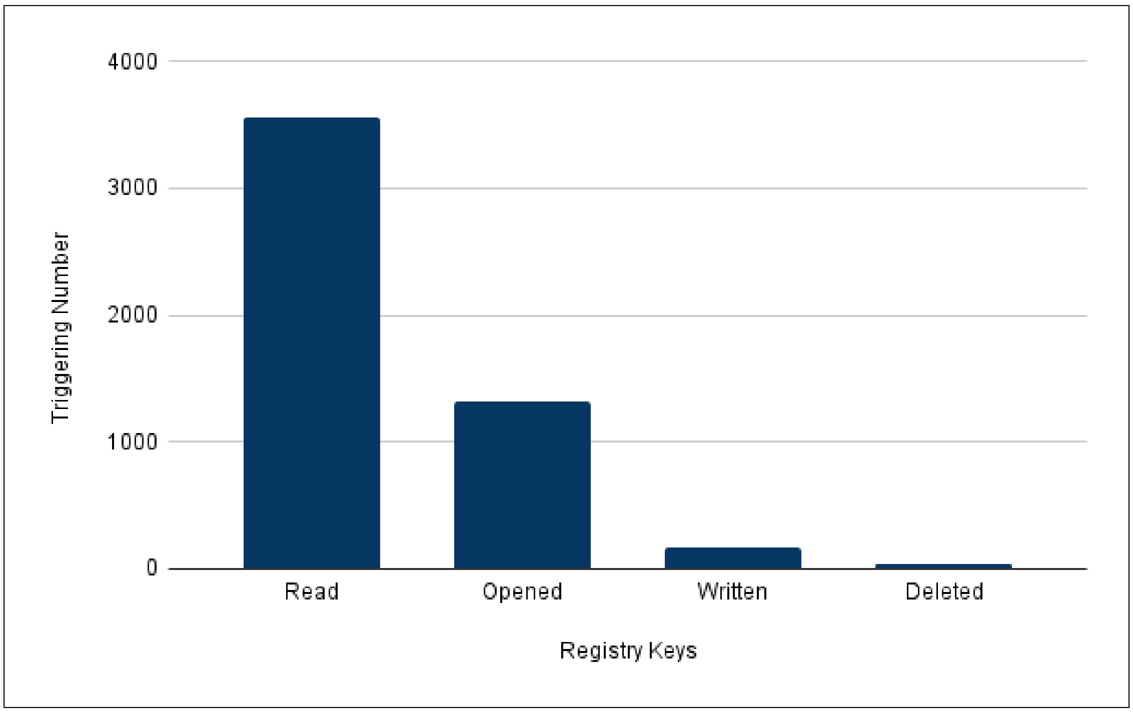


Figure 4.2: Distribution of 5,082 Unique Registry key in Locker Ransomware

4.2 Present in more than 70% locker ransomware

4.2.1 API Calls

A total of 21 APIs were called in more than 70% of locker ransomware samples. The purpose and triggering percentage of these APIs is given in Table 4.2. The triggering percentage is represents the proportion of locker ransomware samples which called the API over the total number of locker ransomware sample.

Table 4.2: 21/21 APIs Called in more than 70% of Locker Ransomware[3]

API	Triggering Percent-age	Purpose	Unique
`NtClose`	100.00%	Closes an object handle.	N
`NtOpenKey`	98.77%	Opens an existing Registry key	N

Continued on next page

Table 4.2 – *Continued from previous page*

API	**Triggering Percent-age**	**Purpose**	**Unique**
`LdrGetProcedureAddress`	95.82%	Retrieves the address of an exported function or variable from the specified dynamic-link library (DLL)	N
`NtQueryValueKey`	95.82%	returns a value entry for a Registry key	N
`LdrLoadDll`	95.09%	Windows will look in the application folder as well as the system folder for the.dll file in question.	N
`NtAllocateVirtualMemory`	94.59%	Reserves, commits, or both, a region of pages within the user-mode virtual address space of a specified process	N
`LdrGetDllHandle`	92.38%	Similar to LoadLibrary, this is a low-level function that loads a DLL into a process [48].	N
`NtCreateFile`	91.65%	Creates a new file or directory, or opens an existing file, device, directory, or volume.	N
`NtFreeVirtualMemory`	89.19%	Releases, decommits, or both releases and decommits, a region of pages within the virtual address space of a specified process.	N
`RegCloseKey`	86.98%	Closes a handle to the specified Registry key.	N
`NtReadFile`	81.82%	Reads data from an open file.	N
`RegOpenKeyExW`	80.34%	Opens the specified Registry key.	N
`GetSystemTimeAsFileTime`	79.61%	Retrieves the current system date and time.	N

Continued on next page

Table 4.2 – *Continued from previous page*

API	**Triggering Percentage**	**Purpose**	**Unique**
`RegQueryValueExW`	78.38%	Retrieves the type and data for the specified value name associated with an open Registry key.	N
`RegOpenKeyExA`	77.89%	Opens the specified Registry key.	N
`NtSetContextThread`	77.40	Sets the context of a thread.	Y
`NtProtectVirtualMemory`	75.68%	changes the protection on a region of committed pages within the virtual address space of the subject process.	N
`NtMapViewOfSection`	75.43%	Maps a view of a section into the virtual address space of a subject process.	N
`NtCreateSection`	73.46%	Create a new memory section.	N
`LdrUnloadDll`	73.22%	Similar to UnLoadLibrary, this is a low-level function that unloads a DLL into a process [48].	N
`NtDuplicateObject`	70.76%	Creates a handle that is a duplicate of the specified source handle.	N

4.2.2 Registry Keys

Similar to APIs, a list of Registry keys triggered by more than 70% of the locker ransomware samples was compiled. Table 4.3 presents this list along with the triggering percentage and a label that indicates if the Registry key is unique or not.

Table 4.3: 17/17 Registry keys Triggered in more that 70% of Locker Ransomware

Regkey	**Triggering Percentage**	**Unique**
REG:DELETED:HKEY_CURRENT_USER\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\INTERNETSETTINGS\ZONEMAP\INTRANETNAME	97%	N
REG:DELETED:HKEY_CURRENT_USER\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\INTERNET SETTINGS\ZONEMAP\PROXYBYPASS	97%	N
REG:DELETED:HKEY_LOCAL_MACHINE\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\INTERNET SETTINGS\ZONEMAP\INTRANETNAME	78%	N
REG:DELETED:HKEY_LOCAL_MACHINE\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\INTERNET SETTINGS\ZONEMAP\000000000000000000000000000000000000000000000PROXYBYPASS	78%	N
REG:DELETED:HKEY_CURRENT_USER\SOFTWARE\MICROSOFT\OFFICE\14.0\COMMON\LANGUAGERESOURCES\LANGTUNEUP	72%	Y
REG:DELETED:HKEY_CURRENT_USER\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\EXPLORER\DISCARDABLE\POSTSETUP\COMPONENT CATEGORIES\{00021493-0000-0000-C000-000000000046}\ENUM\(DEFAULT)	72%	Y
REG:DELETED:HKEY_CURRENT_USER\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\EXPLORER\DISCARDABLE\POSTSETUP\COMPONENT CATEGORIES\{00021494-0000-0000-C000-000000000046}\ENUM\(DEFAULT)	72%	Y
REG:DELETED:HKEY_CURRENT_USER\SOFTWARE\MICROSOFT\OFFICE\14.0\COMMON\LCCACHE\SMARTART\1033\NEXTUPDATE	71%	Y
REG:DELETED:HKEY_CURRENT_USER\SOFTWARE\MICROSOFT\OFFICE\14.0\COMMON\LCCACHE\THEMES\1033\NEXTUPDATE	71%	Y

Continued on next page

Table 4.3 – *Continued from previous page*

Regkey	**Triggering Percentage**	**Unique**
REG:DELETED:HKEY_CURRENT_USER\SOFTWARE\MICROSOFT\OFFICE\14.0\COMMON\LCCACHE\WORDDOCPARTS\1033\NEXTUPDATE	71%	Y
REG:WRITTEN:HKEY_CURRENT_USER\SOFTWARE\MICROSOFT\ASF STREAM DESCRIPTOR FILE\SETTINGS\DON'T SHOW BOOT DIALOG	72%	Y
REG:OPENED:HKEY_LOCAL_MACHINE\SOFTWARE\POLICIES\MICROSOFT\WINDOWS NT\RPC	76%	N
REG:OPENED:HKEY_LOCAL_MACHINE\SOFTWARE\MICROSOFT\RPC	72%	N
REG:READ:HKEY_LOCAL_MACHINE\SYSTEM\CONTROLSET001\CONTROL\NLS\CUSTOMLOCALE\EN-US	96%	N
REG:READ:HKEY_LOCAL_MACHINE\SYSTEM\CONTROLSET001\CONTROL\NLS\EXTENDEDLOCALE\EN-US	96%	N
REG:READ:HKEY_LOCAL_MACHINE\SOFTWARE\MICROSOFT\WINDOWS NT\CURRENTVERSION\GRE_INITIALIZE\DISABLEMETAFILES	94%	N
REG:READ:HKEY_LOCAL_MACHINE\SOFTWARE\MICROSOFT\SQMCLIENT\WINDOWS\CEIPENABLE	72%	N

4.3 Contrasting Percentage

The contrasting percentage was calculated for each API called and each Registry key triggered by subtracting its triggering percentage for benign software from its triggering percentage for locker ransomware. A contrasting percentage greater than +50% indicates the behavior of locker ransomware, and a contrasting percentage less than -50% indicates the behavior of benign software.

4.3.1 Contrasting percentage of API calls

Figure 4.3 displays the contrasting percentage of APIs called by locker ransomware and benign software. It can be easily noted that there are only 4 APIs (out of a total of 282) with greater than +50% contrasting percentage. These APIs indicate the behaviour of locker ransomware.

1. `StartServiceA` - Starts a service [3].
2. `NtSetContextThread` - Context to set to thread [49].
3. `GetKeyboardState` - Copies the status of the 256 virtual keys to the specified buffer [3].
4. `CopyFileA` - Copies an existing file to a new file [3].

Only the other hand, there are only 7 APIs (out of a total of 282) with a contrasting percentage less than -50%. These APIs indicate the behaviour of benign software:

1. `GlobalMemoryStatusEx` - retrieves information about the system's current usage of both physical and virtual memory [50].
2. `DrawTextExW` - draws formatted text in the specified rectangle [51].
3. `CreateActCtxW` - creates an activation context [52].
4. `NtSetInformationFile` - changes various kinds of information about a file object [53].
5. `NtSetValueKey` - creates or replaces a Registry key's value entry [54].
6. `ShellExecuteExW` - performs an operation on a specified file [55].
7. `NtShutdownSystem` - closes system [56].

4.3.2 Contrasting percentage of registry keys

Out of the 21,780 Registry keys recorded, 121 had a contrasting percentage greater than +50% and 194 had a contrasting percentage less than -50% (see Figure 4.4).

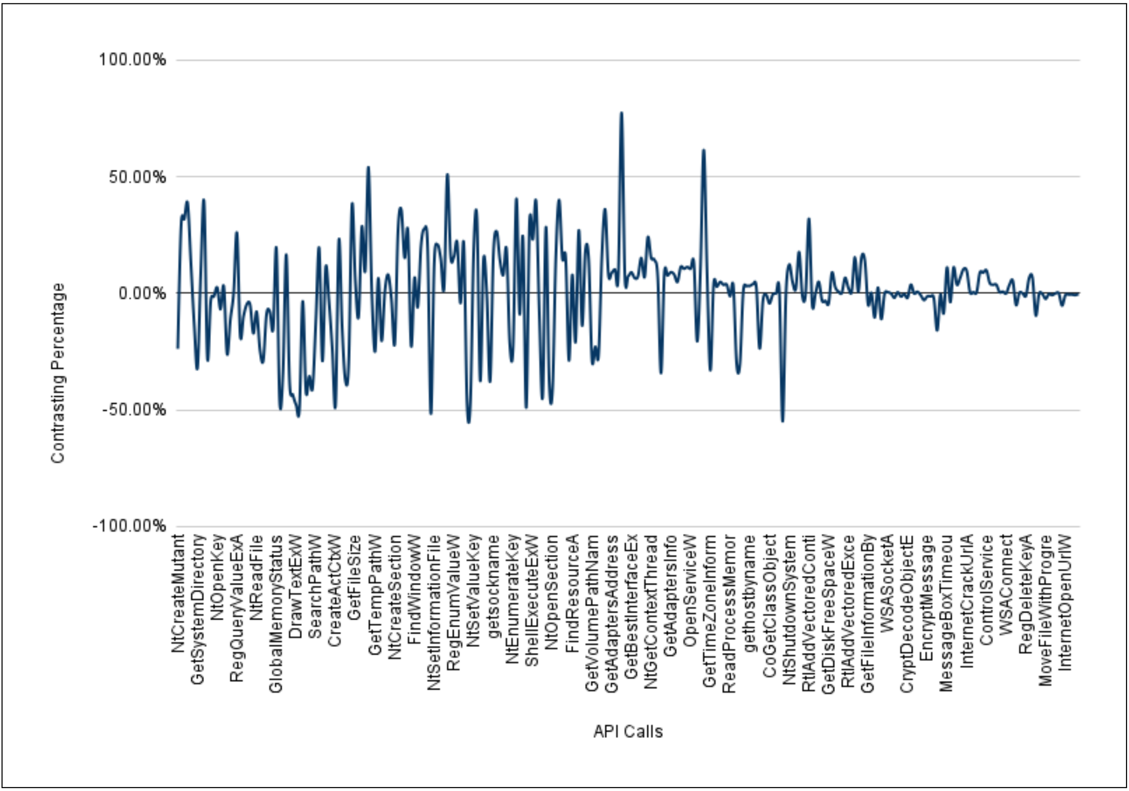


Figure 4.3: Contrasting Percentage for APIs Called

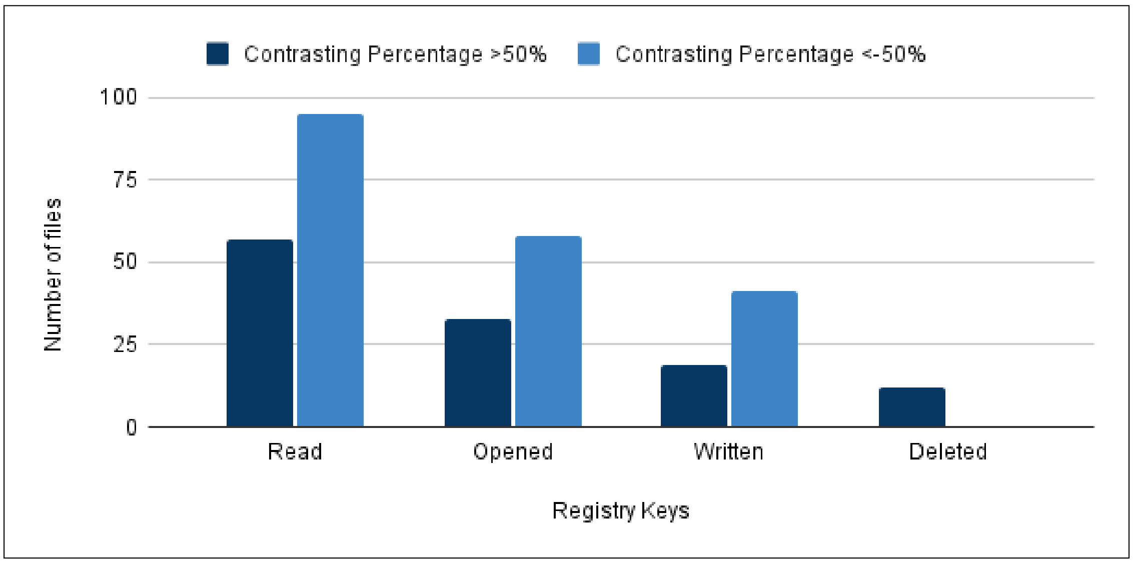


Figure 4.4: Distribution of Contrasting Percentage for Registry keys Triggered

4.4 Summary

The data obtained from the dynamic analysis of 382 samples of benign software and 505 sample locker ransomware using the Cuckoo sandbox environment is analysed in three ways. Table 4.4 summarises the key takeaways. The number of APIs called and Registry keys is shown out of the total of 282 and 21,780 respectively. It is clear that the profile of APIs called and Registry keys triggered by locker ransomware is very distinctive. Only 1 API and 7 Registry keys are unique and have a triggering percentage of more than 70%. These are:

- `NtSetContextThread`
- `REG:DELETED:HKEY_CURRENT_USER\SOFTWARE\MICROSOFT\OFFICE\14.0\COMMON\LANGUAGERESOURCES\LANGTUNEUP`
- `REG:DELETED:HKEY_CURRENT_USER\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\EXPLORER\DISCARDABLE\POSTSETUP\COMPONENT CATEGORIES\{00021493-0000-0000-\ENUM\(DEFAULT)`
- `REG:DELETED:HKEY_CURRENT_USER\SOFTWARE\MICROSOFT\WINDOWS\CURRENTVERSION\EXPLORER\DISCARDABLE\POSTSETUP\COMPONENT CATEGORIES\{00021494-0000-0000-\ENUM\(DEFAULT)`
- `REG:DELETED:HKEY_CURRENT_USER\SOFTWARE\MICROSOFT\OFFICE\14.0\COMMON\LCCACHE\SMARTART\1033\NEXTUPDATE`
- `REG:DELETED:HKEY_CURRENT_USER\SOFTWARE\MICROSOFT\OFFICE\14.0\COMMON\LCCACHE\THEMES\1033\NEXTUPDATE`
- `REG:DELETED:HKEY_CURRENT_USER\SOFTWARE\MICROSOFT\OFFICE\14.0\COMMON\LCCACHE\WORDDOCPARTS\1033\NEXTUPDATE`
- `REG:WRITTEN:HKEY_CURRENT_USER\SOFTWARE\MICROSOFT\ASF STREAM DESCRIPTOR FILE\SETTINGS\DON'T SHOW BOOTDIALOG`

Also, only 4 APIs and 121 Registry keys have a contrasting percentage of more than 50% and therefore indicative of locker ransomware. Following are those 4 APIs:

- `CopyFileA`

- `GetKeyboardState`
- `NtSetContextThread`
- `StartServiceA`

These observations should be effective for the efficient detection and prediction of locker ransomware.

Table 4.4: Summary of Analysis

Type of Analysis	**Number of APIs**	**Number of Registry Keys**
Unique (Locker Ransomware)	31	5,082
More than 70% Triggering Percentage (Locker Ransomware)	21	17
Unique AND More than 70% Triggering Percentage	1	7
More than 50% Contrasting Percentage (Locker Ransomware)	4	121
Less than -50% Contrasting Percentage (Benign Software)	7	194

Chapter 5

Results

This chapter presents the results for LRDPA in the following three configurations:

1. only APIs called data,
2. only Registry keys triggered data, and
3. combined data - both APIs called and Registry keys triggered data.

In each case, the data was divided so that 80% was used for training and 20% was used for testing. The accuracy and F1-scores are reported for each of the three configurations. It is common practice to rely on accuracy only when the classes in a dataset are evenly distributed. However, when the classes are unbalanced, as is this case, it is best to include F1-score as well.

5.1 LRDPA Using Only APIs Called Data

The pattern of APIs called by a program presents a complete profile of its behaviour. Therefore, fairly high accuracy (93.26%) and F1 score (91.78%) were obtained when LRDPA is trained using only API data (see Figure 5.1).

5.2 LRDPA Using Only Registry Keys Triggered Data

LRDPA has an accuracy of 85.39% and an F1 score of 82.19% (also shown in Figure 5.1) when only Registry keys triggered data is used. This is lower than the performance of

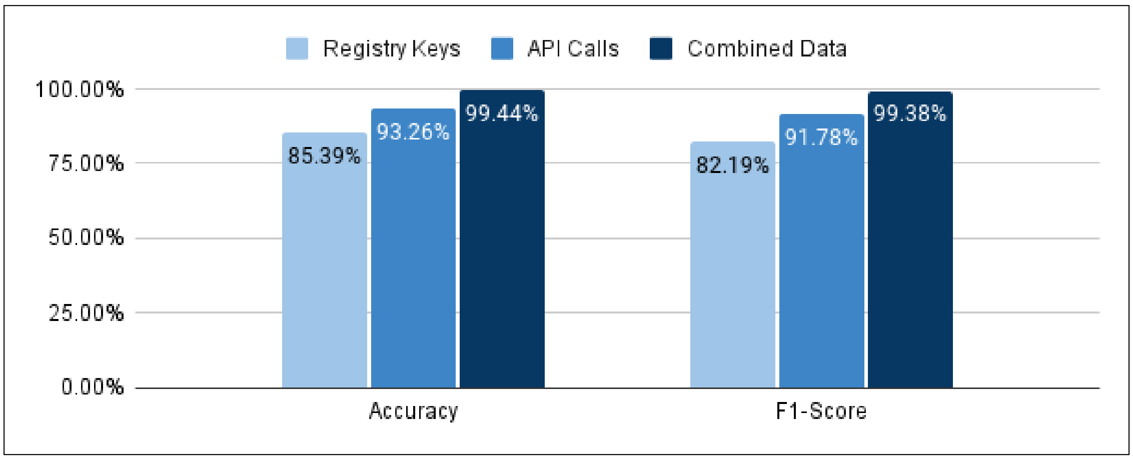


Figure 5.1: Accuracy and F1-Score

LRDPA using only APIs called data. It should be noted though that Registry key data can still contribute and provide some useful insight. During the analysis stage, it was noted that some Registry keys are triggered by majority of the locker ransomware samples and they can be reliable indicators of malicious behaviour. One suitable example is `REG:WRITTEN:HKEY_CURRENT_USER\SOFTWARE\MICROSOFT\ASF_STREAM_DESCRIPTOR_FILE\SETTINGS\DON'T_SHOW_BOOT_DIALOG`. This key was not triggered by any benign software but by 81% of the locker ransomware samples. Similarly, `REG:WRITTEN:HKEY_CURRENT_USER\CONTROL_PANEL\DESKTOP\WALLPAPER` was triggered by 67% of locker ransomware samples. This key is used to change the wallpaper of the desktop and attackers use it to display the ransom message.

The other notable Registry keys belonging to the `regkey-written` category were:

- `REG:DELETED:HKEY_CURRENT_USER\CONTROL_PANEL\MMCPL\MLCFG32.CPL` was triggered in 56% of locker ransomware samples. Deleting this Registry key removes the mail icon from the control panel.
- `REG:DELETED:\STARTPERSIST` is called to start a system and it was removed in 57% of the locker ransomware samples.
- `REG:DELETED:HKEY_LOCAL_MACHINE\SOFTWARE\MICROSOFT\SYSTEM\CERTIFICATES\TRUSTED_PEOPLE\CERTIFICATES` deletes user certificates that are marked as trusted.

Furthermore, some Registry keys were found to be indicative of certain lockers ransomware. For example, `REG:WRITTEN:HKEY_CURRENT_USER\SOFTWARE\MDSLK\SELF` in

Medusalocker and `REG:WRITTEN:HKEY_CURRENT_USER\SOFTWARE\LOCKBIT\PUBLIC` in *Lockbit*.

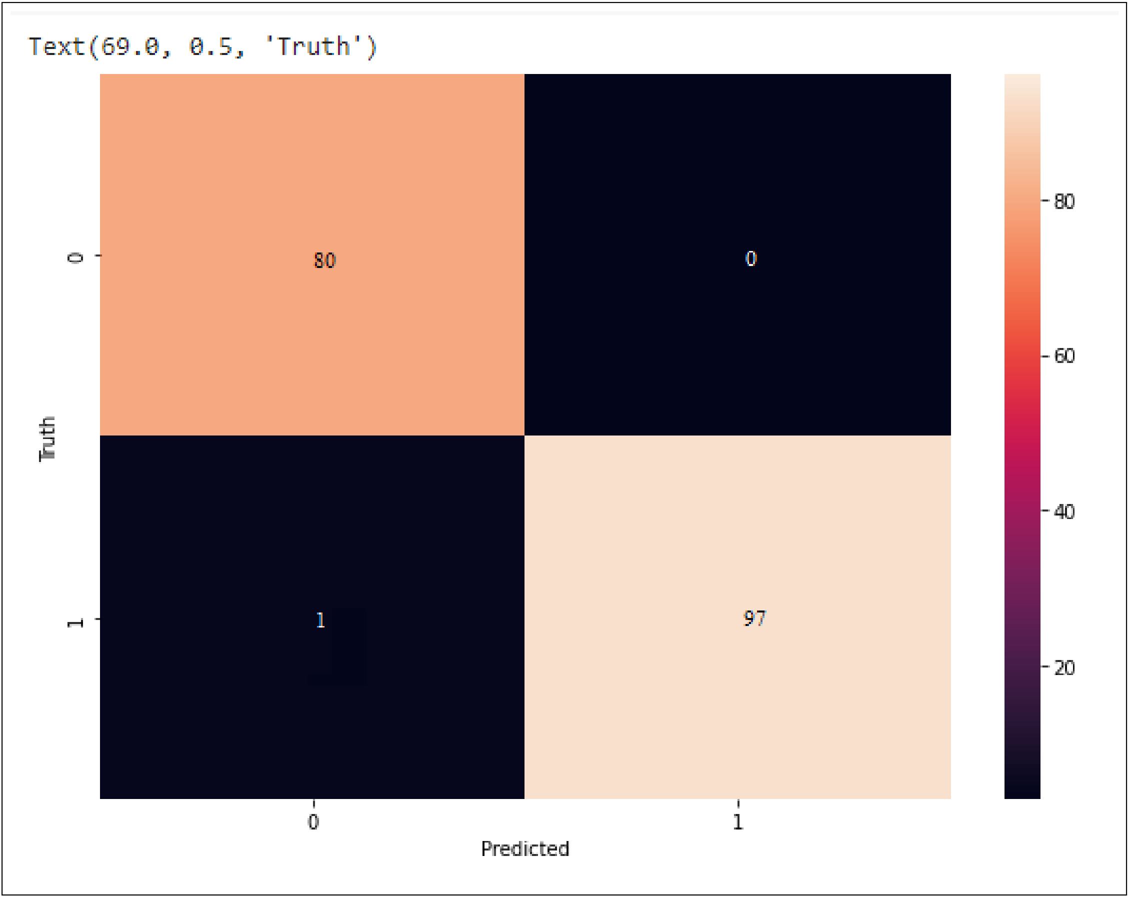
5.3 LRDPA using Combined Data

The combined data when fed to the LRDPA provides the best accuracy of 98.31%. The F1 score is also the highest (98.14%) as shown in Figure 5.1. Additional fine-tuning, discretization, and parsing of the dataset enhances the accuracy of LRPDA to 99.44%. The other metrics confirm the conclusion that combined data provides the best result for LRDPA. The confusion matrix, given in Figure 5.2, shows only 1 False Negative (FN) and 0 False Positives (FP).

A comparison of the remaining eight evaluation metrics — Sensitivity, Specificity, Precision, Negative Predictive Value (NPV), False Discovery Rate (FDR), Matthews Correlation Coefficient (MCC), , False Negative Rate (FDR), False Positive Rate (FPR) is presented in Figure 5.3.

5.4 K-Fold Cross Validation

As there is never enough data to train a model, omitting some data for validation causes under-fitting. By limiting data for training, there is risk of losing essential patterns/trends, which increases bias-induced inaccuracy. This raises a need to have enough data for training and validation. Cross-validation K-fold helps to achieve this requirement. The approach essentially divides the data into k subgroups and repeats the holdout process k times. It employs one of the k subsets as the test/validation set and the other k-1 subsets as the training set. The total efficacy of a model is the error estimation averaged across k trials. Every data point occurs once in the training set and k-1 times in the validation set. This reduces bias since the majority of data is used for fitting and validation. Switching training and test sets improves this strategy. K = 5 or 10 is preferred empirically, although it may be any number. For LRDPA, 10 fold cross validation was used.



```
Text(69.0, 0.5, 'Truth')
```

Figure 5.2: Confusion Matrix for LRDPA - Combined Data

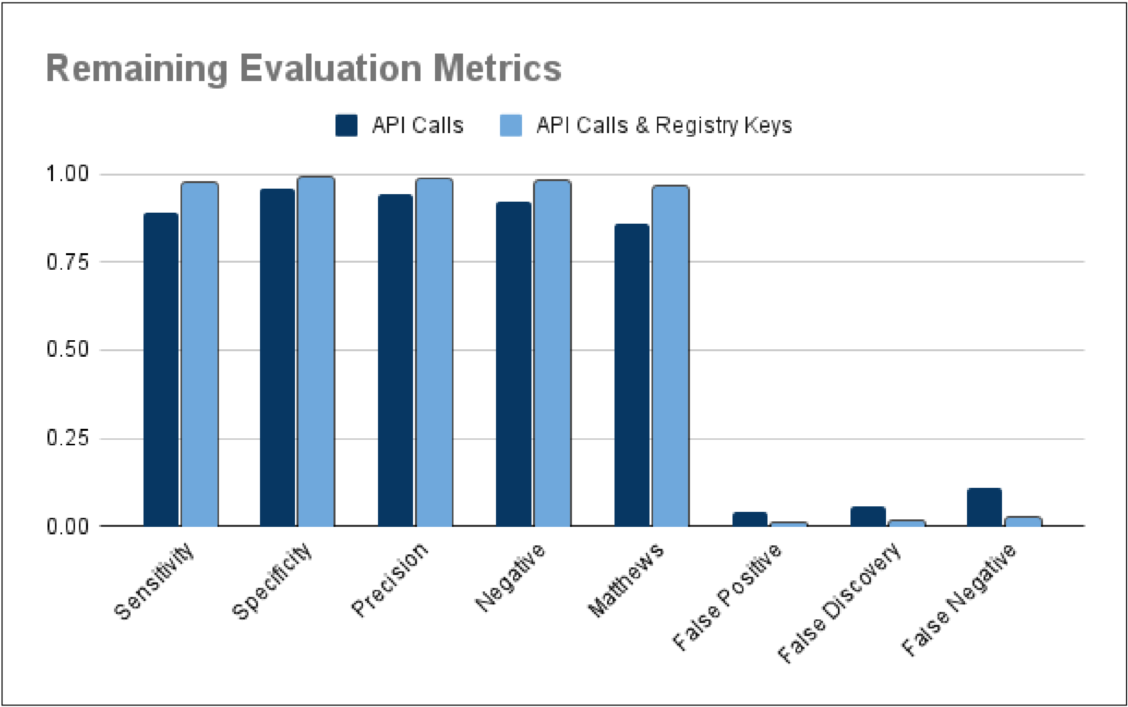
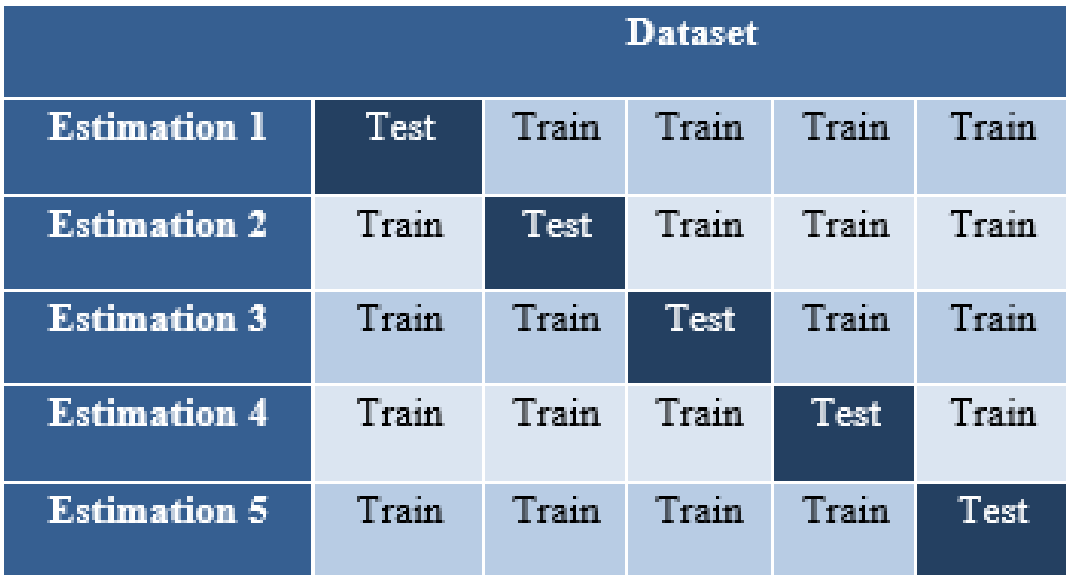


Figure 5.3: Remaining Evaluation Metrics



	Dataset				
Estimation 1	Test	Train	Train	Train	Train
Estimation 2	Train	Test	Train	Train	Train
Estimation 3	Train	Train	Test	Train	Train
Estimation 4	Train	Train	Train	Test	Train
Estimation 5	Train	Train	Train	Train	Test

Figure 5.4: 5 Fold Cross Validation

5.5 Comparison

Table 5.1 provides a summary of the performance reported in existing studies of malware detection. It can be seen that LRDPA provides an accuracy of 0.44% higher than the closest studies that uses only one ML model [37]. Although [38] reports marginally better results, it uses a ensemble classifier instead of a single ML model. It is also expected that the accuracy of LRDPA will improve with the use of addition pre-processing and feature selection.

5.6 Summary

In this chapter, LRDPA is evaluated. In the first scenario, the metrics are obtained when the model is trained using only the APIs called data. In the second scenario, the evaluation is presented when the model is trained using Registry keys triggered data. Lastly, the best results are presented using combined data to train the LRDPA model. The highest accuracy obtained is 99.44%. The highest F1 score of 98.14% is also obtained with combined data. Also, only 1 False Negative (FN) and 0 False Positives (FP) are obtained. These results are very close to those presented in existing literature and can be easily improved through additional pre-processing and feature selection.

Table 5.1: Summary of Existing Work on ML-based Detection of Malware

Ref.	**Type of Malware**	**ML Algorithm(s)**	**Data Analysed**	**Performance**
[27]	Malware	Ensemble Classifier	Registry Behaviour	Accuracy = 98.43%
[28]	Ransomware	Artificial Neural Network (ANN)	Static and Dynamic	Accuracy = 95% (static) = 55% (dynamic)
[29]	Ransomware	Random Forest (RF)	Hex Codes	Accuracy = 88.5%
[30]	Ransomware	Logistic Regression (LR)	APIs, Registry Keys, and File System Operations	Detection Rate = 96.3%
[32]	Ransomware	Support Vector Machine (SVM)	APIs	Accuracy = 97.48%
[33]	Ransomware	Bidirectional Encoder Representations from Transformers (BERT)	Contextual Behaviour	F1-score = 99.5%
[38]	Locker Ransomware	Ensemble Classifier (SVM, DT, RF, LR)	Behavioural Characteristics	Accuracy = 99.98%

Chapter 6

Conclusion and Future Work

6.1 Conclusion

The last few years has seen a surge in incidents involving ransomware particularly zero-day attacks involving locker ransomware. It is imperative that this type of malware is thoroughly analysed so that efficient and reliable detection and prediction methods can be developed.

The most promising approach is based on supervised machine learning using a dataset developed from dynamic analysis of malware. To the best of our knowledge, no comprehensive dataset of locker ransomware has been created yet. Likewise, no ML model has been developed for the detection and prediction of locker ransomware. This study aims to fill this gap by developing the Locker Ransomware Detection and Prediction Algorithm (LRDPA).

The first step in the methodology used to develop LRDPA was the collection of samples. A total of 382 samples of benign software and 505 samples of locker ransomware were collected from various sources. These samples were labeled as either "Benign Software" or "Locker Ransomware" with the help of *VirusTotal*.

LRDPA consists of three stages. In the first stage, the hash digest of an input sample is generated and compared with a signature database containing hash digests of known locker ransomware. This stage enables the efficient signature-based detection of known samples of locker ransomware. If no match is found, then LRDPA moves onto the second stage of dynamic analysis. The input sample is executed in the Cuckoo sandbox environment and all APIs called and Registry keys triggered are recorded. In the third stage, the recorded data is fed into a trained RF model to predict if the input sample is benign software or locker ransomware.

As the accuracy of LRDPA depends on the data used to train the RF model, each sample of benign software and malicious software was dynamically analysed by running it in the Cuckoo sandbox environment. All APIs called and Registry keys triggered were compiled into a dataset.

An analysis of this dataset showed that the profile of APIs called and Registry keys triggered by locker ransomware is very distinct from that of benign software. In fact, 1 API and 7 Registry keys was found to be unique to only locker ransomware and were triggered by 70% of the locker ransomware samples. Likewise, 4 APIs and 121 Registry keys had a contrasting percentage of more than 50%. This data analysis provided strong assurance that the dataset consisting of 282 APIs called and 21,780 Registry keys triggered would be suitable to train and test a ML model for the detection and prediction of locker ransomware.
Three configurations for the dataset were created: only API data, only Registry key data, and combined data. For each configuration, the dataset was split into 80:20 (80% of data for the training and 20% for the testing). The trained RF model was then evaluated (with 10-fold cross-validation) using multiple metrics. The results clearly showed that the combined data configuration for LRDPA provided the highest accuracy (99.44%) and F1 score (99.38%). Only 1 False Negative (FN) and 0 False Positives (FP) were recorded. The other metrics also confirmed that LRDPA performs best with both API and Registry key data.

6.2 Future Work

In the future, LRDPA can be modified to include a variety of machine learning algorithms, such as regression and deep convolutional neural networks. In addition to that, mapping and homogenising of the existing collection of features to the most prominent characteristics in order to employ them may also help enhance the outcomes (e.g. accuracy). Feature selection method, has the potential to highlight the most essential aspects of both supervised and unsupervised learning, with the additional advantage of making the model more effective.

Additionally the developed dataset can be expanded by including other relevant classes of ransomware (e.g. hybrid ransomware) and additional families of locker ransomware.

Bibliography

[1] The 2021 ransomware survey report. https://www.fortinet.com/content/dam/fortinet/assets/reports/report-ransomware-survery.pdf, August 2021.

[2] Laurie Lacono Devon Ackerman, Keith Wojceiszek. Kroll ransomware attack trends – 2020 ytd. https://www.kroll.com/en/insights/publications/cyber/ransomware-attack-trends-2020, October 2020.

[3] Microsoft Windows. Programming reference for the win32 api. https://docs.microsoft.com/en-us/windows/win32/api/.

[4] Mamoona Humayun, NZ Jhanjhi, Ahmed Alsayat, and Vasaki Ponnusamy. Internet of things and ransomware: Evolution, mitigation and prevention. *Egyptian Informatics Journal*, 22(1):105–117, 2021. ISSN 1110-8665. doi: https://doi.org/10.1016/j.eij.2020.05.003. URL https://www.sciencedirect.com/science/article/pii/S1110866520301304.

[5] Gandhi Krunal A. and Patel Viral Kumar D. Survey on ransomware: A new era of cyber attack. *International Journal of Computer Applications*, 168(3):38–41, Jun 2017. ISSN 0975-8887. doi: 10.5120/ijca2017914446. URL http://www.ijcaonline.org/archives/volume168/number3/27859-2017914446.

[6] fortinet. The 2021 ransomware survey report. https://www.fortinet.com/content/dam/maindam/PUBLIC/02_MARKETING/08_Report/report-ransomware-survery.pdf, August 2021.

[7] Frank Dickson Christopher Kissel. Idc's 2021 ransomware study: Where you are matters! https://www.idc.com/getdoc.jsp?containerId=US48093721, July 2021.

[8] MalwareHunterTeam. Id ransomware. https://id-ransomware.malwarehunterteam.com/.

[9] Jon Clay. The latest on ransomware. https://resources.trendmicro.com/2019-NABU-WBN-Latest-Ransomware.html, June 2019.

[10] Rob Masse. Taking data hostage: The rise of ransomware. https://www2.deloitte.com/content/dam/Deloitte/ca/Documents/risk/ca-en-risk-Ransomware_POV_noCM_AODA.PDF, January 2022.

[11] Lena Y. Connolly and David S. Wall. The rise of crypto-ransomware in a changing cybercrime landscape: Taxonomising countermeasures. *Computers Security*, 87:101568, 2019. ISSN 0167-4048. doi: https://doi.org/10.1016/j.cose.2019.101568. URL https://www.sciencedirect.com/science/article/pii/S0167404819301336.

[12] T.R. Reshmi. Information security breaches due to ransomware attacks - a systematic literature review. *International Journal of Information Management Data Insights*, 1(2):100013, 2021. ISSN 2667-0968. doi: https://doi.org/10.1016/j.jjimei.2021.100013. URL https://www.sciencedirect.com/science/article/pii/S2667096821000069.

[13] The BlackBerry Research Intelligence Team. New ransomware family identified: Lokilocker raas targets windows systems. https://blogs.blackberry.com/en/2022/03/lokilocker-ransomware, March 2022.

[14] Per Håkon Meland, Yara Fareed Fahmy Bayoumy, and Guttorm Sindre. The ransomware-as-a-service economy within the darknet. *Computers Security*, 92:101762, 2020. ISSN 0167-4048. doi: https://doi.org/10.1016/j.cose.2020.101762. URL https://www.sciencedirect.com/science/article/pii/S0167404820300468.

[15] Dehghantanha A. Bahrami P.N. et al. Dargahi, T. A cyber-kill-chain based taxonomy of crypto-ransomware features. *Journal of Computer Virology and Hacking Techniques*, 15:277–305, 2019. doi: https://doi.org/10.1007/s11416-019-00338-7. URL https://link.springer.com/article/10.1007/s11416-019-00338-7#citeas.

[16] Lena Yuryna Connolly, David S Wall, Michael Lang, and Bruce Oddson. An empirical study of ransomware attacks on organizations: an assessment of severity and salient factors affecting vulnerability. *Journal of Cybersecurity*, 6(1), 12 2020. ISSN 2057-2085. doi: 10.1093/cybsec/tyaa023. URL https://doi.org/10.1093/cybsec/tyaa023. tyaa023.

[17] Cybersecurity: Ransomware alert. https://www.sec.gov/files/risk-alert-cybersecurity-ransomware-alert.pdf, May 2017.

[18] Aaron Zimba and Mumbi Chishimba. On the economic impact of crypto-ransomware attacks: The state of the art on enterprise systems. *European Journal for Security Research*, 4, 04 2019. doi: 10.1007/s41125-019-00039-8.

[19] Eduardo Berrueta, Daniel Morato, Eduardo Magaña, and Mikel Izal. A survey on detection techniques for cryptographic ransomware. *IEEE Access*, PP:1–1, 10 2019. doi: 10.1109/ACCESS.2019.2945839.

[20] Masarah Paquet-Clouston, Bernhard Haslhofer, and Benoît Dupont. Ransomware payments in the Bitcoin ecosystem. *Journal of Cybersecurity*, 5(1), 05 2019. ISSN 2057-2085. doi: 10.1093/cybsec/tyz003. URL https://doi.org/10.1093/cybsec/tyz003. tyz003.

[21] Ezhil Kalaimannan, Sharon K. John, Theresa DuBose, and Anthony Pinto. Influences on ransomware's evolution and predictions for the future challenges. *Journal of Cyber Security Technology*, 1(1):23–31, 2017. doi: 10.1080/23742917.2016.1252191. URL https://doi.org/10.1080/23742917.2016.1252191.

[22] Sudhir Kumar Pandey and B.M. Mehtre. A lifecycle based approach for malware analysis. In *2014 Fourth International Conference on Communication Systems and Network Technologies*, pages 767–771, 2014. doi: 10.1109/CSNT.2014.161.

[23] Bander Ali Saleh Al-rimy, Mohd Aizaini Maarof, and Syed Zainudeen Mohd Shaid. Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions. *Computers Security*, 74:144–166, 2018. ISSN 0167-4048. doi: https://doi.org/10.1016/j.cose.2018.01.001. URL https://www.sciencedirect.com/science/article/pii/S016740481830004X.

[24] Rhyme Upadhyaya and Aruna Jain. Cyber ethics and cyber crime: A deep dwelved study into legality, ransomware, underground web and bitcoin wallet. In *2016 International Conference on Computing, Communication and Automation (ICCCA)*, pages 143–148, 2016. doi: 10.1109/CCAA.2016.7813706.

[25] James Wyke & Anand Ajjan. The current state of ransomware. https://www.sophos.com/en-us/medialibrary/PDFs/technical%20papers/sophos-current-state-of-ransomware.pdf, December 2015.

[26] Joshua Saxe and Konstantin Berlin. expose: A character-level convolutional neural network with embeddings for detecting malicious urls, file paths and registry keys, 2017. URL https://arxiv.org/abs/1702.08568.

[27] Abadi M. Tajoddin, A. Ramd: registry-based anomaly malware detection using one-class ensemble classifiers, 2019. URL https://link.springer.com/article/10.1007/s10489-018-01405-0#citeas.

[28] Syed Ibrahim Imtiaz, Saif ur Rehman, Abdul Rehman Javed, Zunera Jalil, Xuan Liu, and Waleed S. Alnumay. Deepamd: Detection and identification of android malware using high-efficient deep artificial neural network. *Future Generation Computer Systems*, 115:844–856, 2021. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2020.10.008. URL https://www.sciencedirect.com/science/article/pii/S0167739X2032985X.

[29] Bheemidi Vikram Reddy, Gutha Jaya Krishna, Vadlamani Ravi, and Dipankar Dasgupta. Machine learning and feature selection based ransomware detection using hexacodes. In Vikrant Bhateja, Sheng-Lung Peng, Suresh Chandra Satapathy, and Yu-Dong Zhang, editors, *Evolution in Computational Intelligence*, pages 583–597, Singapore, 2021. Springer Singapore.

[30] Ping Wang and Yu-Shih Wang. Malware behavioural detection and vaccine development by using a support vector model classifier. *Journal of Computer and System Sciences*, 81(6):1012–1026, 2015. ISSN 0022-0000. doi: https://doi.org/10.1016/j.jcss.2014.12.014. URL https://www.sciencedirect.com/science/article/pii/S0022000014001780. Special Issue on Optimisation, Security, Privacy and Trust in E-business Systems.

[31] Seunghwan Lee Kichang Kim Jinsoo Hwang, Jeankyung Kim. Two-stage ransomware detection using dynamic analysis and machine learning techniques, 2020. URL https://link.springer.com/article/10.1007/s11277-020-07166-9#citeas.

[32] Yuki Takeuchi, Kazuya Sakai, and Satoshi Fukumoto. Detecting ransomware using support vector machines. In *Proceedings of the 47th International Conference on Parallel Processing Companion*, ICPP '18, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450365239. doi: 10.1145/3229710.3229726. URL https://doi.org/10.1145/3229710.3229726.

[33] Muhammad Ejaz Ahmed, Hyoungshick Kim, Seyit Camtepe, and Surya Nepal. Peeler: Profiling kernel-level events to detect ransomware. In Elisa Bertino, Haya Shulman, and Michael Waidner, editors, *Computer Security – ESORICS 2021*, pages 240–260, Cham, 2021. Springer International Publishing.

[34] Lorenzo Fernández Maimó, Alberto Huertas Celdrán, Ángel L. Perales Gómez, Félix J. García Clemente, James Weimer, and Insup Lee. Intelligent and dynamic ransomware spread detection and mitigation in integrated clinical environments. *Sensors*, 19(5), 2019. ISSN 1424-8220. doi: 10.3390/s19051114. URL https://www.mdpi.com/1424-8220/19/5/1114.

[35] Muhammad Ijaz, Muhammad Hanif Durad, and Maliha Ismail. Static and dynamic malware analysis using machine learning. In *2019 16th International bhurban conference on applied sciences and technology (IBCAST)*, pages 687–691. IEEE, 2019.

[36] Bander Ali Saleh Al-rimy, Mohd Aizaini Maarof, Yuli Adam Prasetyo, Syed Zainudeen Mohd Shaid, and Asmawi Fadillah Mohd Ariffin. Zero-day aware decision fusion-based model for crypto-ransomware early detection. 10, Nov. 2018. URL https://publisher.uthm.edu.my/ojs/index.php/ijie/article/view/2828.

[37] Arthur Moses Opio. *Detection of WannaCry Ransomware using machine learning techniques*. PhD thesis, Makerere University, 2022.

[38] Dan Su, Jiqiang Liu, Xiaoyang Wang, and Wei Wang. Detecting android locker-ransomware on chinese social networks. *IEEE Access*, 7:20381–20393, 2019. doi: 10.1109/ACCESS.2018.2888568.

[39] Luis Lizama, Leonardo Arrieta, Flor Mendoza, Luis Servín, and Eric Simancas-Acevedo. Public hash signature for mobile network devices. *Ingeniería Investigación y Tecnología*, 20:1–10, 04 2019. doi: 10.22201/fi.25940732e.2019.20n2.018.

[40] Virustotal. https://www.virustotal.com/gui/home/upload.

[41] Wu Liu, Ping Ren, Ke Liu, and Hai-xin Duan. Behavior-based malware analysis and detection. In *2011 First International Workshop on Complexity and Data Mining*, pages 39–42, 2011. doi: 10.1109/IWCDM.2011.17.

[42] Muhammad Ijaz, Muhammad Hanif Durad, and Maliha Ismail. Static and dynamic malware analysis using machine learning. In *2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, pages 687–691, 2019. doi: 10.1109/IBCAST.2019.8667136.

[43] Masoume Aghaeikheirabady, Seyyed Mohammad Reza Farshchi, and Hossein Shirazi. A new approach to malware detection by comparative analysis of data structures in a memory image. In *2014 International Congress on Technology, Communication and Knowledge (ICTCK)*, pages 1–4, 2014. doi: 10.1109/ICTCK.2014.7033519.

[44] Guoping Zeng. On the confusion matrix in credit scoring and its analytical properties. *Communications in Statistics - Theory and Methods*, 49(9):2080–2093, 2020. doi: 10.1080/03610926.2019.1568485. URL https://doi.org/10.1080/03610926.2019.1568485.

[45] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 233–240, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933832. doi: 10.1145/1143844.1143874. URL https://doi.org/10.1145/1143844.1143874.

[46] Dell Zhang, Jun Wang, and Xiaoxue Zhao. Estimating the uncertainty of average f1 scores. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, ICTIR '15, page 317–320, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450338332. doi: 10.1145/2808194.2809488. URL https://doi.org/10.1145/2808194.2809488.

[47] Jurman G. Chicco, D. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *Communications in Statistics - Theory and Methods*. doi: 10.1186/s12864-019-6413-7. URL https://doi.org/10.1186/s12864-019-6413-7.

[48] Security Ninja. Windows functions in malware analysis. https://resources.infosecinstitute.com/topic/windows-functions-in-malware-analysis-cheat-sheet-part-2/, June 2015.

[49] Security Ninja. Ntapi undocumented functions. http://undocumented.ntinternals.net/index.html?page=UserMode%2FUndocumented%20Functions%2FNT%20Objects%2FThread%2FThread%20Context%2FNtSetContextThread.html, December 2000.

[50] Microsoft. Globalmemorystatusex function (sysinfoapi.h). https://docs.microsoft.com/en-us/windows/win32/api/sysinfoapi/nf-sysinfoapi-globalmemorystatusex, November 2021.

[51] Microsoft. Drawtextexw function (winuser.h). https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-drawtextexw, November 2021.

[52] Microsoft. Createactctxw function (winbase.h). https://docs.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-createactctxw, November 2021.

[53] Microsoft. Ntsetinformationfile function (ntifs.h). https://docs.microsoft.com/en-us/windows-hardware/drivers/ddi/ntifs/nf-ntifs-ntsetinformationfile, November 2022.

[54] Microsoft. Zwsetvaluekey function (wdm.h). https://docs.microsoft.com/en-us/windows-hardware/drivers/ddi/wdm/nf-wdm-zwsetvaluekey, April 2022.

[55] Microsoft. Shellexecuteexw function (shellapi.h). https://docs.microsoft.com/en-us/windows/win32/api/shellapi/nf-shellapi-shellexecuteexw, November 2021.

[56] Ntapi undocumented functions. http://undocumented.ntinternals.net/index.html?page=UserMode%2FUndocumented%20Functions%2FHardware%2FNtShutdownSystem.html.