A CRYPTO-RANSOMWARE DETECTION MODEL FOR THE PRE-ENCRYPTION STAGE USING RANDOM FOREST ALGORITHM

SUBMITTED BY:

NJOROGE, PRISCILLAH WANGUI

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF MASTER OF SCIENCE IN DATA COMMUNICATIONS IN THE FACULTY OF COMPUTING AND INFORMATION MANAGEMENT AT KCA UNIVERSITY

NOVEMBER, 2022

DECLARATION

I declare that this dissertation is my original work and has not been previously published or submitted elsewhere for award of a degree. I also declare that this contains no materials written or published by other people except where due reference is made and author duly acknowledged.

Student Name: Njoroge, Priscillah Wangui Reg No.: 14/04352

Sign: _____ Date: 11/11/2022

I do hereby confirm that I have examined the Masters dissertation of

Njoroge, Priscillah

And have approved it for examination

Sign: _____ Date: 14/11/2022

Dr. Stephen Njenga

Dissertation Supervisor

ABSTRACT

Cryptographic ransomware is a challenging cybersecurity threat that encrypts the victim's files and demands a ransom in exchange for the decryption key. Traditional signature-based protection methods, such as antivirus and anti-malware, have proven in-effective at preventing crypto-ransomware attacks, therefore the production of ransomware is on the rise. Additionally, crypto-ransomware incorporates advanced encryption algorithms causing irreversible effects even if the victim chooses to pay the ransom. Given the magnitude and variety of threats we face today, it is critical to have solutions in place to effectively analyse and detect crypto-ransomware attacks during the pre-encryption stage before encryption happens. Only if these threats are identified during the pre-encryption phase can they be adequately mitigated. Existing methods for early detection of crypto-ransomware rely on a timing thresholding methodology to set the border of the pre-encryption stage. However, the fixed time threshold strategy, suggests that the samples begin encryption at the exact moment. This is not always the case since timing varies between crypto-ransomware families as a result of the obfuscation techniques used to evade detection. Furthermore, scarcity of data during an attack's initial stages reduces the ability of feature extraction algorithms in early detection solutions to discover attack features lowering detection accuracy. This research therefore, proposed development of a Dynamic Crypto-Ransomware Detection Model (DCRDM). DCRDM monitors the pre-encryption stage for every case separately relying on the initial appearance of any APIs related to cryptography to establish the pre-encryption stage boundary, whereby features are extracted and used in training a prediction model using the Random Forest machine learning algorithm. The sample data was obtained from widely used ransomware repositories. The model achieved a detection accuracy of 98.6% with False Positive Rate of 1.9%.

Keywords: encryption, cryptographic ransomware, signature-based, machine learning, Random Forest, early detection, pre-encryption stage

ACKNOWLEDGEMENTS

To God be the glory for the wonderful things He has done. This endeavour has been completed through His grace and mercies.

I am grateful to my family and friends for their unwavering support and understanding. Special thanks to my mum and late dad, who pushed me and longed to see me finish this thesis. They truly are the ultimate role models.

My sincere gratitude to my supervisor, Dr. Stephen Njenga, whose advice and constant guidance helped shape this dissertation. Thank you also to the entire team at KCA University's School of Technology for your selfless efforts to help us grow professionally and complete our course.

I thank you all most sincerely.

DEDICATION

This work is especially dedicated to my late dad Geoffrey Njoroge, my mum Eunice Njoki, and my entire family for their love, moral support, and understanding throughout the course of this study.

ACRONYMS AND ABBREVIATIONS

AES	Advanced Encryption Standard
API	Application Programming Interface
AUC	Area under the Curve
CIAAA	Confidentiality, Integrity, Availability, Authenticity, and Accountability
COVID	Coronavirus Disease
CPU	Central Processing Unit
CSPRNG	Cryptographically Secure Pseudo-Random Number Generator
CSV	Comma-Separated Value
DCRDM	Dynamic Crypto-Ransomware Detection Model
DLL	Dynamic Link Library
DNGA	Domain Name Generation Algorithm
DT	Decision Tree
ECDH	Elliptic-curve Diffie–Hellman
FN	False Negative
FNR	False Negative Rate
FP	False Positive
FPR	False Positive Rate
IT	Information Technology
JSON	JavaScript Object Notation
KNN	K-Nearest Neighbor
ML	Machine Learning
MLP	Multi-Layer Perceptron
NN	Neural Network
OS	Operating System
PC	Personal Computer
PRNG	Pseudo-Random Number Generator
RDP	Remote Desktop Protocol
RF	Random Forest
RRF	Rocchio Relevance Feedback
ROC	Receiver Operating Characteristic
RSA	Rivest, Shamir and Adleman
SHA	Secure Hash Algorithm
SVM	Support vector machine
TF-IDF	Term Frequency-Inverse Document Frequency
TN	True Negative
TP	True Positive
TPR	True Positive Rate
TZ	TheZoo
URL	Uniform Resource Locator
USB	Universal Serial Bus
VS	VirusShare
VSM	Vector-Space Model

GLOSSARY

Cryptography:	conversion of data into scrambled text to conceal its readability and meaning, and deciphering it using a key
Cryptographic API or CryptoAPI:	is an application programming interface that provides services that allow developers to secure applications using cryptography.
Cryptographic Ransomware:	a type of malware that encrypts its victim's files and holds them hostage for ransom.
Dynamic analysis:	the process of activating the ransomware in a controlled environment
Dynamic thresholding:	values that analyse data and generate events when a violation occurs vary from case to case.
Machine learning:	a branch of artificial intelligence (AI) that enables systems to learn and improve based on their own experiences without being explicitly programmed.
Model:	to create a representation of something on a small scale so as to base your predictions of the future outcome.
Random forest:	a supervised learning algorithm that randomly creates and merges multiple decision trees into one “forest.”
Ransomware:	a type of malicious software (malware) designed to block access to a computer system until a sum of money (ransom) is paid.
Static analysis:	the process of analysing malicious files without executing them.
Static thresholding:	Manually-defined static values at specific intervals, which analyse data and generate events when a violation occurs
Supervised learning:	an algorithm that learns from labelled training data to help predict outcomes for unforeseen data.
Zero-Day:	unknown or newly discovered.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE: INTRODUCTION

1.1 Background of the Study

In recent years, almost every member of society has used the internet in their daily lives. This is because the internet has become an integral part of our daily lives, facilitating social interactions, online banking, and marketing, among other things. The surge of online activity presents both opportunities and challenges for individuals and organizations as the number of online extortions escalates. Cyber criminals utilize malicious software (Malware) to launch cyber-attacks against their victims' machines. One malware of major concern is the ransomware.

Ransomware is a “type of malicious software (malware) that once executed on a computer system, prevents the user from accessing the computer or its data and demands payment (ransom) in exchange for the computer's restoration” (Nieuwenhuizen, 2017).

Ransomware has grown dramatically in recent years. It has quickly become a serious threat due to its aggressive and destructive mode of operation. This malware has targeted not just ordinary people, but also governments and businesses in practically every industry. Companies, banks, cloud providers, manufacturers, governments, hospitals, schools, and even police departments have all been victims of ransomware attacks around the world (Sahay, Sharma, & Rathore, 2020).

The first ransomware families primarily targeted end users, owing to a lack of security awareness. But as the malware evolved, many organizations were frequently targeted. In these types of attacks, cybercriminals pre-select their targets and try to create as much disruption in the hopes of receiving a large payment for ransom (Oz, Aris, Levi & Uluagac, 2021). According to a 2016 survey of 290 firms from a number of industries in the United States, Germany, Canada, and the United Kingdom, almost half of them had been victims of ransomware attacks in the preceding year, (Osterman Research and Inc., 2016), with roughly 40 percent stating that they had paid the ransom.

Currently, ransomware attacks impede computer operation in two ways: by preventing access to the computer (locker ransomware) and by using encryption techniques to render user data unusable (crypto-ransomware) (Nieuwenhuizen, 2017)

Unlike locker ransomware whose harm can be circumvented with ease, the consequences of crypto-ransomware are irreversible because it uses encryption on user files. Subsequently, the victim will find it challenging, if not impractical, to retrieve their data

without the decryption key. As a result, detecting crypto-ransomware attacks as soon as possible, before they encrypt user files and data, is critical (Homayoun et al., 2017).

In 2016, The European Union's law enforcement agency (Europol) professed that crypto-ransomware had become “the most prominent malware threat, overshadowing data stealing malware and banking Trojans…. becoming a key threat for citizens and enterprises alike,” (Wainwright, 2016). The malware with the highest profit margins in history has been identified as encrypting ransomware, with several hundred millions of dollars believed to be extorted from victims annually (Morato, Berrueta, Magaña, & Izal, 2018).

The infamous AIDS Trojan, which first appeared in 1989, was most likely the first ransomware. It was distributed to conference attendees on a floppy disk, and the software encrypted file names before displaying a demand for payment to a Panama location (Morato et al., 2018).

Ransomware, which was initially thought to be a simple Trojan horse program evolved into a sophisticated family of malware in September 2013 with the appearance of CryptoLocker, the first variant of crypto-ransomware, which infected approximately 500,000 machines and collected about $3 million from victims. Following CryptoLocker's success, subsequent ransomware variants adopted its business model (Morato et al., 2018). WannaCry ransomware made headlines in May 2017 when it attacked approximately 300,000 machines in approximately 150 countries, including hospitals, banks, telecommunication companies, and manufacturers, among others. According to reports from the Communications Authority of Kenya (CA), the Wannacry ransomware hit at least 19 organizations in Kenya. NotPetya launched another devastating attack in June 2017, this time costing an estimated $10 billion globally. (Greenberg, 2018).

PureLocker ransomware first appeared in 2018. The crypto ransomware was meticulously designed to avoid detection by masking dangerous or suspicious behaviour in sandbox environments, impersonating the Crypto++ cryptography library, and employing features common in music playback libraries. (Morato et al., 2018).

Another devastating crypto ransomware named SamSam appeared in late 2018. It specialized in targeted ransomware attacks, infiltrating networks and encrypting multiple machines within a company before demanding a large sum of money SamSam caused a shut down in the city of Atlanta after encrypting multiple municipal systems. It was demanding ransom in Bitcoin (Oz et al., 2021).

The ransomware situation evolved substantially in 2020. The attacks grew in quantity, scale, and complexity. An increase of more than 150 percent was seen in ransomware attacks.

(Greengard, 2021). According to the Sophos 2021 threat report, 37 percent of respondents' organizations were affected by ransomware attacks in 2020, based on a survey of 5400 IT Managers from 30 countries (Sophos, 2021).

The surge in popularity of remote work as a result of the pandemic resulted in many people accessing work-related data and corporate systems via relatively insecure private networks and personal devices (Greengard, 2021).

Additionally, criminal groups frequently used COVID-19-themed phishing lures to prey on individuals' fears about the pandemic (Greengard, 2021). Given the high degree of worry, users were more likely to click on COVID-19 themed ransomware bait emails. For example

- Fraudulent financial claims of government assistance during the economic shutdown.
- Free downloads for in-demand technological products like video and audio conferencing platforms.
- Information on vaccines, masks, and other items

Moreover, the number of ransomware attacks on health-care institutions increased dramatically, and a new crypto ransomware variant known as Corona was discovered. This variant targeted hospitals and encrypted patient health records. It then displayed a COVID-19-themed ransom message and demanded Bitcoin payment (Oz et al., 2021).

It is estimated that by 2031, organizations will lose around $265 billion to ransomware attacks, with a new victim every 2 seconds (Oz et al., 2021). However, there is no guarantee that the files will be recovered, even if the ransom is paid.

When organizations are infected by ransomware, the costs include not only significant financial losses, but also the disruption of operations, data corruption or loss, time spent cleaning up the infection and dealing with the fallout, the impact on the company's public image, and legal and insurance implications.

Several enablers, primarily technological advancements, attract cybercriminals to ransomware. These factors contribute to ransomware’s continuous development and popularity. The first factor is the widespread use of the Internet, which allows for global connectivity. Although this facilitates communication, it also raises the risk of cross-border cyber-attacks (Diro et al., 2020). A cybercriminal, for example, can launch an attack against a targeted firm in Kenya from anywhere in Russia. Because of the geographical distance and the various regulations that govern each country, this multinational attack will complicate matters for the authorities.

The second factor is the widespread acceptance of cryptocurrencies like the well-known Bitcoin. The introduction of Bitcoin undeniably inspired massive ransomware attacks. This primarily owes to the fact that these currencies impede traceability, allowing the receiver to remain anonymous to the authorities (Paquet-Clouston, Haslhofer, & Dupont, 2019). As a result, cyber-criminals have a safe and secure method of receiving ransom payments.

The availability of encryption algorithms is another factor. Encryption is an important technology that helps an information system achieve its security objectives, which include confidentiality, integrity, availability, authenticity, and accountability (CIAAA), as defined by the security community (Young &Yung, 2005). With the exception of availability, encryption helps in achieving these goals. However, in crypto-ransomware attacks, encryption is used against users for malicious purposes such as capturing data for ransom. According to Young & Yung (2005), denial-of-service and extortion attacks can be launched using public-key cryptography by a malicious program. They demonstrated how a malicious software can exploit Microsoft's cryptographic libraries, which are included in the Windows Operating System, to encrypt user's documents for extortion purposes. Cybercriminals gain from malware's use of cryptography in various ways. It provides the malicious program with complete privacy as well as anonymity while criminals communicate with malware. (Abidin, Kumar &Tiwari, 2012). Types of encryption algorithms employed by crypto-ransomware will be discussed later.

Another enabler is the ease with which ransomware can be obtained. The Dark Web is full of ransomware development kits like TOX, Hidden Tear, Torlocker among others that can be obtained for free or at a low cost (Kharraz, Robertson, Balzarotti, Bilge & Kirda, 2015). There's also Ransomware as a Service (RaaS), which is a profit-sharing plan between cybercriminals in which one party writes the ransomware source-code and the other distributes it to prospective victims' by using malicious distribution services for a fee to swiftly spread ransomware around the world via platforms such as spam e-mails, drive-by-downloads and malvertising. (Kok & Jhanjhi, 2020). These platforms provide a cost-effective way of ransomware distribution.

The conventional malware detection methods, especially signature-based solutions, are ineffective for detecting crypto-ransomware attacks, owing to their reliance on malicious signature repositories of previously known malware, meaning that they cannot identify zero-day (unknown) attacks. Detection methods capable of dealing with zero-day ransomware attacks strive to detect the infection based on broader features such as ransomware-specific operations rather than just file signatures alone. In this regard, several strategies for detecting

crypto-ransomware attacks have been proposed; they can be classified as data-centric or process-centric (Al-Rimy et al., 2021).

The data-centric solutions monitor the victim's computer's digital assets and sound an alarm if any suspicious changes are found (Al-Rimy et al., 2021). They examine the file structure changes to see whether they are suspicious. They employ techniques such as contents similarity measures, file entropy, and decoy strategies to keep track of the structure of the file both prior to and after access (Al-Rimy et al., 2021, Kharaz et al., 2016). The Process-centric solutions, are classified into two groups. One, by monitoring system resources as memory, CPU, I/O buffer, and network, and raising an alarm when certain encryption-related events are encountered (Kharaz et al., 2016). Two, by monitoring the running process's behaviour and gathering various forms of behavioural data which are subsequently utilized to train different machine learning classifiers (Al-Rimy et al., 2021).

In recent years, there has been a rise in interest in using machine learning (ML) to detect malware automatically based on its dynamic behaviours. Because of their self-learning capabilities, ML approaches have been deemed effective, and they have significantly improved detection rates on a wide scale when compared to conventional malware analysis method. (Chen, Yang, Paul & Sahita, 2018).

"Machine learning," according to artificial intelligence pioneer Arthur Samuel, is a “set of methods that gives computers the ability to learn without being explicitly programmed.” (Naqa & Murphy, 2015). A machine learning system finds and recognizes the principles that underpin the data it encounters. The algorithm can utilize this information to 'reason' the features of samples it has never seen before, who’s hidden behaviour could be malicious or benign.

The concept of developing machine learning-based detection models utilizing data extracted at the beginning of crypto-ransomware attacks was proposed by Sgandurra et al. (2016). The authors recommended a fixed time-based threshold method to specify the amount of data necessary, in which data acquired during the first 30 seconds of ransomware execution was gathered and utilized to develop an early detection model. Similarly, Homayoun et al. (2017) lowered the threshold to 10 seconds, and Rhode et al. (2018) reduced it to 5 seconds.

This study proposes the concept of dynamic thresholding to describe a stage in the lifecycle of crypto-ransomware before encryption, i.e. the pre-encryption stage. The proposed approach differs from fixed thresholding in that it monitors the pre-encryption stage for every case separately relying on the initial appearance of any APIs related to cryptography to

establish the pre-encryption phase boundary. That is, rather than a time-based criteria, this approach employs a threshold determined by the API calls related to cryptography.

The runtime data features generated by the crypto-ransomware attacks during the pre-encryption phase are then extracted and used in training a prediction model using the Random Forest (RF) machine learning algorithm, a supervised machine learning technique widely utilized in numerous research (Homayoun et al., 2017; Singh et al., 2014). Because RF is built on the principle of ensemble learning, rather than relying solely on one decision tree, the random forest uses each tree's predictions to predict the final result based on the prediction with the most votes (Rhode, 2021).

1.2 Statement of the Problem

Crypto-ransomware is one of today's most dangerous types of malware. Once infected, the malware encrypts the victim's data and blocks access until ransom is paid, resulting in multi-million dollar cyber-extortion each year. This type of malware has caught the interest of cybercriminals due to numerous success stories with global ramifications, such as cryptowall, Wannacry, and NotPetya (Oz et al., 2021). Because of the huge level of interest, plenty of other new crypto-ransomware have been developed, as well as improving existing ransomware with new variants.

Crypto-ransomware is distinguished by its irreversible effect even after detection and removal. As such, early detection is critical to safeguarding user data and files from being held for ransom. Cybercriminals have refined crypto-ransomware attack aspects such as greater encryption algorithms, worm-like capabilities, pseudo-anonymous payment methods, and the availability of Ransomware-as-a-Service (RaaS) on the dark-web, which facilitates creation of new ransomware variants. (Oz et al., 2021). As a result, crypto-ransomware attacks are on the rise. However, conventional malware detection methods are ineffective for detecting crypto-ransomware (Al-Rimy et al., 2021).

Many commercial and open-source anti-crypto-ransomware solutions rely on signature-based detection methods, which are quick and accurate for detecting known malware but far too restrictive for detecting zero-day attacks (Kok et al., 2020). The signature repositories must be updated on a regular basis, and due to cybercriminals' great interest in ransomware, new variants of ransomware that can bypass antiviruses continue to emerge at a rapid pace. Despite continuous improvements or updates, malware authors maintain a one-step advantage because new variants are produced quicker than new signatures can be generated, tested, and added to malicious signature repositories (Kok et al., 2020).

Detection methods capable of dealing with zero-day crypto-ransomware attacks strive to detect the infection based on broader features such as ransomware-specific operations rather than just file signatures alone. In this regard, several strategies for detecting crypto-ransomware attacks have been proposed; they can be classified as data-centric or process-centric (Al-Rimy et al., 2021).

The data-centric solutions monitor the victim's computer's digital assets and sound an alarm if any suspicious changes are found (Al-Rimy et al., 2021). They examine the file structure changes to see whether they are suspicious. This approach, however, cannot tell if the file structure change was caused by a crypto-ransomware attack or by benign application,

resulting in a high rate of false alarms (Al-Rimy et al., 2021). Furthermore, the data-centric method may not entirely guard against ransomware attacks since it sacrifices a portion of the files before detection (Scaife et al., 2016). These files may be worth more to the victim than the other data.

Process-centric solutions, on the other hand, are classified into two groups. One, by monitoring system activities such as, file system access for example privilege elevation, network activity, resource usage and interactions with the operating system, and triggering an alarm when particular encryption-related events occur (Kharaz et al., 2016). However, relying on ad hoc incidents for crypto-ransomware attack detection raises the likelihood of false alerts because crypto-ransomware is not always the cause; benign applications can also cause them. (Al-Rimy et al., 2021). Additionally, there is no assurance that such ad hoc events will always occur prior to encryption; they may occur after encryption due to changes in attack techniques (Kharaz et al., 2016). As a result, this strategy is ineffective for early detection.

The second type of process-centric solutions monitors the running process's behaviour and gathers various forms of behavioural data which are subsequently utilized to train different machine learning classifiers. The key obstacle with the existing methods for early detection of crypto-ransomware is lack of adequate data during the initial stages of an attack, which limits the capacity of feature extraction algorithms in early detection solutions to discover attack features, resulting in data loss, low detection accuracy, and a high false-positive rate (Al-Rimy et al., 2020). Furthermore, they adopt a set time-based thresholding method to determine the pre-encryption stage borders (Scaife et al. 2016: Hwang et al. 2020).

However, the set-time thresholding strategy, suggests that the samples begin encryption at the exact moment. This is not always the case since timing varies between crypto-ransomware families as a result of the obfuscation techniques used to evade detection (Al-Rimy et al., 2020). As a result, this strategy may miss the start of the encryption operation, resulting in encryption of several files before detection (Scaife et al., 2016).

Regardless of the efforts put forth, the existing approaches, inevitably, have limitations. Crypto-ransomware is still an intricate problem that requires further study to improve current detection methods. In light of this, this work proposes development of a model capable of detecting crypto-ransomware during the pre-encryption stage, before encryption happens.

1.3 Main Objective

The main objective of this thesis is to develop a model capable of detecting crypto-ransomware during the pre-encryption stage using Random Forest algorithm.

1.4 Specific Objectives

1) To identify common characteristics and behaviours of crypto-ransomware.
2) To review the current methods for crypto-ransomware analysis and detection.
3) To develop a crypto-ransomware detection model for the pre-encryption stage using Random Forest algorithm.
4) To test and validate the effectiveness of the crypto-ransomware detection model.

1.5 Research questions

1) What are the common characteristics and behaviours of crypto-ransomware?
2) What are the current methods for crypto-ransomware analysis and detection?
3) How can a crypto-ransomware detection model for the pre-encryption stage be developed to prevent harm?
4) How can crypto-ransomware detection model be validated for effectiveness?

1.6 Significance of the Study

Crypto-ransomware has emerged as one of the most serious global cyber-security threat, owing to an increasing number of high-profile variants and rising economic losses. This malware has not only targeted ordinary people, but also governments and businesses in almost every industry. Its method of operation entails retrieving the files to be blocked, encrypting them, and then initiating an extortion process in which the victim is forced to pay for the decryption key in order to recover the files. Every day, new crypto-ransomware threats emerge, wreaking havoc on victims by causing significant financial losses, data corruption or loss, business interruption, brand damage, and legal and insurance implications.

By investigating the issues associated with encrypting ransomware and developing a dynamic thresholding strategy for defining the pre-encryption stage and detecting crypto-ransomware attacks before they encrypt data, this study fills a detection gap while also adding to the body of knowledge on crypto-ransomware. The study findings will also serve as a foundation for future research.

1.7 Motivation of the Study

Crypto ransomware has been at an all-time high for the past couple of years, with multiple variants being released on a daily basis. The amount paid by victims of these attacks increased by more than 350% in 2021. Additionally, it is estimated that by 2031, organizations will lose around 265 billion USD to ransomware attacks, with a new victim every 2 seconds (Oz et al., 2021). This surge is concerning since the rate of development of these attacks is significantly faster than the development of preventive and detection methods against crypto ransomware attacks. The attackers are always in the lead, but the defenders are always catching up. Without effective efforts to close the gap between attackers and defenders, the gap will continue to increase. In this regard, the researcher is highly motivated to learn more about crypto-ransomware and develop a solution to stop these attacks or, at the very least, mitigate the damage caused by crypto-ransomware.

1.8 Scope of the Study

- Due to the limited time available for conducting this research, it will be impossible to investigate all current methods for detecting and preventing crypto-ransomware.
- Some malware is capable of detecting a virtual environment and, as a result, fails to execute in the same manner as it would in a real environment.
- This thesis is not concerned with mobile operating systems (OS), but rather with desktop operating systems, specifically malware affecting Microsoft Windows, the world's most popular desktop operating system (OS).

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

This chapter will cover the background information on ransomware and related security challenges before diving into cryptographic ransomware, the threat type studied in this work. The stages of ransomware infection will be presented, as well as ransomware behaviours and characteristics, for a better understanding of the threat. Based on previous research works, the section will also cover the various approaches for detecting crypto-ransomware attacks.

The term "ransomware" is derived from two words "ransom" and "ware." As the name implies, this malware family demands a ransom from victims and threatens to delete data if the ransom is not paid. (Gazet, 2010).

2.2 Types of Ransomware

Ransomware can roughly be divided into two types based on its behaviour and functionality (Gazet, 2010). The non-encryption ransomware and the encryption ransomware. The non-encryption ransomware, often known as locker ransomware, are ransomware variants that use various lockout mechanisms to prevent access to infected systems and demand ransom from the victims. Normally, the victim has limited access to the system, requiring them to interact solely with the ransomware criminal. For example, sections of the keyboard could be disabled, or the mouse could be frozen. This is to allow interaction with the ransom demand window so that the payment can be made. Aside from that, the machine is inoperable. However, the locker malware normally doesn't go for files; instead, it only wants to lock you out. As a result, complete data destruction is unlikely (Gazet, 2010). Because non-encryption ransomware doesn't employ any encryption algorithms, systems infected with it are easy to recover.

The encryption ransomware, commonly known as crypto-ransomware, is a kind of malicious software that encrypts files using strong cryptographic methods in an attempt to extort money from users. Crypto ransomware takes advantage of system flaws and vulnerabilities to encrypt valuable data before disclosing itself to the victim. This data can include data files, image files, video files, and MS Office files, among other things.

Based on the characteristics, unique behaviour, and origin, each ransomware type is subdivided into numerous different ransomware families. This research focuses on the encrypting ransomware.

2.3 Ransomware infection vector

According to Thakkar (2014), “ransomware authors use common malware distribution methods such as phishing emails, drive-by downloads from a compromised website, malvertising and exploitkits.” Intrusion detection systems, anti-spam filters and other security techniques should typically detect these threats. Unfortunately, ransomware perpetrators are increasingly relying on social engineering to target their victims.

2.3.1 Phishing Electronic mail (e-mail)

According to Branche (2017), electronic mail (e-mail) is the most common source of ransomware infection. It is also the most successful and simple method of infection. Phishing e-mails are sent to a list of e-mail addresses obtained through various techniques by cybercriminals. This e-mail usually includes an attachment or a legitimate-looking website link, but it contains ransomware. When the victim clicks on the attachment, the ransomware is activated and begins to spread throughout the victim machine. Advanced ransomware will scan the victim's computer for vulnerabilities before contacting the developer or the attack Command and Control (C&C) Centre. After establishing the communication with C&C, It sends the acquired victim's information and receives the encryption key to encrypt the victim’s files. (Humayun, Jhanjhi, Alsayat, & Ponnusamy, 2021)

In organizations, once ransomware has infiltrated one machine, more advanced ransomware variants will propagate to other machines on the network (PCs and servers) (Kok & Jhanjhi, 2020). An entire organization can be compromised by one person opening an attachment in a phishing email.

This strategy by ransomware authors has been effective for several reasons. To begin with, some attachments elude anti-spam filters; additionally, a carefully crafted spam message deceives and persuades a victim to open an attachment or click on a link that redirects you to a malicious file. And the majority of users are unaware of the security risks associated with unsolicited attachments. Finally, using a well-known cloud service such as Google Drive to distribute malware does not trigger alarm among many people (Humayun et al., 2021)

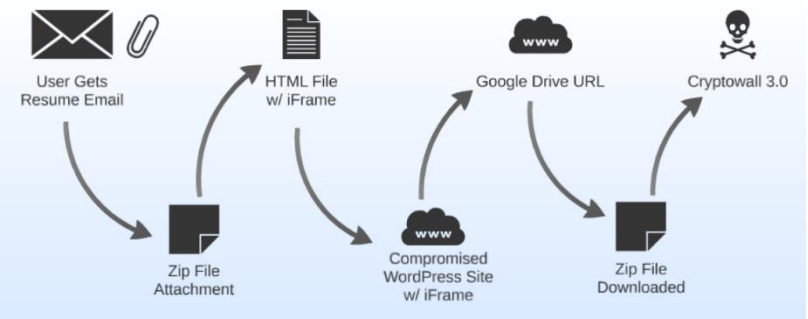


Figure 2.1 : Infection chain for CryptoWall through use of spam email (Biasini, 2015)

2.3.2 Drive-by downloads & Malvertising

"A drive-by download involves compromising a legitimate website and then configuring that web-page to redirect visitors to site containing exploit kits." (Kotov & Massacci, 2013)

Cyber criminals take advantage of well-known weaknesses in legitimate websites' software. They then exploit these flaws to redirect the victim to a site under their control that holds the exploit kits. Kotov & Massacci, (2013) defines exploit kits as a "web-based tool that exploits various security vulnerabilities on the browser plugins and applications running on the victim's system." Typically, the cybercriminals acquire exploit-kits that incorporate the malware they plan to distribute to victims. Exploit kits allow hackers to silently scan a visitor's device for weaknesses and, if any are identified, execute some background code. The user will then be confronted with a ransom note informing them of the infection and demanding ransom.

Malvertising, or malicious advertisement, is similar to a drive-by attack in that the victim visits a legitimate website, however, the attacker who has embedded malicious code into the advertising space may mislead the website.

Because existing solutions mainly rely on URL categorization and domain-based blocking, exploit kits used in both drive-by downloads and malvertising can easily evade the network-based detection systems. (Kotov & Massacci, 2013)

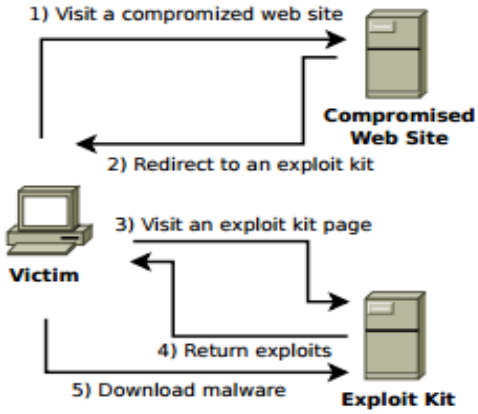


Figure 2.2 : Drive-by download attack scheme (Kotov & Massacci, 2018)

2.3.3 Remote Desktop protocol

Remote Desktop Protocol (RDP) was intended to allow IT administrators to safely access a user's system remotely, either to perform configurations or to simply use the machine. This approach has various advantages, but it can also be used illegally by a criminal. (Garg, Thakral, Nalwa, & Choudhury, 2018). RDP often runs on port 3389. An estimated ten million workstations advertise themselves as running RDP over 3389 on the internet. Cybercriminals can easily find and identify susceptible devices by searching the internet. Once the target system has been discovered, the hacker can employ brute-force to get administrator access to the machine. Once the hacker has administrator privileges, the ransomware encryption operation can be easily launched. (Garg et al., 2018). There are numerous open-source password-cracking tools to help accomplish this objective

2.3.4 USB and Removable Media

Ransomware has also been known to penetrate an environment via USB and removable media. The USB devices may disguise as a promotional application, but once opened, they would deploy ransomware on the PC of an unsuspecting user. (Pham, Halgamuge, Syed & Mendis, 2010)

Ransomware has become the preferred method of revenue generation for cybercriminals. Ransomware-as-a-Service (RaaS) is easy to buy on the dark web, and attacks are reasonably easy to launch using one of the methods listed above. Individuals and businesses must be aware of how their systems can be compromised and take proactive measures to protect themselves and their data.

2.4 Common Characteristics and Behaviours of Crypto-ransomware

Encrypting files is typically the ultimate goal of ransomware on a target device. As a result, the majority of ransomware exhibits some common behaviours that may indicate the presence of a malicious program (Gazet, 2010). The characteristics of crypto-ransomware resemble those of other malware in many ways. Most notably the installation tactic prior to delivering the payload.

Crypto-ransomware behaviours are broadly classified into two categories: pre-encryption behaviours and file encryption process behaviours (Kok et al., 2019).

2.4.1 Pre-encryption Behaviours

Among the pre-encryption behaviours are:

Payload persistence: This action ensures that the attack remains persistent even after rebooting the system. The most commonly used techniques are inserting an executable file in the start-up directory, creating a scheduled task and addition of new registry key (Kok et al., 2019).

Restrict system restore: This is designed to stop the victim from reverting the system to its pre-infection state. Frequently used techniques include deleting scheduled backups, backup systems, and backup files (Kok et al., 2019).

Stealth/evasion techniques: This is done to keep the victim unaware of the attack. The most common tactic is to use a name that is identical to the standard system executable (Kok et al., 2019).

Environment mapping: This procedure ensures that the infection is in the victim's system rather than in a sandbox environment where it is being analysed. Among the most commonly used techniques are verifying the security settings and policies, geographical region, and file system design (Kotov & Massacci, 2013).

Communication Masking: This is done to guarantee that the connection to the Command and Control Center is successful for either downloading payload-related data or communicating the encryption key (Kotov, & Rajpal, 2014).

Privilege elevation: The user account access rights granted to the victim may not be adequate to carry out malicious system-related operations. Many system-related activities are exclusively available to administrators; hence, elevation to the administrator level assures that all actions may be performed without restrictions (Garg et al., 2018).

2.4.2 File Encryption Process Behaviours

The ability to convert data from a usable to an unusable state is at the heart of crypto-ransomware behaviour. The ransomware activity may exhibit certain behaviours during the encryption process. These behaviours include:

Increased file system activity: Ransomware increases file system activity by scanning the file system, encrypting all or a portion of data, and deleting or overwriting a huge number of existing files including shadow copies (Gazet, 2010).

High Input/output accesses: The operations performed by ransomware (i.e., encryption, deletion or overwrite) involve repetitive I/O access activities such as read operations to read user files without the user's authorization. It performs write operations either to produce encrypted copies or to overwrite the originals. In the case of the first option, ransomware conducts further processes to delete the original data (Beaman et al., 2021).

High resource usage: Because ransomware relies on encryption operation, significant CPU or memory utilization might indicate the presence of ransomware in the system (Kok et al., 2019).

Changes in registry activity: ransomware performs changes in the registry to remain persistent after system reboots (Kok et al., 2019).

2.5 Ransomware Lifecycle

According to Kok, Jhanjhi & Supramaniam, (2019), "the lifetime of ransomware is divided into seven stages, each of which requires close collaboration between the "creator" and the "campaigner." The creator is the programmer who creates the malware, whereas the campaigner is mainly the malware distributor." This form of partnership allows both parties to improve and refine their skill and knowledge in their focus area. As a result, cyber criminals with advanced competencies emerge.

The creation of ransomware is the initial stage. The creator's main goal is to develop ransomware using programming codes.

The campaigner's major goal in the second stage is to distribute ransomware to the victim's machine. Individuals and institutions are the two main types of target victims. The purpose of disseminating to individual victims is usually to reach as many individuals as possible. However, for institutional victims, the ransomware attack is often targeted. This is because, in most cases, there is already some type of security defence in place. (Kok et al., 2019). Compromised websites and Email attachments links are the common infection vectors.

The third stage is infection. The ransomware set-up behaviour begins as soon as the payload is delivered to the victim's system. More advanced ransomwares, on the other hand, could take additional precautions to avoid detection (Kok et al., 2019).

Communication with the Command and Control (C&C) servers is the fourth stage. The cyber-criminals create ransomware Command & Control servers for generating encryption keys, tracking installation and pay-outs, and potentially download more files for more advanced infections. Once installed, the ransomware program sends a POST request containing system information to an embedded domain or a dynamically generated domain using a Domain Name Generation Algorithm. (DNGA). The C&C server generates a key pair and sends the public key to the ransomware in order for it to encrypt files. (Kotov, & Rajpal, 2014). After gaining the encryption key, the ransomware might begin searching for seemingly valuable assets such as documents, spreadsheets, presentations, photos, and databases in stage five. (Kok et al., 2019).

Encryption is the next step. Once the valuable files have been identified, the ransomware starts the encryption operation. The encryption techniques employed by ransomware are covered in the next section.

The last stage is extortion. Displaying the ransom demand is the cycle's last stage. The victim is informed of the infection, and the payment mechanism might well be specified on the extortion letter. The letter may also mention the ransom payment deadline, beyond which the ransomware would start erasing the valuable files (Gazet, 2010).

Among the steps mentioned above, the pre-encryption stage in the crypto-ransomware attacks lifecycle includes (2) infection - installation, (3) encryption key creation and (4) file search (Al-Rimy et al., 2021).

Figure 2.3 : Crypto-ransomware Lifecycle (Kok et al., 2019).

2.6 Ransomware Encryption Techniques

"The most important functionality of encrypting ransomware is ability to use robust cryptographic libraries to encrypt files." (Kotov & Rajpal, 2014).

Crypto-ransomware employs various types of encryption algorithms. Symmetrical encryption employs the same key for both encryption and decryption processes. Although this method of encryption performs better, it does not provide sufficient encryption strength.

Asymmetric encryption employs the use of two keys. The encrypting operation uses one key, known as the public key, while the decrypting operation uses a different key, known as the private key. This method of encryption is strong, but is resource intensive hence slow to complete, it also has a high chance of being detected due to high Central Processing Unit (CPU) usage for a long time (Kotov & Rajpal, 2014).

In order to achieve good performance and strong encryption, crypto-ransomware commonly uses a hybrid algorithm that incorporates both symmetric and asymmetric approaches to encrypt the victim's information. This usually entails symmetrical encryption of data or files, followed by asymmetrical encryption of the secret key (Kotov & Rajpal, 2014). However, the crypto-ransomware can utilize any encryption method.

2.6.1 Acquiring Encryption Keys

Crypto-ransomware uses different approaches to obtain the encryption keys. (Fischer, 2014). The first strategy entails obtaining public keys from Command-and-Control (C&C) servers. The keys can also be derived using Cryptographically Secure Pseudo-Random Number Generator (CSPRNG) function outputs, e.g., in the WannaCry ransomware (Genç, Lenzini & Ryan 2018). Existing functions in programming languages, such as the non-cryptographic Pseudo-Random Number Generator (PRNG) in the C Library, can also be used to generate pseudo-random numbers (keys). Finally, before distributing ransomware, the encryption keys may be inserted into the malware e.g., Cryzip ransomware (Genç, Lenzini & Ryan 2018).

Figure 2.2 shows the CryptoLocker malware's step-by-step process for encrypting files on a victim's computer. These procedures are used by the majority of crypto-ransomware variations (Fischer, 2014). The public key encryption technique or RSA, is an asymmetric key cryptography method that employs both a public and a private key. The public key encrypts files, while the private key decrypts them. AES, on the other hand, is symmetric, which means

it utilizes the same key for both encryption and decryption. The ransomware encrypts the targeted files with the AES key, the key is then encrypted with the RSA public key, which is either included into the ransomware or obtained from the command and control server. Once files have been encrypted, only the private key held by the ransomware creators can decrypt them. The encryption strength has improved over the years from RSA+AES to ECDH+AES.

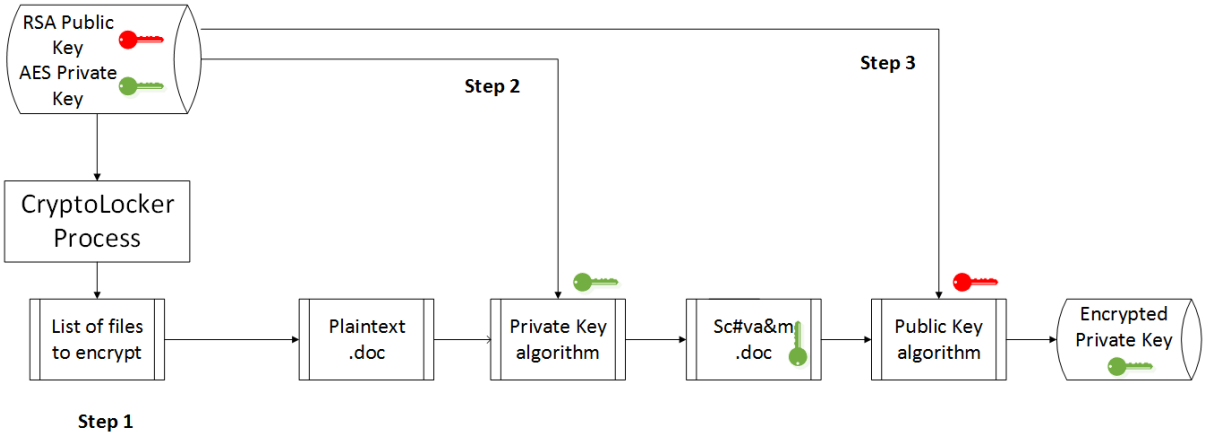


Figure 2.4 : CryptoLocker encryption steps (Fischer, 2014)

In Kotov & Rajpal (2014) "CryptoLocker and CryptoWall depend on Microsoft CryptoAPI to create keys and to encrypt data with the RSA and AES algorithms." In such cases, embedding the keys aided early detection and allowed for session key interception. As a result recent malware variants avoid this kind behaviour detection. Furthermore, these variants inscribe encrypted content into the original file and remove the volume shadow copies to limit the possibility of recovering the original file (Kotov & Rajpal, 2014).

2.7 Ransomware Analysis Methods

The objective of ransomware analysis is to gain a better understanding of the behaviour and functionality of various ransomware families. Based on this understanding, defensive steps can be formulated to prevent future attacks (Kok et al., 2019). In general, there are two types of malware analysis: static and dynamic.

2.7.1 Static Analysis

Static analysis is a method for analysing malicious files without executing them. It is carried out by quickly assessing the features of an executable piece of code and comparing them to previously discovered malicious code (Kok et al., 2019).

Static analysis uses different approaches to achieve the objective. One, the operation code, commonly known as opcode is extracted from the source code to generate the n-gram

codes sequence. A machine-learning technique can be used to train the pattern of n-gram codes to create a ransomware detection predictive model (Zhang et al., 2019). The second method tries to find the ransom note embedded in the source code or a code that attempts to import the ransom note after the encryption process is complete (Alzahrani, et al., 2018). The third approach extracts the portable executable header information (Zhang et al., 2019). The fourth method aims to generate rules based on the source-code (Cimitile et al., 2018).

This approach has the advantage of being able to quickly and simply analyse malicious code. The method is also considered safer because the ransomware can be detected before it can be executed. (Gandotra, Bansal & Sofat, 2014).

Static analysis, on the other hand, has a number of drawbacks. For starters, it's vulnerable to code obfuscation, simple operational code insertion can result in a mismatch with malicious code that had previously been discovered. Additionally, when the code is encrypted, the analysis may not be effective since using brute force to decrypt an encryption is currently ineffective and time consuming (Kok et al., 2019). Static analysis is ineffective against multi-phase attacks as well. The initial code could simply be a mechanism to open a backdoor for additional scripts to be downloaded, and hence may not have a similar malicious action (Kok et al., 2019).

2.7.2 Dynamic Analysis

Dynamic analysis or behavioural-based analysis is the process of activating the ransomware in a controlled environment known as virtual machine or sandbox. This allows for direct monitoring and interaction with the malware in order to analyse its behaviour (Gandotra et al., 2014). Dynamic analysis is considered to be more powerful and reliable in detecting malware's behaviour and functionality; it is also less susceptible to code obfuscation and can analyse encrypted code (Kok et al., 2019).

However, this method has drawbacks, such as the risk of ransomware infecting the host machine and the malware becoming inactive after detecting the virtual environment. In some cases, malware programs may detect the analysis environment through the use of anti-virtual machine and anti-debugger techniques and either fail to display the original behaviour or terminate themselves (Gandotra et al., 2014).

There are several approaches to dynamic analysis, one of which is to track I/O operations to monitor file-system activities while taking into account the repetitive activities in the file system caused by encryption (Kardile, 2017). The second method monitors the frequency of encryption in order to determine the legitimate use and likelihood of a

ransomware attack. (Gandotra et al., 2014). The third method monitors the System API to track ransomware interactions with the operating system (Scalas et al., 2019). The fourth uses the dynamic runtime opcode traces for the specific processes performed (Carlin, Kane & Sezer 2019).

Parameters captured during dynamic analysis

A wide range of parameters can be captured during dynamic analysis. They include Application programming interfaces, File Input/output operations, Opcode and Bytecode sequences, Network traffic, and Registry (REG) key among others. These parameters are discussed briefly below.

Application programming interfaces (APIs)

API calls are the mechanism via which programs communicate. They allow information exchange between applications. Ransomware, for example, may exploit APIs to access or alter files, elevate privileges, acquire decryption or encryption key(s) from the command and control server, and so on. (Beaman et al., 2021). API calls and the frequency of API instances can provide further insight into the process's behaviour. In addition, each API call's input arguments provide information concerning the activity, such as the Registry-key accessed, the mutex generated or updated, the DLL file loaded, and so on. (Beaman et al., 2021).

File Input/output operations

File operation metrics include the amount of files read or written to, the average file-write operation entropy, the frequency of file operations for each type of file, and the amount of files accessed (Beaman et al., 2021).

Opcode and Bytecode sequences

The "operation-codes," commonly known as Opcodes, are the fundamental processor instructions that a system will execute. Bytecode, on the other hand, is a kind of instruction intended for execution by an application interpreter for example Java Virtual Machine (Carlin et al., 2019).

Network traffic

The packet size, the quantity of packets sent and received by the host and other devices, and the source and destination IP addresses included in the packet headers are some of the features of network traffic.

Some researchers propose a hybrid strategy, a blend of both dynamic analysis and static methods (Gandotra et al., 2014).

2.8 Ransomware Detection Methods

Aslan & Samet (2020) defines malware detection as "the process of examining the content of a program to determine whether it is malicious or benign." In general, detection techniques are classified into three types: behaviour-based (those that use dynamic analysis), signature-based (those that use static analysis), and hybrid (those that use both dynamic and static analysis).

2.8.1 Signature-based detection

Signature-based detection method examines an application's code before it is executed to determine whether it is capable of malicious activity. If the static analysis detects any malicious code, the executable will be prevented from running. Signature analysis involves extracting code string patterns (signatures) from the code of the target application and comparing them to a database of known malicious-code patterns (Nieuwenhuizen 2017).

This method is quick and effective for detecting known malware. The inability of signature-based detection to detect unknown malware is its major disadvantage. A malicious executable can only be detected after it has been reported as malicious and added to the malicious signature database (Nieuwenhuizen 2017). This has several consequences for the efficacy of static-based detection: The first is that it is ineffectual against code-obfuscation – ransomware developers use code-obfuscation strategies to continuously change malware so that each variant appears different to avoid signature-based detection.

Two, signature-based detection is ineffective against malware with short development cycles. This is an issue for signature-based detection systems since new ransomware variants are produced quicker than new signatures can be created, tested, and added to malicious signature repositories (Nieuwenhuizen 2017).

Three, in the face of targeted ransomware, signature-based detection is ineffective. Using the Ransomware-as-a-Service paradigm, bots can modify signatures to target specific organizations. This enables the development of a highly customizable ransomware variant capable of evading signature-based detection (Kok et al., 2019).

Related works for signature-based detection

Several researchers have proposed static analysis-based methods for detecting crypto-ransomware attacks.

Zhang et al. (2019) proposed a model for converting opcode (operation code) sequences into N-gram sequences, which were then used for training the machine learning model, the model had an accuracy of 91.43%.

In Baldwin & Dehghantanha (2018), Static analysis was used to extract opcodes from malicious and benign Portable Executable files. The extracted opcode characteristics served as input to SVM machine-learning classifier. The best accuracy obtained from five crypto ransomware families was about 96.5%.

Alzahrani et al. (2018) proposed a static analysis approach referred to as RanDroid. This strategy looks for a potentially dangerous message in the application code, which can be text or an image. The reason for this is that ransomware's goal is to extort money from its victim. As a result, it needs to include a threatening-message in the code. The main disadvantage of this method is that the threatening-message may be delivered as a payload after the victim's data has been encrypted.

2.8.2 Behaviour-based detection

To address the shortcomings of signature-based detection methods, detection methods capable of dealing with crypto-ransomware focus on broader features such as ransomware-specific operations rather than just file signatures alone.

Behaviour-based detection entails monitoring a process's behaviours and interactions with its surroundings in real time to detect malicious intent. All executables are considered unknown, and it is the executable's responsibility to demonstrate that it is safe and non-malicious. Any maliciously behaving process will be identified and terminated (Nieuwenhuizen 2017).

Behaviour-based detection methods can be broadly classified as data-centric or process-centric (Al-Rimy et al., 2021). The data-centric solutions monitor the victim's computer's digital assets and sound an alarm if any suspicious changes are found (Al-Rimy et al., 2021). They examine the file structure changes to see whether they are suspicious. They employ techniques such as contents similarity measures, file entropy, and decoy strategies to keep track of the structure of the file both prior to and after access (Al-Rimy et al., 2021, Kharaz et al., 2016)

The process-centric solutions are classified into two groups. One, by monitoring system resources such as file system access for example privilege elevation, network activity, resource usage, and interactions with the operating system and raising an alarm when certain encryption-related events are encountered. Among these, the crypto-ransomware system activity is distinguished by its aggressive encryption of the victim's data, resulting in unusual file system activity (Kharaz et al., 2016; Sgandurra, 2019). Two, by monitoring the running process's behaviour, evaluating each line of code and all possible actions performed by that code are

examined, such as access to any critical or irrelevant files, processes, or internal services and gathering various forms of behavioural data which are subsequently utilized to train different machine learning classifiers (Al-Rimy et al., 2021).

One of the major drawbacks of behaviour-based detection is that it is significantly more difficult to implement, and two, dynamic analysis across multiple dimensions introduces delay, which negatively affects performance. Three, advanced code obfuscation techniques make proper malware examination impossible. Finally, some malware does not behave properly in virtual machines or sandboxes (Oz et al., 2021).

Related works for Behaviour-based detection

Kharaz et al. (2016) proposes Unveil, a dynamic analysis system that runs apps in a virtual environment and monitors file system operations and desktop interactions for anomalies that could indicate ransomware infection. Unveil had a detection rate of 96.3%. Unveil, on the other hand, may mistake heavy file system activity for ransomware.

Hwang et al. (2020) proposed a two-phase mixed detection model for ransomware, a Random Forest model and a Markov model. They tested API classification and reported that it was 97.3% accurate with a 4.8% FPR and a 1.5% FNR

Takeuchi, Sakai & Fukumoto (2018) present a ransomware detection strategy based on support vector machines (SVMs) that learn the ransomware API calls as features, allowing the SVM to detect previously unknown ransomware. They used 276 ransomware samples and 312 goodware samples and reported a detection rate of 97.48%.

In 2016, Weckstén, Frick, Sjöström & Järpe (2016), studied the behaviour of four different types of crypto ransomware in a virtual machine running the Windows 7 operating system. They observed that vssadmin.exe is heavily used in crypto ransomware attacks and recommended that users should avoid using the vssadmin.exe to avoid this attack.

In Continella et al. (2016) the authors try to locate AES encryption key in process memory by intercepting file access system calls. In the event that the original file is encrypted, the method allows for its restoration. However, this method only works with ransomware that uses AES symmetric encryption; if the ransomware utilizes another encryption method, this method will fail. The method can also be uninstalled by ransomware with administrator privileges.

In other proposals, (Shukla et al., 2016; Kharraz, 2017; Scaife et al., 2016;) file-type changes such as the number of read, written, modified, and/or deleted files can be used to detect ransomware activity. Access to a wide variety of file types, a search through a large number of

directories, and the creation of files with significantly higher entropy than the original file all indicate the possibility of data encryption (Morato et al., 2018). These papers use machine learning techniques or a linear combination of measured indicators to analyse system call patterns. These algorithms may produce good detection results, but they may have a negative impact on host performance due to high CPU usage; they also produce a high number of false positive detections when the user performs actions similar to ransomware; and they are susceptible to malware that elevates privileges and disables detection tools. (Morato et al., 2018).

REDFISH (Ransomware Early Detection from File Sharing Traffic) proposed by Morato et al., (2018), rely on a set time-based threshold approach, implying that all samples begin encryption at the same moment, this is not always true for all samples because the timing varies between crypto-ransomware families as a result of the obfuscation techniques used (Al-Rimy et al., 2020).

Lee et al. (2017) and Mehnaz et al. (2018) proposed the use of decoy files. In addition, CryptoStopper, a commercial anti-ransomware program, detects ransomware by using decoy files (Genç, 2020). This strategy is also known as deception-based ransomware detection, and it involves placing carefully crafted files in the file system alongside the user's files as a decoy. (Genç, 2020). Because the user is not supposed to change or delete the decoy files, any write/delete request to them is interpreted as an indication of ransomware activity. Monitoring decoy file access uses up fewer system resources than behavioural analysis/detection; however, because there is no assurance that the ransomware will start by attacking the decoy files, a significant number of files must be monitored in order to detect malware prior to the loss of a huge number of user files. This method is also susceptible to ransomware, which can detect and avoid decoys when generating a list of files to encrypt (Genç, 2019).

2.8.3 Hybrid detection

The hybrid system detects crypto-ransomware using a combination of behaviour-based and signature-based detection techniques.

Subedi, Budhathoki, & Dasgupta (2018) proposed an integrated approach that combines static analysis (using a data-mining technique) and run-time analysis to correlate code segments with the dynamic behaviour of ransomware They created CRSTATIC, a reverse engineering-based analytic tool for detecting crypto ransomware.

Shaukat & Ribeiro (2018) introduced a Strong Trap Layer that combines dynamic and static analysis to generate a novel compact set of attributes that employs machine learning

approaches to define Ransomware behaviour. The detection-rate was 98% while using the Gradient-Tree Boosting Algorithm.

Other proposals to combat the crypto-ransomware threat include the use of key escrow. Key escrow is a method of storing keys and cryptographic materials for later use. In ransomware perspective, key escrow seeks to collect the secret keys and other variables whereas the ransomware encrypts files so that they can be recovered after the threat. This strategy is premised on the idea that if the encryption keys are retrieved after the attack, ransomware-encrypted files can be recovered. Kolodenker et al., (2017) proposed PayBreak, a cryptographic API monitoring technique that extracts cryptographic materials and encryption keys and stores them in a key vault, so that in the event of a ransomware attack, encrypted files can be decrypted using brute-force methods based on the keys and other variables obtained from the key vault. However, according to the authors, ransomware can evade this method by utilizing third-party libraries and code obfuscation. (Kolodenker et al., 2017).

2.9 Machine Learned Detection Methods

Ransomware detection as a machine learning problem

The amount of malware samples produced every day poses a difficult task for analysts. The evolving challenge of malware detection as well as the increasing popularity of machine learning, may explain why more malware detection articles use machine learning (Rhode, 2021).

In recent years, there has been a rise in interest in using machine learning to detect malware automatically based on its dynamic behaviours. Because of their self-learning capabilities, ML approaches have been deemed effective, and they have significantly improved detection rates on a wide scale when compared to conventional malware analysis method. (Chen, Yang, Paul & Sahita, 2018).

"Machine learning," according to artificial intelligence pioneer Arthur Samuel, is a “set of methods that gives computers the ability to learn without being explicitly programmed.” (Naqa & Murphy, 2015). A machine learning system finds and recognizes the principles that underpin the data it encounters. The algorithm can utilize this information to 'reason' the features of samples it has never seen before, who’s hidden behaviour could be malicious or benign. The concept behind machine learning is to use an algorithm to train a model to do a specific task, such as classification, regression, clustering and so on.

Machine learning models accept data for input then return some value. Models go through a learning (training) phase, which is typically followed by a testing (inference) phase

(Rhode, 2021). In the learning phase, the algorithm's parameters are modified according to certain optimization expressions, such as the agreement between the algorithm's malware or benignware labels and the ground truth indicating whether the program is malicious or benign phase (Rhode, 2021).

Supervised and unsupervised Learning

There are two approaches to machine learning: supervised learning and unsupervised learning (Chen et al., 2018). Labelled data or the ground-truth serves as the foundation for learning in Supervised Learning. “Supervised learning algorithms iteratively adjust the parameters of the model to minimise the disparity between the ground truth labels and the output of the model.” (Rhode, 2021). The aim of supervised learning is to build a model from a labelled training dataset that can correctly predict previously unknown features (Taylor, 2017). .

In the context of ransomware detection, “supervised-machine learning refers to the process of training a decision algorithm to recognize specific behavioural traits (the input data) of running processes in order to differentiate between crypto ransomware and non-malicious apps (the target label).” Nieuwenhuizen (2017).

There are two types of supervised learning techniques. Regression and Classification (Chen et al., 2018). Regression method makes predictions based on the interaction between independent variables and dependent variables. The method predicts a single output-value (Chumachenko, 2017). The classification method determines which class an observation belongs to based on past seen labelled training data (Taylor, 2017). The method predicts a categorical variable (Chumachenko, 2017)

Unsupervised learning is a type of machine learning that infers patterns from unlabelled input data. Unsupervised learning seeks to extract structures from input data; it does not require supervision; it discovers patterns in the data on its own (Taylor, 2017). However, to assess if a sample is malicious, the model's author must select a baseline and a threshold of deviation from that baseline (Rhode, 2021). Unsupervised learning models are mainly used for three tasks: association, clustering and dimensionality reduction. The association method employs a variety of rules to discover relations between variables in a provided set of data (Taylor, 2017). The clustering technique categorizes and groups unlabelled data based on similarities and differences discovered. When the number of features in a given dataset is too large, the dimensionality reduction technique is used. It minimizes the number of data inputs while maintaining data integrity (Chumachenko, 2017).

2.9.1 Machine Learning Classification Algorithms

There are various ML algorithms, and a number of them have been used in malware detection. Common factors that influence algorithm selection include the error rate and computational burden during training and deduction (Rhode, 2021). Despite the variety of ML approaches available, some are more widely used than others. Several algorithms are detailed further below.

Decision Tree

The Decision Tree (DT) is a supervised learning method which can be used for regression as well as classification tasks. It is most commonly used, however, to solve classification problems. DT is a classifier with a tree structure, where internal nodes depict dataset characteristics, the branches depict the decision rules, and each leaf node depicts the result (Chumachenko, 2017). Decisions are based on the characteristics of a specific set of data.

K-Nearest Neighbor

The k-nearest neighbour (K-NN) algorithm is a non-parametric pattern recognition method, which means it makes no inferences about the data it is working with (Taylor, 2017). The algorithm makes use of an n-dimensional feature space in which n represents the number of features (independent variables) (Taylor, 2017). The training data-set makes a prediction relying on the k closest training data-points to the input vector of feature values. The class membership of the k closest data points is polled, and the class with the most neighbours becomes the prediction (Chumachenko, 2017). Although the K-NN algorithm can be used for both classification and regression, it is most commonly used for classification tasks.

Support Vector Machine

Support vector machines (SVMs) is a supervised learning algorithm that could be used to solve regression and classification problems. However, it is mostly utilized to solve classification problems. When given a labelled training set, SVMs, try to locate a hyperplane in the n-dimensional feature space that maximizes the difference between two classes (Taylor, 2017). The SVM algorithm's goal is to arrive at a decision borderline for categorizing n-dimensional space into classes such that new data-points can be quickly positioned in the appropriate group in the future (Taylor, 2017). However, SVMs struggle with big datasets because of their training difficulty, because the separating hyperplane need to account for a large number of data-points (Rhode, 2021).

Naive Bayes

The Naive Bayes is a supervised learning algorithm that makes use of the Bayes Theorem and posterior probability (Taylor, 2017). The main idea is that each feature should be addressed separately. Naive Bayes assesses the likelihood of every feature individually, irrespective of any similarities, and bases its prediction on the Bayes Theorem. As a result, this method is referred to as "naive." since features in real-world situations frequently have some degree of similarity (Taylor, 2017).

Random Forest

Random Forest (RF) is a well-known supervised learning algorithm. RF is built on the principle of ensemble learning, this means that rather than relying solely on one decision tree, the random forest uses each tree's predictions to predict the final result based on the prediction with the most votes. According to Fawagreh et al. (2014), "Random Forest is a classifier that consists of a number of decision trees on various subsets of a given dataset and takes the average to enhance the prediction accuracy of that dataset." By merging various classifiers, RF overcomes decision tree classifiers' low performance in modelling difficult functions and improves the model's performance (Fawagreh et al. 2014). The more trees there are in the forest, the more accurate and the lower the risk of overfitting. RF may be used for both regression and classification tasks. However, the number of trees used in the Random Forest has an effect on the size of the classifier as well as the time it takes for classification of data. (Khammas, 2020)

Neural network

According to Rosenblatt, (1958), “Neural networks are brain-inspired general function approximators which map input data to outputs through a network of neurons connected by weighted paths.” A feed-forward neural network, better referred to as a Multi-Layer Perceptron (MLP), is the most basic type of neural network (NN) and was initially presented by Rosenblatt, (1958) as a simplified depiction of how neurons in the brain process information. Hardware acceleration has allowed for the rapid training of multi-layer and extremely large perceptrons i.e the deep neural networks (Rhode, 2021). The hyper-parameters used to construct a neural network can have a significant impact on its success. NNs require more resources to train than several different ML models (Rhode, 2021).

Although there are trends, there is no ideal malware detection algorithm; the best-performing method is decided by the input data, which includes both the features used and the dataset samples (Rhode, 2021). This makes it challenging for researchers to decide which

algorithm to use. Based on research presented by Rhode et al., (2019), Random Forests, Support Vector Machines and NNs are the most common methods used to detect malware.

2.10 Comparison of Selected Early Detection Methods

In this section, we provide a few works that are closely related to this study.

Solution	Contribution	Result	Limitation	Run-time limit
Scaife et al. (2016) CryptoDrop	-An early warning detection system that examines file activity depending on encryption behaviour.	Accuracy 93% Error rate 6%	- Static Thresholding -Needs user interaction. - Data loss as a result of encryption prior to behavior analysis - Ten files are lost on average. -40 files encrypted before detection	30 seconds
Sgandurra et al. (2016) EldeRan,	- Monitors a series of tasks carried out by programs during their initial phases of installation, looking for indications of ransomware	Accuracy 97.89% AUC 99.49%, Test Error 2.38% FPR 1.61% Detection Rate 96.34%	- Static Thresholding -Cannot detect ransomware that awaits user action or launches after 10 seconds - Subject to code obfuscation	10 seconds
Homayoun et al. (2017)	-Application of pattern mining techniques to monitor ransomware activities	Accuracy 98.8% AUC 99.5%, F-Score 98%, Recall 98%, FPR 0.97%	- Static Thresholding Cannot detect ransomware that awaits user action or launches after 10 seconds - Subject to code obfuscation	10 seconds
Rhode et al. (2021)	-5 second snapshot of behavioural data utilizing recurrent neural network	Accuracy 96.01%, FPR 3.17%, FNR 4.72%	- Static Thresholding -Subject to code obfuscation - Source code being examined	5 Seconds

			might be incomplete at the beginning.	
Stiborek et al. (2018)	-Generative probabilistic clustering using the Bernoulli mixture model	Accuracy 97.3% FPR 4.8% FNR 1.5%	- Fails on instances that don't interact with sandbox-monitored resources. -High likelihood of benign applications being predicted incorrectly	No time limit
Shaukat & Ribeiro (2018) RansomWall Trap layer	- A tiered defensive mechanism for cryptographic ransomware protection.	Accuracy TPR 98.25% FPR 0.56%	- User-critical files could be attacked & encrypted before honey files. - Some ransomware limits file system operations	No time limit
Lin et al. (2018)	-Time-based ransomware detection is fast.	Precision 98.6%	- Might not work for non-time-based ransomware	120 seconds
Alzahrani et al. (2018) RanDroid	- Retrieves texts and images to determine whether or not a threatening notice is available.	91% accuracy	- Dependent on the availability of a ransom note.	No time limit

Table 2.1 : Comparison of selected detection methods

2.11 Conceptual Framework

The perceived link between independent variables and the dependent variable is depicted in the conceptual framework shown below. It is believed that the independent variables (API calls and functions) are connected to the detection of crypto-ransomware attacks; a premise that partly guides this study.

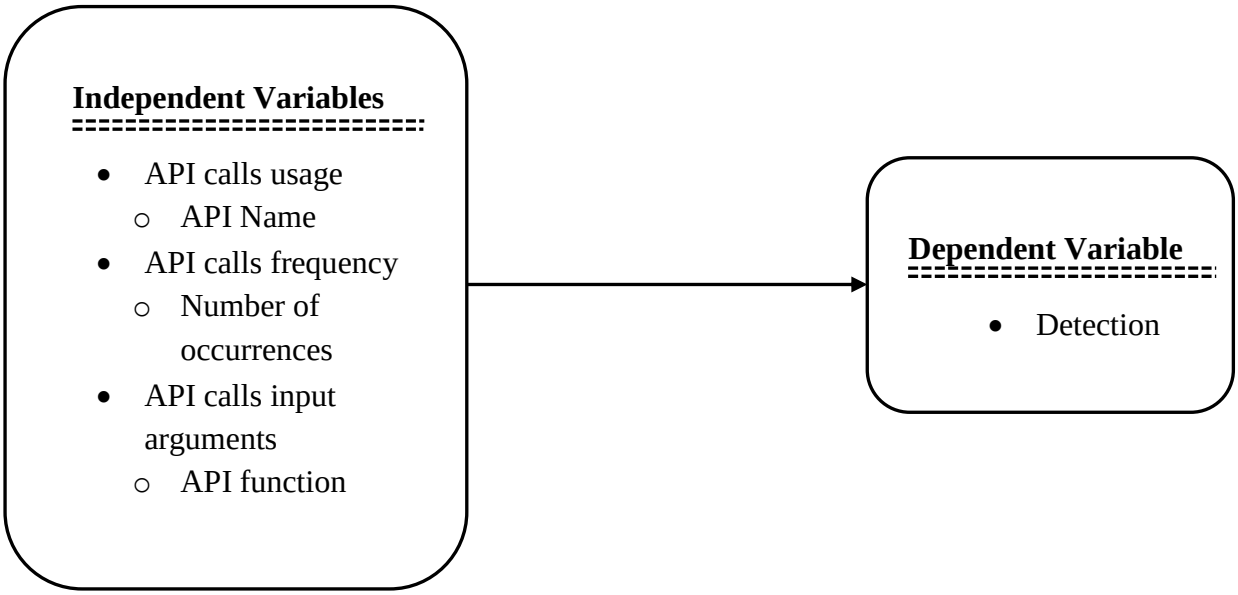


Figure 2.5 : Conceptual Framework (Author)

2.12 Operationalization of Variables

Variable	Sub-Variable	Indicator	Values
API calls usage	APIs called	API Name	Name
API calls frequency	API incidence	Number of occurrences	Frequency
API calls Input arguments	API activity	API function	Open, start, read, create, delete, modify, move
Detection	Prediction	Malicious infection	Stop, Allow

Table 2.2 : Operationalization of Variables

2.13 Summary

This chapter examined the literature about crypto-ransomware analysis as well as detection approaches and tools. The state-of-the-art in crypto-ransomware detection work has been presented. The motivation for investigating crypto-ransomware detection is evident: malicious software causes social and economic disruption, the consequences of ransomware infections are rising with time. We found that detecting crypto-ransomware at the earliest opportunity is crucial for minimizing harm. So the question arises, how early and how to create an early detection system? Time has been utilized as a cut-off point by some experts, however this may not accurately reflect the ransomware attack's stage. As a result, in this study, we proposed an early detection model at the pre-encryption stage, which is before any encryption has begun.

Figure 2.6 summarizes the obstacles encountered in crypto-ransomware detection, as well as the limitations of existing solutions.

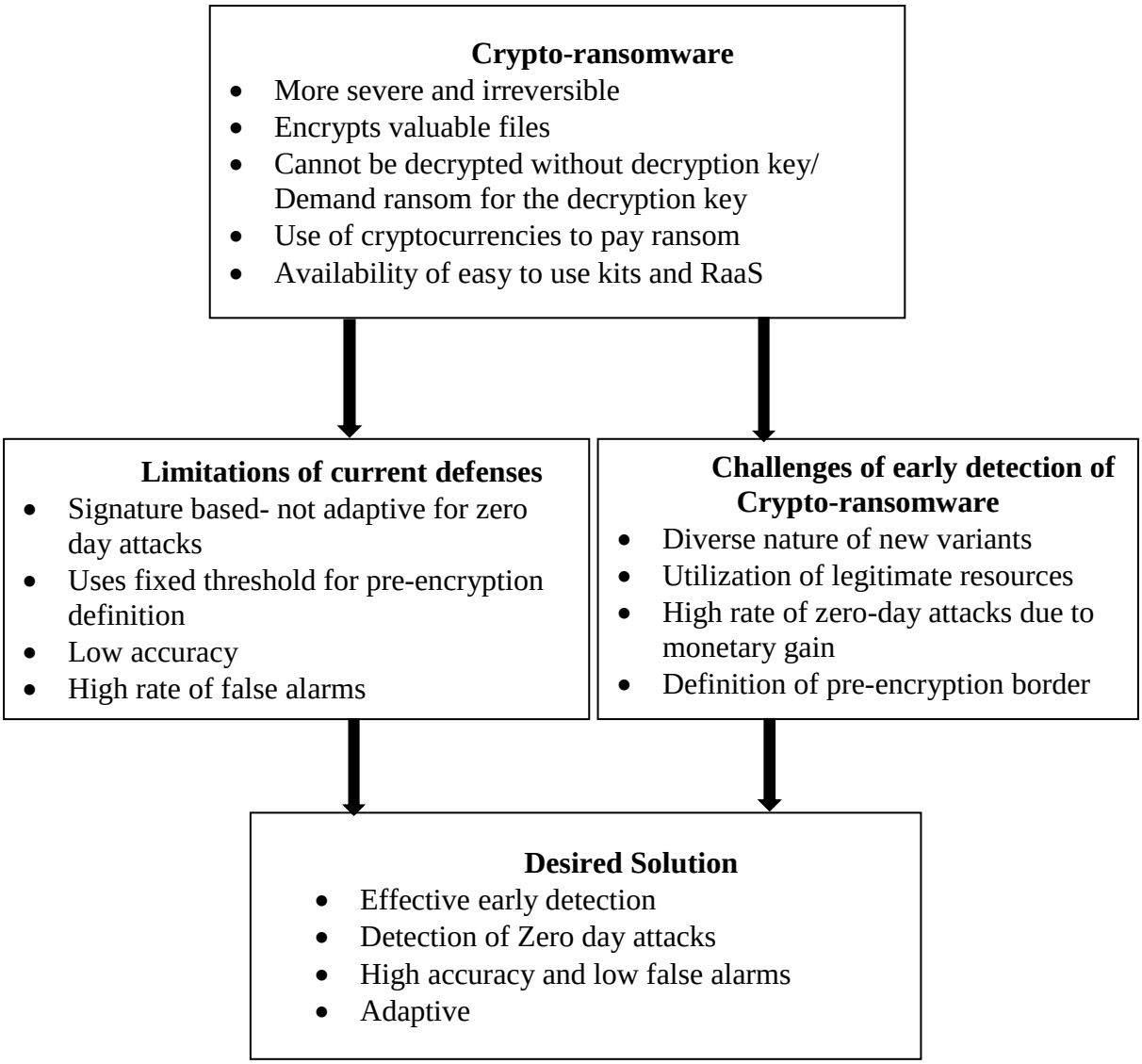


Figure 2.6 : Obstacles encountered in crypto-ransomware detection & limitations of existing solutions (Author)

CHAPTER THREE: RESEARCH METHODOLOGY

3.1 Introduction

The previous chapter provided a detailed examination of ransomware taxonomy, evolution, characteristics, and behaviours. The available crypto-ransomware analysis and detection approaches were also covered. According to the literature review, the damage caused by Crypto-ransomware is more often irreversible and severe (Kok et al., 2019). As a result, early detection of a ransomware attack prior to encryption, i.e. during the pre-encryption stage, is critical. This chapter will cover the research design for this work, as well as the virtual environment design and tools chosen for the crypto-ransomware detection model. Finally, the chapter will discuss the pros and cons of the chosen research design, tools, and approach.

3.2 Research Design

Kharaz et al (2016) defines research design "as a master plan specifying the methods and procedures for collection and analysing the needed information."

3.2.1 Method Selection – Controlled Experiment Process Design

Previous research work in the field of study has a large influence on the research method. While some ransomware research has been presented as case studies, the majority of previous academic research works on crypto-ransomware analysis have been performed in a virtual environment. (Kharaz et al. 2016). This is done to prevent the ransomware from compromising any unintended systems because of the dangerous nature of the real malicious samples being analysed. As a result, given the ethical constraints and security risks, a controlled environment research design was preferable for this work.

Annis (2008) presented a method for malware research that uses a controlled-environment research design. In light of this, a similar procedure tailored for crypto-ransomware research is depicted in figure 3.1. The experimental analysis involves dynamic analysis of crypto-ransomware to capture the behaviour throughout execution especially during the p re-encryption stage.

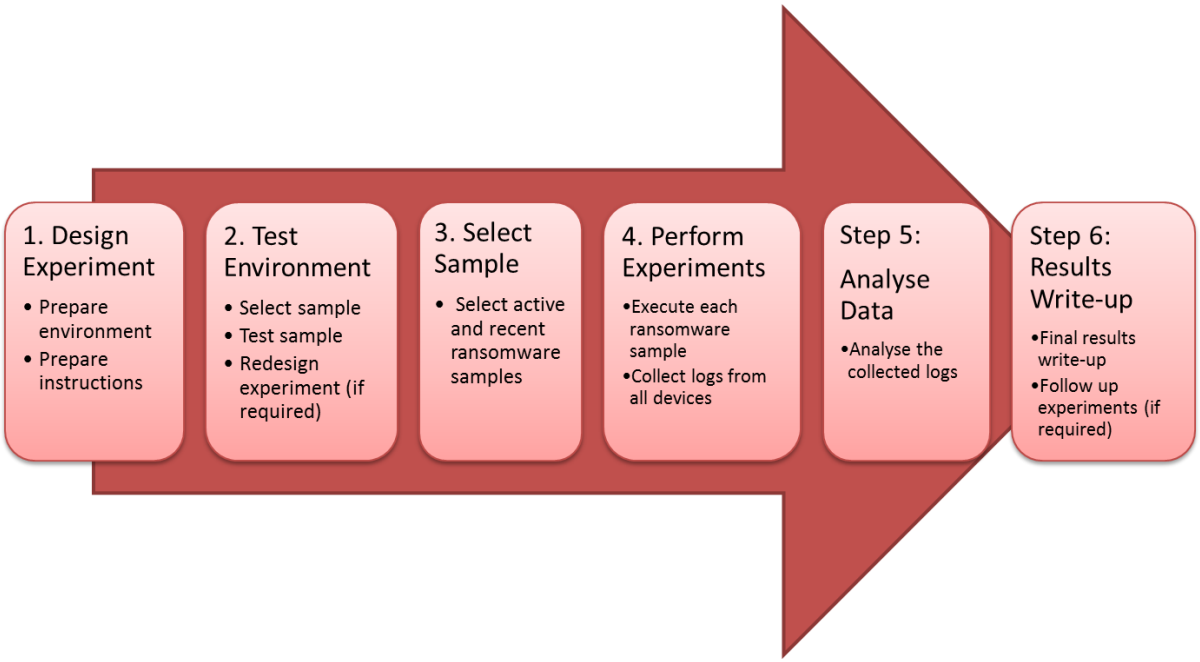


Figure 3.1 : Controlled experiment process design (Annis et al., 2008)

3.3 Target Population

A sufficient sample size was required for the researcher to make inferences about the general population. Secondary data obtained from online repositories were an appropriate sample for this study

3.3.1 Data collection procedure

Secondary data was used for this research. We collected ransomware samples from the VirusShare (VS) databases. VS is a malware repository that makes its collection available to security researchers, incident-response teams, forensic analysts, and the morbidly curious (Sgandurra, Muñoz-González, Mohsen & Lupu 2016). Anyone wishing to gain access to this database must first register and be verified by the administrator of the website. Anyone can download samples from these repositories without restriction. Furthermore, benign samples (goodware) were acquired from kaggle.com, a data science and machine learning online community platform.

3.3.2 Sample size analysis (Datasets)

The ransomware dataset contained 40,500 samples. Based on the study's requirements, which primarily included the use of encrypting malware, the researcher downloaded samples classified

as crypto malware by the majority of antivirus programs on virustotal. Those samples originate from various families, including WannaCry, Cerber, Petya, TeslaCrypt, and CryptoWall, among others. The goodware dataset comprised of 18,075 samples. The available datasets were large and comprehensive enough to cover the research objective while also providing knowledge for the learning algorithm to avoid under-fitting or over-fitting.

3.4 Data Analysis and Processing

This study utilizes dynamic analysis for features extraction. Dynamic analysis was selected because it is an efficient way for circumventing the polymorphism and packing strategies employed by crypto-ransomware attacks to avoid detection. According to Darem et al. (2019), "dynamic analysis methods are effective for detecting obfuscated malware because it is difficult for malware authors to hide the actual intent of the program." Furthermore, because the proposed early detection model seeks to detect attacks as early as possible in the runtime, dynamic analysis is employed. The malware sample is initially run in a controlled environment to discover its true behaviour. During analysis, various sorts of data may be obtained, including API calls, file system CPU registers, network traffic, processes, and memory, among others (Darem et al., 2019).

The Cuckoo Sandbox, a popular and extensively used platform for malware analysis, was used as the environment for analysis. Cuckoo Sandbox is a secure and isolated malware analysis system. It monitors and obtains the runtime behaviour of a program by running it in a sandbox (Kok et al., 2019). It records the information in a behavioural report, also known as the dynamic analysis report (Guarnier et al., 2012) in a JavaScript Object Notation (JSON) file format. The Cuckoo sandbox logs the API calls and categorizes them according to the kind of action done for example, network, crypto, process, services, registry, system, file, resources, and so on (Guarnier et al., 2012). The API calls category gives insight on the actions of the executable file.

3.4.1. Feature Extraction.

Feature extraction is the technique of extracting patterns from malicious programs' runtime data to describe their behaviour (Rhode, 2021). According to Aboaoja et al. (2020), feature extraction is a key phase in classification and detection process since it has a considerable effect on the performance of the model.

Obtaining attack patterns

The crypto-ransomware and benign samples in the dataset were run individually to capture the attack's features from run-time data. Each sample was subjected to dynamic analysis. Following the submission of the sample, the cuckoo sandbox agent captures the sample's procedure and collects run-time data, such as API calls, and puts it in a separate trace-file allocated to that sample. To ensure that the subsequent sample is unaffected by the previous infection, the guest-machine was returned to the initial, clean condition after each run. The trace files form the corpus from where the dataset was generated, and features extracted for training the detection model. Because APIs serve as input to the model, just the API calls and arguments were retained from each trace file's execution data, while all other data was purged.

3.4.2 Features selection

Feature selection involves selecting a subset of variables for use in predictive model construction. Feature selection methods choose a sub-set of the initial features to lower the input dimension. One goal of feature selection is to remove content from the original set of features that is redundant, irrelevant, or insignificant so as to reduce model overfitting (Aboaoja et al. 2020). By doing so, the model's performance could be improved. Furthermore, by lowering the dimensionality of the dataset for processing, the computational load is reduced (Aboaoja et al. 2020).

Because the ransomware's runtime data is kept in textual trace files, these files may be regarded as documents, while the API calls included within them can be regarded as terms (words). In this respect, the feature selection approach employed in this study is based on information retrieval.

Information retrieval is “part of computer science which studies the retrieval of information (not data) from a collection of written documents.” (Billhardt, Borrajo & Maojo, 2002). The goal of the retrieved documents is to “achieve a user’s information need.” (Billhardt et al., 2002). The procedure may be understood of as searching through a collection of documents, known as the corpus, to identify a specific piece of information that is related to a particular item, known as the query. Information retrieval techniques include vector space models, statistical confidence models, and latent semantic indexing models, among others.

According to Rieck et al., (2008), “the vector-space model for information retrieval represents documents and queries as vectors of weights, with each weight representing the

relevance of an index term in a document or a query." VSM implements full-text automatic indexing and relevance ranking (Billhardt et al., 2002).

Rocchio Relevance Feedback (RRF)

Rocchio Relevance Feedback is an algorithm used by information retrieval systems for automatic document processing and word classification (Rieck et al., 2008). It represents a method of adding relevant feedback information into the vector-space model (Billhardt et al., 2002). The Rocchio theory's main idea is clear, to move the query point away from non-relevant documents and toward relevant documents (Billhardt et al., 2002).

RRF uses the Term Frequency-Inverse Document Frequency (TF-IDF) weighted vectors to represent text documents (Billhardt et al., 2002)

Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF is a popular weighting technique in information retrieval and text mining. It is a numerical statistic that shows the importance of words in collections of documents or corpus (Belaoued et al. 2019). The importance of a word increases with its frequency of recurrence in the document. TF-IDF weights are typically made up of two words. The normalized term frequency is calculated by the first word (TF), it reflects the number of occurrences of a word in a document divided by the total word count in that document (Belaoued et al. 2019). The inverse document frequency is the second term (IDF) expressed as a logarithm of the number of documents in the corpus divided by the number of documents containing the specific word (Belaoued et al. 2019). Multiplying the two measurements yields the final weight.

In this study, TF-IDF obtains and ranks the most prevalent APIs being called by every kind of ransomware and by benign files. It calculates the features' cross-entropy between the population and the malware sample.

The formula used to calculate TF-IDF is presented in Equation 1

$$w\left(api_k^j\right) = tf\left(api_k^j\right) \cdot \log \frac{N}{idf(api_k)} \quad (1)$$

Where api_k indicates the K[th] API, $tf\left(api_k^j\right)$ is the term-frequency which counts the number of occasions the ransomware instance r_j in the dataset called the api_k. While, idf(api_k) is the inverse-document-frequency which counts the number of times a ransomware instance r_j in the dataset calls api i_k. N is the number of ransomware cases in the data-set.

The decision to use the TF-IDF approach is influenced by the fact that it is distinguished by its speed and efficacy (Darem et al., 2019,) both of which are key in our context i.e. crypto-ransomware detection. TF-IDF is also ideal for translating textual data to numerical representation (Darem et al., 2019) suited for feature extraction and classification since the derived features from binary files are in text format, i.e. the names of the API call functions. Moreover, the TF-IDF approach retains human-readable features which is helpful in understanding the analytics (Darem et al., 2019)

Pre-encryption border definition

The pre-encryption border indicates the instant at which crypto-ransomware begins to use cryptography-related APIs or/and functions. The pre-encryption border is created using trace files obtained following the dynamic analysis. In contrast to fixed time-based thresholding, DCRDM monitors the beginning of encryption for each instance to establish the pre-encryption stage boundary depending on the APIs related to cryptography being called.

To precisely define the pre-encryption stage border, the Rocchio Relevance Feedback is used to create the pre-encryption border vector.

The cryptography-related APIs utilized throughout the attack are included in this vector. The vector reflects the pre-encryption stage of a crypto-ransomware event, the initial appearance of any entry in the vector marking the end of the pre-encryption stage and the beginning of the encryption phase. The premise is that using any of the cryptography-related APIs signals the onset of an encryption operation that could mark the commencement of an attack. The first API distinguishes between the pre-encryption step and the actual encryption procedure.

Generating the initial subsets

The initial boundary vector known as V_{initial} containing cryptography-specific APIs was built from the trace files. The CryptoAPI calls were identified and placed in V_{initial} . Using V_{initial} the raw-data in every trace-file was split into three parts: InitialPre, InitialAt, and InitialPost. The InitialPre consists of API calls ranging from the initial API in the trace-file to the first instance of an entry in V_{initial} . The InitialAt is made up of the APIs found between the first and last instances of V_{initial} entries. The InitialPost consists the APIs obtained immediately after the last instance of an entry in V_{initial} till the trace file ends. The aim of this split is to retrieve all API calls made during the encryption stage (the InitialAt API calls) and identify API calls related to the encryption operation.

Incorporating these APIs into the border vector is critical to guaranteeing that the approach can determine the pre-encryption border prior to the attack.

Rocchio Relevance Feedback

Cryptography-related APIs comprise the fairly obvious cryptoAPIs as well as other APIs that take the cryptoAPIs functions as arguments. These APIs are found in the InitialAt subset. As a result, the InitialAt subset serves as the focus point for locating pre-encryption border vector entries. However, some APIs in the InitialAt subset are not cryptography-related, as a result, including all of them in the final border vector may lead to exclusion of numerous pre-encryption data. Consequently, the model may be unable to learn valuable patterns for pre-encryption. To eliminate the irrelevant APIs, RRF was used to determine the weight of every API in the InitialAt sub-set, where only those with a weight greater than the threshold were retained.

The original Rocchio-relevance-feedback defines the relevance of query phrases based on equation 2 (Salton & Buckley, 1990).

$$V_f = \frac{1}{n}\sum_{\text{relevant}} d_j - \frac{1}{N-n}\sum_{\text{irrelevant}} d_j \quad (2)$$

Where V_f represents the feedback-vector; N represents the size of the corpus; n represents the number of relevant trace-files; $\sum_{\text{relevant}} d_j$ is the relevant group and $\sum_{\text{irrelevant}} d_j$ is the irrelevant group.

The RRF method employed TF-IDF to generate two vectors: relevant-vector and irrelevant-vector. The relevant-vector was generated by executing TF-IDF on our main sub-set, the InitialAt sub-set. The vector is deemed 'relevant' since it was constructed using data from InitialAt, which comprises of APIs gathered throughout the encryption process. The irrelevant-vector was generated by executing TF-IDF on the InitialPre and InitialPost combined. After building the vectors, the improved Rocchio approach suggested in Salton & Buckley (1990) was used, as stated by equation 3.

$$V_f = V_{\text{initial}} + \frac{1}{n_1}\sum_{\text{relevant}} d_j - \frac{1}{n_2}\sum_{\text{irrelevant}} d_j \quad (3)$$

Where V_f represents the feedback-vector; V_{Initial} is the original-vector; $\sum_{\text{relevant}} d_j$ is the relevant-group (InitialAt) and $\sum_{\text{irrelevant}} d_j$ is the irrelevant-group (InitialPre & InitialPost). Since both the relevant and irrelevant sets were subsets of the same instance of the crypto-ransomware, equation 3, could be rewritten as in equation 4

$$V_f = V_{\text{Initial}} + \frac{1}{n}\left(\sum_{\text{relevant}} d_j - \frac{1}{2}\sum_{\text{irrelevant}} d_j\right) \quad (4)$$

The pre-encryption border vector was obtained from feedback vector, V_f using a threshold so that APIs with TF-IDF values that exceed or equal the threshold would be added into the vector selected V_s. The threshold was defined by averaging the TF-IDF values in the V_{Initial} entries. The average was calculated according to equation 5

$$W_{\text{avg}} = \frac{1}{Q}\sum_{i=0}^{i=q-1} w_i \quad (5)$$

Where w_{avg} represents the vector's weighted average; w_i represents the ith API weight; and $Q = \{0,1,2,3,\dots,q-1,q$ is the whole set of API calls.

To ensure that all cryptography-related APIs were retained in the pre-encryption border vector, V_s and V_{Initial} were merged to form V_{pb}. As a result, vector V_{pb} represents the pre-encryption border vector.

3.4.2 Model Training - machine learning algorithm

The machine learning algorithms were implemented using Python 3.6, which included several python packages such as Sklearn, Numpy, Keras, Pandas, Seaborn and Matplotlib. Microsoft Excel was also used to plot the graphs.

```
# Importing libraries and reading features list

import os
File: Cryptodataset
import pandas as import pdb; pdb.set_trace()
import numpy as numpy
import seaborne as seaborne
import matplotlib.pyplot as matplotlib

# Reading features list
with (open) ("Cryptodataset.txt" , 'Reading') as File:
    print (File.read())
```

Figure 3.2 **:** Importing python libraries

Following feature selection and the creation of the pre-encryption border, the resulting dataset is utilized to train the prediction model to identify ransomware before encryption begins. We divided the dataset in an 80:20 ratio. 80% of the dataset was utilized to train the predictive

model, with the remaining 20% used for testing. The conventional cross validation procedure with 10 K-Folds was utilized in validation.

```
Import train_test_split function
from sklearn.cross_validation import train_test_split
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.80, random_state=20)
# 80% training and 20% test
from sklearn.ensemble import RandomForestClassifier
clf=RandomForestClassifier(n_estimators=100)
# Train the model using the training sets
clf.fit(X_train,y_train)
# prediction on test set
y_pred=clf.predict(X_test)
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Figure 3.3 **:** Model training

The model is trained using Random Forest (RF) machine learning algorithm which was selected for this study, however for comparison purposes, four other learning algorithms are used. They include K-Nearest Neighbour (KNN), AdaBoost, Support Vector Machines (SVM) and Multi-Layer Perceptron (MLP)

Why Random Forest?

Random Forest has been widely used in many related studies. RF was chosen for this study because it performs well, particularly for discrete datasets, and can also categorize a large amount of data with a low error rate. Moreover, since the algorithm chooses trees at random, over-fitting is avoided, and scaling is not required for Random Forest to train efficiently. According to Fawagreh et al. (2014), "Random forest approach has proved its high accuracy and superiority."

3.5 Testing and Performance Evaluation

Evaluation metrics are crucial tools for measuring the performance of a machine learning-based predictive model (Kok et al., 2020). To evaluate the performance of the proposed model, standard metrics were used including accuracy, f-score, recall, precision, false negative rate (FNR), false positive rate (FPR), ROC-AUC and cross-validation.

3.5.1 Binary classification metrics

Binary classification predictions could be true or false, negative or positive. Classification predictions are classified into four categories. True positive predictions are positive predictions which are predicted correctly, whereas false positive predictions are positive predictions that are incorrectly predicted. The true negative predictions are negative predictions which are predicted correctly, whereas false negative predictions are negative predictions that are incorrectly predicted. (Taylor, 2017). The above four category values can be used to generate metrics expressing the efficacy of a binary classifier.

		Predicted Class	
		Negative (Normal)	Positive (Attack)
Actual Class	Negative (Normal)	True Negative (TN)	False Positive(FP)
	Positive (Attack)	False Negative (FN)	True Positive(TP)

Table 3.1 : Confusion Matrix for a binary classifier

Accuracy

Accuracy is a metric that measures the prediction model's ability to properly recognize as much data in a dataset as possible (Taylor, 2017). It calculates the proportion of correctly classified instances out of all instances. Accuracy isn't necessarily a good indicator of a model's ability to make correct predictions especially when dealing with imbalanced dataset.

In this study's context, accuracy is the ratio of all correct predictions (ransomware or not ransomware.)

$$\text{Accuracy} = \frac{TP+TN \text{ (correctly classified instances)}}{TP+TN+FP+FN \text{ (Total Population)}}$$

Detection rate / Sensitivity / Recall/ True Positive Rate (TPR)

This metric measures a prediction model's ability to properly predict that an actual positive will test positive. In this study's context, detection rate represents predicted ransomware that was correctly classified as ransomware.

$$\text{True Positive Rate (TPR)} = \frac{TP \text{ (correctly detected ransomware)}}{TP+FN \text{ (Positive population)}}$$

Specifity / True Negative Rate (TNR)

This metric measures a prediction model's ability to properly predict that an actual negative will test negative. In this study's context, specifity equates to the percentage of non-ransomware samples correctly predicted as such.

$$\text{True Negative Rate (TNR)} = \frac{TN}{TN+FP \text{ (Negative population)}}$$

Fallout /False Positive Rate (FPR)

This metric measures frequency with which the prediction model misidentifies negative trained data as positive trained data (Taylor, 2017). In this study's context, FPR equates to the percentage of benign samples incorrectly predicted as ransomware.

$$\text{False Positive Rate (FPR)} = \frac{FP \text{ (benign instances wrongly classified)}}{TN+FP \text{ (Negative population)}}$$

Precision (PPV)

This metric measures the prediction model's ability to distinguish between data from positive and negative conditions. It evaluates how true the positive prediction.

$$\text{Precision} = \frac{TP}{TP+FP \text{ (Predicted Positive)}}$$

F-Measure

Represents the mean between recall and precision. It defines the model's robustness and precision.

$$\text{F-Measure} = 2 * \left(\frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \right)$$

3.5.2 Cross Validation

Cross validation is a method for determining how well a prediction model performs when given a different set of data (Chen et al., 2018). A model in machine learning is fed known data for training and an unknown set of data for testing. Cross validation evaluates the model during the training phase for the purposes of minimizing issues like overfitting (Kok et al., 2020).

Holdout Method

The holdout technique divides a dataset into two distinct parts. One set is the training set, while the other is the testing set (Taylor, 2017). The training data is for building the prediction model, while the testing data is for evaluating the model's accuracy.

We divided the dataset in an 80:20 ratio. 80% of the dataset was utilized to train the predictive model, with the remaining 20% used for testing.

K-Folding Method

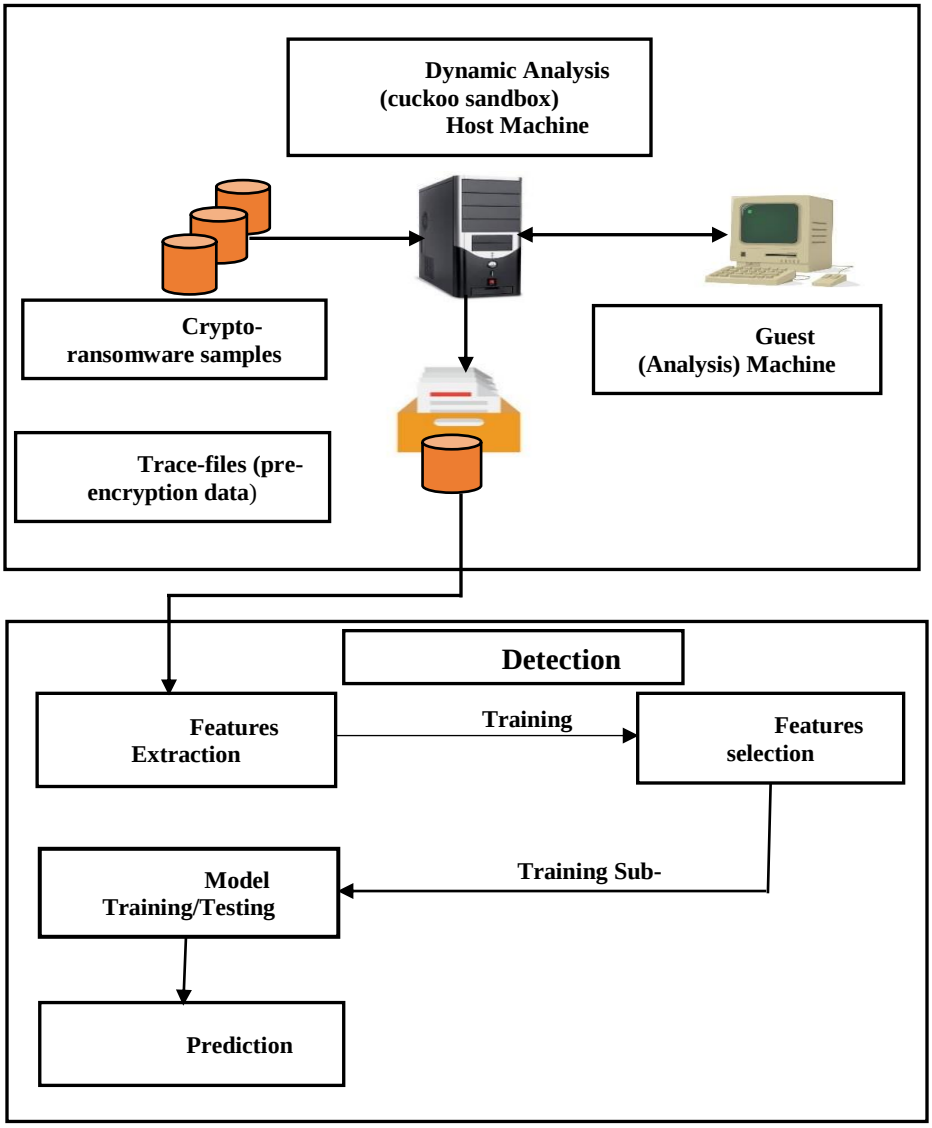
According to Taylor (2017), "The k-fold method divides a data set into k subsets where during one of k different trials each subset of data will serve as the training data while the other k-1 subsets are combined together to form the test data set." The model's accuracy is determined by calculating the average error in all k trials. We used 10-fold cross-validation to ensure that the result were as objective as possible (Kok et al., 2019).

3.5.3 ROC Curve

Additionally, the testing phase included areas under the curve of the receiver operating characteristic (AUC ROC). Because ROC is the curve of FPR against TPR, the AUC of ROC examined the predictive model's capacity to differentiate between negative and positive outcomes.

3.6 Summary

The Architecture overview of the proposed model is shown below



3. 4 : Architecture overview of the proposed model

Table 3.2 below shows the strategies used to accomplish the study's objectives.

Objective	**Method/Strategy**	**Description**
To conduct a review of the literature on crypto-ransomware attacks, including current analysis and detection methods	Literature Review	- The researcher looked into journals, books and conference papers.
To develop a crypto-ransomware detection model for the pre-encryption stage using machine learning.	Controlled experiment process design	- Sample collection (secondary data) - Analysis and feature extraction (Cuckoo sandbox) - Pre-encryption border definition / features selection (RRF, TF-IDF) - Model development (ML technique - Random forest & four others ML algorithms)
To test and validate the effectiveness of the crypto-ransomware detection model	Model Evaluation - Experiments	- Employed the evaluation metrics, (Binary Classification Metrics, Cross validation and AUC ROC curve) -Comparison with related works

Table 3.2 : Strategies to achieve study objectives

CHAPTER FOUR: DATA ANALYSIS, FINDINGS AND DISCUSSION

4.1 Introduction

This chapter builds on the preceding chapter, methodology. It serves as an application of the methods stated in chapter three. In this chapter, we will discuss the tasks involved in developing the crypto-ransomware detection model. We will also go through the findings for each research objective.

4.2 Descriptive Statistics

We started off by collecting samples (ransomware and goodware) from VirusShare (VS). The ransomware dataset was composed of 40,500 samples. The samples were classified according to their families and described as crypto malware by most of antivirus programs listed on Virustotal. Those samples represent different families such as Cerber, TeslaCrypt, CryptoWall, Petya and WannaCry. The goodware dataset consisted of 18,075 acquired from kaggle.com, a data science and machine learning online community platform.

Family	Number of Samples
Cryptolocker	5460
Cryptesla	4630
Filecryptor	3928
Cryptowall	3625
CryptXXX	3134
Petya	2932
Reventon	2871
CTB_Locker	2664
Locky	2552
Wannacry	1924
Cyrar	1825
Teslacrypt	1532
Satana	1341
Cerber	1323
Sage	759

Table 4.1 : Families of crypto-ransomware utilized in this research

The Environment Setup

The dynamic analysis of ransomware was carried out in a controlled environment built on a computer with Core i5 processor @ 2.3 GHZ and 8 GB RAM. The operating system for the host

computer is Linux Ubuntu 18 whereas the guest machine runs a Windows 7 guest operating system. To build a convincing setting in the guest machine where the samples were run, several applications were installed, including Google Chrome, Mozilla Firefox, Adobe Acrobat Reader, and Microsoft Office. Additionally, some files, such as MS Word documents, PPT, Excel, PDF, GIF, and JPG, were created in multiple places on the guest machine's local storage. The proposed methodologies, analysis, and findings were implemented using Python 3.6 modules.

4.3 Research Findings

4.3.1 Objective One Results

Objective 1 – To identify common characteristics and behaviours of crypto-ransomware

The majority of crypto-ransomware exhibits some common characteristics and behaviours that may indicate the presence of a malicious program. Crypto-ransomware behaviours are broadly classified into two categories: pre-encryption behaviours and file encryption process behaviours. Among the pre-encryption behaviours are payload persistence, restriction of system restore, stealth/evasion techniques, environment mapping, communication masking and privilege elevation. The file encryption process behaviours identified include increased file system activity, high Input/output accesses, high resource usage and changes in registry activity.

4.3.2 Objective Two Results

Objective 2 – To review the current methods for crypto-ransomware analysis and detection

The purpose for ransomware analysis is to gain a better understanding of the behaviour and functionality of various ransomware families. Malware analysis is generally categorized into static and dynamic analysis. Static analysis examines binary file contents, whereas dynamic analysis investigates a process's behaviour and actions during execution (Gandotra et al., 2014).

The methods for crypto-ransomware detection are based on distinguishing between ransomware and benign application. They include signature-based, behaviour-based, and hybrid approaches. Behaviour based methods are classified as data-centric or process-centric. Machine learning detection approaches that are part of process-centric solutions were also investigated.

Research shows that in recent years, the use of machine learning to detect malware automatically based on its dynamic behaviours has gained popularity. Because of their self-learning capabilities, ML approaches have been deemed effective, and they have significantly improved detection rates on a wide scale (Chen, Yang, Paul & Sahita, 2018).

Several research works on early detection of crypto-ransomware using machine learning exist. They propose a time-based threshold strategy to define the boundaries of the pre-encryption stage, from which the attack patterns are gathered and utilized to develop an early detection model (Al-rimy et al., 2018). The major disadvantage of this approach is that not all samples begin encryption at the same time. This study takes a similar approach, but instead of using fixed thresholding, the proposed solution uses dynamic thresholding, which involves tracking each cryptography-related API independently.

4.3.3 Objective Three Results

Objective 3 - To develop a crypto-ransomware detection model for the pre-encryption stage using Random Forest algorithm

Given the security risks and previous research on ransomware, a controlled environment research design (Annis et al., 2008) was used for this work to undertake the dynamic ransomware analysis. This model is designed on top of cuckoo sandbox, a popular and extensively used platform for malware analysis, so that ransomware analysis and detection may be done without risk of data loss or file encryption. However, any alternative dynamic analysis tool might be used to implement the suggested model. The sandbox was designed in line with the instructions in Guarnier et al. (2012).

The Proposed DCRDM

In this section, the proposed Dynamic Crypto-Ransomware Detection Model (DCRDM) model is presented and discussed briefly. DCRDM has been designed in four stages, namely, features extraction, features selection, training/testing and prediction.

1. Features extraction

The crypto-ransomware and benign samples are introduced one by one into the host machine's cuckoo sandbox instance. The cuckoo agent runs the binary files provided and captures the attack's features.

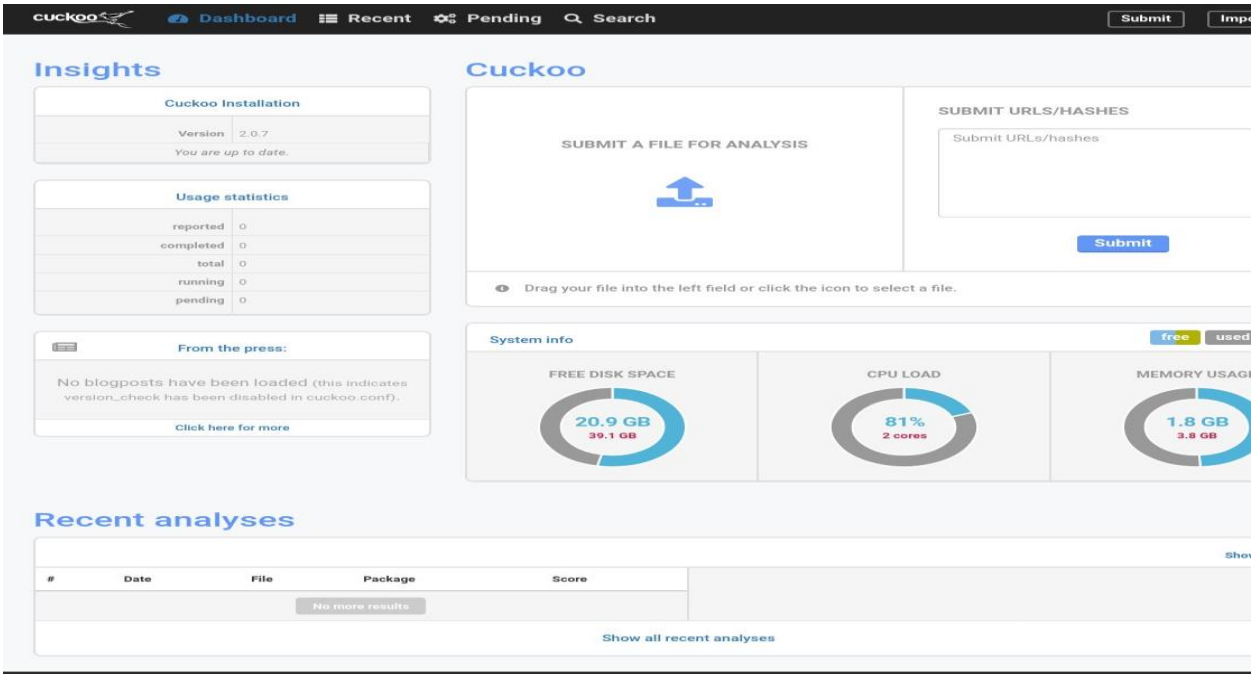


Figure 4.1 : Cuckoo sandbox homepage

The activity report in JSON format for each file is stored in the sample's own trace-file. The API list and API sequences, which are required for the detection model, are then retrieved from the activity reports.

APIs List

These features show the names of all APIs called by a certain program, as well as how many times they occur throughout the execution procedure

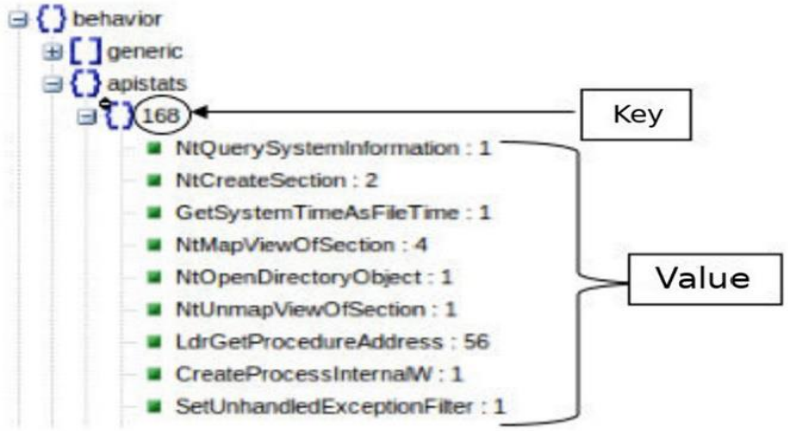


Figure 4.2 : API list activity report

2. Features selection

Rocchio relevance feedback with TF-IDF generated the pre-encryption border vector, as described in section 3.4.1 of the methodology. This procedure resulted in the definition of the pre-encryption border vector, as seen in Table 4.2. Every API signifies a border vector entry. Each entry's weight is also indicated as determined by equation (4).

API	Weight (feature Importance)
Cryptencrypt	0.896
Cryptdecrypt	0.821
Cryptacquirecontexta	0.812
Cryptacquirecontextw	0.779
Cryptunprotectdata	0.723
Cryptcreatehash	0.604
Crypthashdata	0.593
Cryptgenkey	0.581
Cryptexportkey	0.556
Encryptmessage	0.524
Decryptmessage	0.521
Cryptdecodeobjectex	0.509
CryptGetObjectUrl	0.503
CryptReleaseContext	0.499
CryptGetHashParam	0.474
CertGetNameStringW	0.476
Certcontrolstore	0.401
Certopenstore	0.401
Ntcreateprocessex	0.379
Removedirectorya	0.347
Ntdeletefile	0.343
Ntdeletevaluekey	0.303
Rtlcompressbuffer	0.297
internetgetconnectedstate	0.290
Ntqueryfullattributesfile	0.289
Openservicea	0.273

Table 4.2 : The pre-encryption border vector.

The variables considered to be significant were:

API calls usage: The most prevalent APIs that were invoked by the ransomware. Table 4.3 shows the names of the common APIs being called.

CryptUnprotectData	CryptGenKey	CryptEncrypt
CryptDecrypt	EncryptMessage	CryptExportKey
CryptDecodeObjectEX	CryptReleaseContext	CryptHashData
CryptCreateHash	CryptAcquireContextA	CryptGetHashParam
CertGetNameStringW	CryptAcquireContextW	CryptGetObjectUrl

Table 4.3 : Names of cryptoAPIs captured

API calls input arguments: The APIs that accept cryptography related APIs and/or functions as parameters. They indicate the file's precise behaviour.

Among them are, CertGetNameStringW, Certcontrolstore, Certopenstore, Ntcreateprocessex, Removedirectorya, Ntdeletefile, and Ntdeletevaluekey

API calls frequency: The APIs that invoke excessive and redundant cryptography related APIs. The weights and rank for the API occurrences are shown in table 4.2.

Table 4.2 demonstrates that the cryptoAPIs that are explicit were scored higher than other APIs related to cryptography. This means that the proposed method was effective in identifying the cryptograph-related APIs more precisely among all APIs included in the InitialAt subset.

3. Training and testing

The random forest machine learning algorithm is used to train the model. However, four more learning algorithms were also trained for comparison purposes. They include Support Vector Machines (SVM), K-Nearest Neighbour (KNN), Multi-Layer Perceptron (MLP), and AdaBoost. Model testing is discussed in the next section.

4. Prediction

The prediction capability in our early detection of crypto-ransomware model is explained by the good results produced. It can be shown from the results that the model achieved a detection accuracy of 98.6%, F score of 99.3%, ROC-AUC of 99.1%, Precision of 98.3% and a recall of 99.1%.

Table 4.4 Shows experiment results using different classifiers on the pre-encryption data-set (retrieved using the boundary definition technique proposed)

Metric	DCRDM	SVM	AdaBoost	KNN	MLP
Accuracy	0.986	0.915	0.975	0.978	0.935
F1-score	0.993	0.968	0.988	0.989	0.977
Precision	0.983	0.899	0.969	0.973	0.925
Recall	0.991	0.993	0.967	0.979	0.998
ROC-AUC	0.991	0.935	0.959	0.961	0.942

Table 4.4 : Experiment results using different classifiers

Figure 4.3 shows the model's performance at varying time stamps. Although the performance varies from second to second, it is consistently high (more than 97%) during execution, even within the initial few seconds.

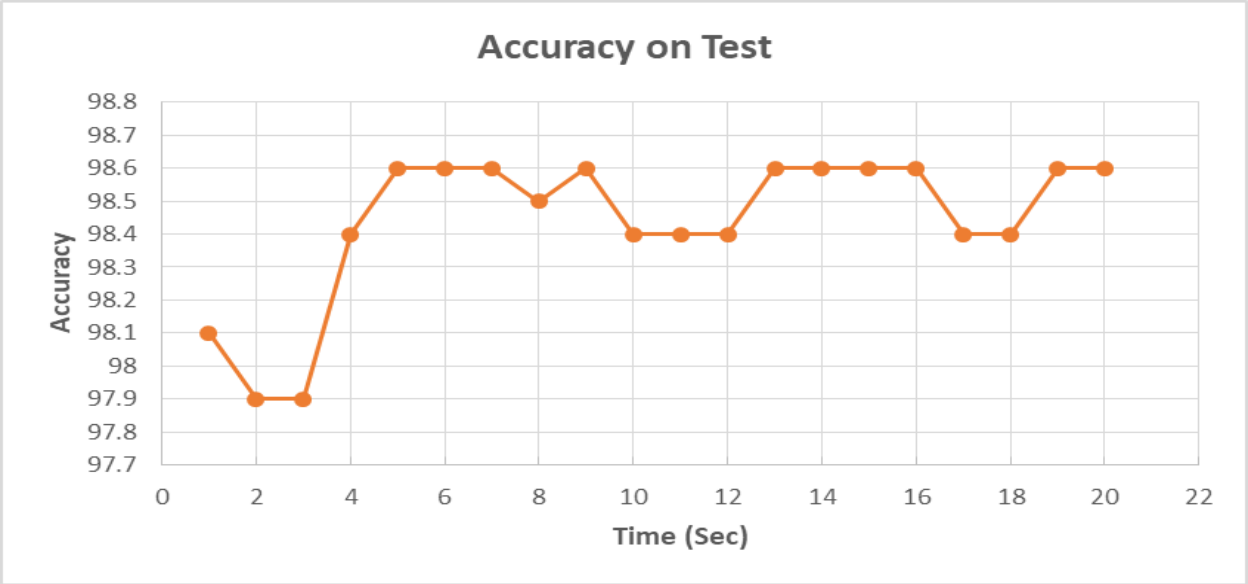


Figure 4.3 : DCRDM performance at varying time stamps

4.3.4 Objective Four Results

Objective 4 - To test and validate the effectiveness of the crypto-ransomware detection model

The effectiveness of the proposed model was assessed using several performance evaluation metrics, as stated in section 3.5, including accuracy, f-score, recall, precision, false positive rate (FPR), false negative rate (FNR), ROC-AUC and cross-validation. We tested DCRDM against other machine learning algorithms used in malware classification and also conducted a comparative analysis with models presented by related works.

The first comparison is between DCRDM and four different learning algorithms as shown in table 4.4 above. According to the findings, the model attained a detection accuracy of 98.6%, Precision of 98.3%, recall of 99.1%, F score of 99.3% and a ROC-AUC of 99.1%.

Accuracy represents the ratio of all correct predictions. A higher precision means the lower the number of false positive errors committed by the classifier. The higher the recall value, the fewer cases misclassified as negative. A high ROC-AUC value indicates that the model can differentiate between negative and positive occurrences well.

The highest accuracy on unseen test-set within the first 20 seconds of execution is shown in Table 4.5, along with the related false positive rate (FPR) and false negative rate (FNR). The highest accuracy of 98.6% was achieved in the fifth second of execution. However, as shown in figure 4.3, performance is consistently high (greater than 97%) even in the first few seconds. The model's sustained stability at various time intervals is an indication that it can predict the threat's appearance earliest possible.

Classifier	Accuracy (%)	Time (s)	FPR (%)	FNR (%)
DCRDM	98.6	5	1.9	2.7
SVM	91.5	10	3.4	5.1
AdaBoost	97.5	8	4.7	7.2
KNN	97.8	4	1.9	3.2
MLP	93.5	14	3.9	5.5

Table 4.5 : The highest accuracy on unseen test set

A low FPR indicates that the model is less prone to generating false alarms, while a low FNR indicates that it is less likely to predict a negative class wrongly.

The highest average accuracy during 10-fold cross validation on the training-set within the first 20 seconds of execution is shown in Table 4.6, along with the related false positive rate (FPR) and false negative rate (FNR).

Classifier	Accuracy (%)	Time (s)	FPR (%)	FNR (%)
DCRDM	98.4	4	2.3	2.1
SVM	90.1	10	4.5	3.6
AdaBoost	95.6	4	6.7	5.2
KNN	97.1	3	6.4	5.7
MLP	92.9	13	7.2	5.8

Table 4.6 : Highest average accuracy during 10-fold cross validation

We also conducted a comparative analysis with few models presented by related works. This is shown in Table 4.7 and Figure 4.4

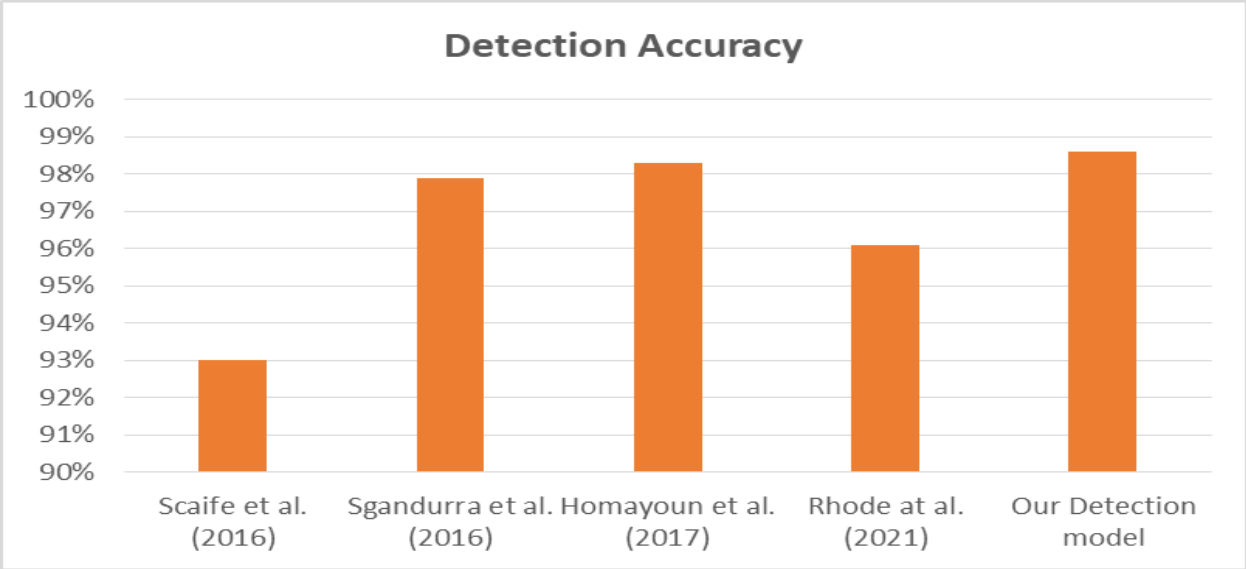


Figure 4.4 : Comparison of detection accuracy

The detection accuracy is the proportion of all correct predictions. Accuracy, however, isn't necessarily a good indicator of a model's ability to make correct predictions. Despite being a useful assessment metric, it may not be adequate owing to the nature of the dataset. If a dataset is imbalanced, for instance, there is a bias in favour of the largest represented group. To address this issue, other performance metrics including F-score, recall, precision, FPR and FNR were used in this study

Technique	Accuracy	Precision	F-score	Recall	FPR	FNR
Scaife et al. (2016)	93%	-	-	-	-	6%
Sgandurra et al. (2016)	97.9%	-	-	96.3%	1.6%	2.3%
Homayoun et al. (2017)	98.3%	-	98%	98%	0.9%	-
Rhode at al. (2021)	96.1%	-	-	-	3.1%	4.7%
Our Detection model	98.6%	98.3%	99.3%	99.1%	1.9%	2.7%

Table 4.7 : Comparison of detection results with related works

4.4 Discussion of Results

This research proposed the idea of early crypto-ransomware detection depending on dynamic pre-encryption phase boundaries. The DCRDM was proposed and developed to dynamically define the pre-encryption stage border in the lifecycle of crypto-ransomware and extract the features before encryption that were utilized in training the detection model. In contrast to fixed time-based thresholding, DCRDM utilizes a dynamic thresholding approach, which entails creating a border vector prior to encryption that contains the cryptography-related APIs being called. This vector's entries were selected based on their computed weights utilizing the proposed approach.

The comparative findings in Figure 4.4 show that our approach was more effective in identifying the pre-encryption border than the fixed-time thresholding method used in related works. This is due to the dependence on cryptography-related APIs to monitor the start of encryption for every sample, irrespective of how long it took for encryption to begin. Table 4.3 experimental findings show that the cryptoAPIs that are explicit were scored higher than other APIs related to cryptography. This means that the proposed method was effective in identifying the cryptography-related APIs more precisely among all APIs included in the InitialAt subset.

Table 4.7 and Figure 4.4 also demonstrate how well the model performed compared to the related works in Scaife et al. (2016), Sgandurra et al. (2016), Homayoun et al. (2017) and Rhode et al. (2021). Accordingly, the DCRDM dynamic thresholding proved more successful than other thresholding strategies in capturing the behavioural component of crypto-ransomware attacks at earlier stages. This is due to the DCRDM's ability to incorporate more data than other strategies since it monitors the onset of encryption for each instance separately. It utilizes the supplementary traits that certain ransomware samples in the dataset may have, allowing it to collect more pre-encryption data. Despite the fact that some ransomware instances begin encryption pretty fast, there are certain cases when the encryption begins late. As a result, dynamic thresholding compensates for a lack of data in samples that begin encryption quite fast with data gathered by samples that begin encryption late.

4.5 Summary

This chapter covered the experimentations linked to the development of an early-stage crypto-ransomware detection model. The dataset utilized was described; the dataset contained ransomware and benign files whose run-time behavioural activities were captured. The model was trained by features selected after creation of the pre-encryption border vector. The results indicated

that our model could detect crypto-ransomware with 98.6% accuracy and False Positive Rate of 1.9%. The findings also show that DCRDM was effective in defining the pre-encryption stage border and extracting the features most closely related to this stage precisely than related studies. This confirms that feature selection can enhance or degrade a learning algorithm's performance.

CHAPTER FIVE: CONCLUSIONS AND RECOMMENDATIONS

5.1 Introduction

The goal of this research was to create a scheme for detecting crypto-ransomware attacks during the pre-encryption stage. The solution was effective in defining the crypto-ransomware lifecycle's pre-encryption borders, extracting the distinctive features that depict the attack patterns in this stage, and incorporating the features into the creation of the crypto-ransomware detection model.

5.2 Conclusions

Crypto ransomware is a particularly dangerous kind of malware. It is distinguished by its irreversible effect even after detection and removal. As such, early detection is critical to safeguarding user data and files from being held for ransom. However, the idea of early crypto-ransomware detection is constrained in how it determines the pre-encryption border. The existing studies either utilizes a set threshold to define the pre-encryption border or utilize the entire data obtained during the attack which is ineffective in early detection.

In this study, an early detection model for crypto-ransomware was proposed. The model focuses on API calls and functions to detect malware encryption operation. APIs allow programs to interact and send information to one another; once we know which APIs are being called, the machine learning model can identify and halt the process. This gives us control and the ability to examine requesting application to determine whether it is a ransomware and then terminate it before it encrypts any files.

The model creates the pre-encryption border vector, which comprises all cryptography-related APIs responsible for determining the pre-encryption stage border during the crypto-ransomware lifecycle. The model proved more effective than previous works in defining the pre-encryption stage border of crypto-ransomware attacks.

Four classifiers including Support Vector Machines (SVM), K-Nearest Neighbour (KNN), Multi-Layer Perceptron (MLP), and AdaBoost, were utilized to evaluate the model's classification abilities. Furthermore, a comparison was done between the related works and our model. The findings show that DCRDM performed better across most calculated metrics, including accuracy, precision and recall. DCRDM also has a low FPR of 1.9%, indicating that it is highly unlikely to misclassify a benign program. This demonstrates the effectiveness of the proposed approach in precisely defining and extracting the features most closely related with the pre-encryption stage,

and then using these features to build the detection model. Consequently, we can conclude that the proposed scheme may well be used for early detection of crypto-ransomware attacks during the pre-encryption stage.

5.3 Contributions of the study

The damage caused by crypto-ransomware is more often irreversible and severe. As a result, early detection of a ransomware attack prior to encryption, i.e. during the pre-encryption stage, is critical. Therefore, the major contribution of this research was the development of a Dynamic Crypto-Ransomware Detection Model (DCRDM) based on the initial appearance of any APIs related to cryptography to establish the pre-encryption stage boundary, from which the features are extracted and used in training the prediction model. The findings show that DCRDM was effective in defining the pre-encryption stage border and extracting the features most closely related to this stage more precisely than related works.

Additionally, in meeting the research objectives, this study also provided the following contributions.

1) The identification of key characteristics and behaviours of crypto-ransomware attacks, with a focus on the critical pre-encryption period.
2) The use of a dynamic pre-encryption border defining approach for determining the pre-encryption stage border in the lifecycle of the crypto-ransomware. This helps in overcoming data insufficiency while attempting to extract the features of the attack during the pre-encryption stage.

5.4 Limitations of the study

To study ransomware using dynamic analysis, the ransomware must be sandboxed in a setting that permits for a thorough examination while also stopping it from propagating further. This strategy, however, has limitations, including the possibility for ransomware to infect the host machine and malware's inability to execute after detecting the virtual environment. In some instances, some malware programs might identify the analysis setup by utilizing anti-virtual system and anti-debugger methods and either terminate themselves or fail to reveal the actual behaviour.

Another limitation is the dependence use of Windows API, DCRDM might fail to detect ransomware that employs its own native encryption code. Therefore DCRDM should be used as a compliment detection system rather than the only ransomware detection system.

The dynamic analysis used in this study allows for direct monitoring and interaction with the malware in order to analyse its behaviour; however, the dynamic analysis introduces some prediction latency.

Due to the difficulty in obtaining sufficient new encrypting malware datasets, an old dataset (2017) was used. The datasets may be inadequate indication of how a model would function today.

5.5 Recommendations for future research

The DCRDM result is quite encouraging; however, it may need to be improved further before it is widely adopted. This is due to the difficulty that most people may have installing and configuring supporting applications such as cuckoo sandbox. Therefore, for future work, DCRDM should be built as a stand-alone application, without the requirement for a separate setup of supporting programs.

This study focused primarily on malware that affects Microsoft windows, the world's most popular desktop operating system. This work could be expanded to investigate if the model can detect malicious programs running on other operating systems.

Our findings also show that behavioural data could reveal important information determining if a file is malicious or not depending solely on its initial actions, even if the model has never encountered a certain malware type. This means that dynamic analysis might be effective in end-point antivirus applications while keeping in mind the concerns of dynamic solutions such as one, making feature analysis as quick as possible, and two, being able to distinguish between legit encryption and crypto-ransomware activities. We believe that dynamic analysis is key to improving crypto-ransomware detection.

Malware detection samples evolve over time as a result of technology advancements, prospects for cybercrime, and many other factors. This necessitates the collection of fresh datasets for research purposes, as old datasets may be may be an insufficient representation of how a model would perform presently.

Lastly, it is not an issue of "if," but of "when" an attack will occur, thus it is imperative to be prepared.

REFERENCES

Abidin, S., Kumar, R., & Tiwari, V. (2012). A review report on cryptovirology and cryptography. *International Journal of Scientific & Engineering Research*, *3*(11), 1.

Aboaoja, F. A., Zainal, A., Ghaleb, F. A., Al-rimy, B. A. S., Eisa, T. A. E., & Elnour, A. A. H. (2022). Malware Detection Issues, Challenges, and Future Directions: A Survey. *Applied Sciences*, *12*(17), 8482.

Ahmad Azami, N. I., Yusoff, N., & Ku-Mahamud, K. R. (2018). Fuzzy discretization techique for bayesian flood disaster model. *Journal of Information and Communication Technology*, *18*(2), 167-189.

Al-Rimy, B. A. S., Maarof, M. A., Alazab, M., Shaid, S. Z. M., Ghaleb, F. A., Almalawi, A., ... & Al-Hadhrami, T. (2021). Redundancy coefficient gradual up-weighting-based mutual information feature selection technique for crypto-ransomware early detection. *Future Generation Computer Systems*, *115*, 641-658.

Alzahrani, A., Alshehri, A., Alshahrani, H., Alharthi, R., Fu, H., Liu, A., & Zhu, Y. (2018, May). RanDroid: structural similarity approach for Detecting ransomware applications in android platform. In *2018 IEEE International Conference on Electro/Information Technology (EIT)* (pp. 0892-0897). IEEE.

Annis, J. (2008). *Zombie Networks: An investigation into the use of anti-forensic techniques employed by botnets*. Technical Report 22, The Open University, Walton Hall, Milton Keynes, MK7 6AA United Kingdom.

Aslan, Ö. A., & Samet, R. (2020). A comprehensive review on malware detection approaches. *IEEE Access*, *8*, 6249-6271.

Baldwin, J., & Dehghantanha, A. (2018). Leveraging support vector machine for opcode density based detection of crypto-ransomware. In *Cyber threat intelligence* (pp. 107-136). Springer, Cham.

Beaman, C., Barkworth, A., Akande, T. D., Hakak, S., & Khan, M. K. (2021). Ransomware: Recent advances, analysis, challenges and future research directions. *Computers & Security*, *111*, 102490.

Belaoued, M., Boukellal, A., Koalal, M. A., Derhab, A., Mazouzi, S., & Khan, F. A. (2019). Combined dynamic multi-feature and rule-based behavior for accurate malware detection. *International Journal of Distributed Sensor Networks*, *15*(11), 1550147719889907.

Biasini, N., Esler, J., Herbert, N., Mercer, W., Olney, M., Taylor, M., & Williams, C. (2015). Threat spotlight: Cisco talos thwarts access to massive international exploit kit generating $60 m annually from ransomware alone. *Cisco Talos. Retrieved from http://www. talosintel. com/angler-exposed*.

Billhardt, H., Borrajo, D., & Maojo, V. (2002). A context vector model for information retrieval. *Journal of the American Society for Information Science and Technology*, *53*(3), 236-249.

Branche, P. O. (2017). *Ransomware: An Analysis of the Current and Future Threat Ransomware Presents*. Utica College.

Carlin, D., O'Kane, P., & Sezer, S. (2019). A cost analysis of machine learning using dynamic runtime opcodes for malware detection. *Computers & Security*, *85*, 138-155.

Chen, L., Yang, C. Y., Paul, A., & Sahita, R. (2018). Towards resilient machine learning for ransomware detection. *arXiv preprint arXiv:1812.09400*.

Choudhary, S. P., & Vidyarthi, M. D. (2015). A simple method for detection of metamorphic malware using dynamic analysis and text mining. *Procedia Computer Science*, *54*, 265-270.

Chumachenko, K. (2017). Machine learning methods for malware detection and classification.

Cimitile, A., Mercaldo, F., Nardone, V., Santone, A., & Visaggio, C. A. (2018). Talos: no more ransomware victims with formal methods. *International Journal of Information Security*, *17*(6), 719-738.

Continella, A., Guagnelli, A., Zingaro, G., De Pasquale, G., Barenghi, A., Zanero, S., & Maggi, F. (2016, December). ShieldFS: a self-healing, ransomware-aware filesystem. In *Proceedings of the 32nd Annual Conference on Computer Security Applications* (pp. 336-347).

Darem, A. A., Ghaleb, F. A., Al-Hashmi, A. A., Abawajy, J. H., Alanazi, S. M., & Al-Rezami, A. Y. (2021). An adaptive behavioral-based incremental batch learning malware variants detection model using concept drift detection and sequential deep learning. *IEEE Access*, *9*, 97180-97196.

Diro, A., Reda, H., Chilamkurti, N., Mahmood, A., Zaman, N., & Nam, Y. (2020). Lightweight authenticated-encryption scheme for Internet of Things based on publish-subscribe communication. *IEEE Access*, *8*, 60539-60551.

El Naqa, I., & Murphy, M. J. (2015). What is machine learning?. In *machine learning in radiation oncology* (pp. 3-11). Springer, Cham.

Fawagreh, K., Gaber, M. M., & Elyan, E. (2014). Random forests: from early developments to recent advancements. *Systems Science & Control Engineering: An Open Access Journal*, *2*(1), 602-609.

Fischer, T. (2014). Private and public key cryptography and ransomware. *Center for Internet Security (CIS). December*.

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

Garg, D., Thakral, A., Nalwa, T., & Choudhury, T. (2018, June). A Past Examination and Future Expectation: Ransomware. In *2018 International Conference on Advances in Computing and Communication Engineering (ICACCE)* (pp. 243-247). IEEE.

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

Genç, Z. A. (2020). *Analysis, Detection, and Prevention of Cryptographic Ransomware* (Doctoral dissertation, University of Luxembourg, Luxembourg, Luxembourg).

Genç, Z. A., Lenzini, G., & Ryan, P. Y. (2018, June). No random, no ransom: a key to stop cryptographic ransomware. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (pp. 234-255). Springer, Cham.

Genç, Z. A., Lenzini, G., & Sgandurra, D. (2019, June). On deception-based protection against cryptographic ransomware. In *International conference on detection of intrusions and malware, and vulnerability assessment* (pp. 219-239). Springer, Cham.

Greenberg, A. (2018). The untold story of NotPetya, the most devastating cyberattack in history. *Wired, August*, *22*.

Greengard, S. (2021). The worsening state of ransomware. *Communications of the ACM*, *64*(4), 15-17.

Guarnieri, C., Tanasi, A., Bremer, J., & Schloesser, M. (2012). Automated Malware Analysis-Cuckoo Sandbox.

Homayoun, S., Dehghantanha, A., Ahmadzadeh, M., Hashemi, S., & Khayami, R. (2017). Know abnormal, find evil: frequent pattern mining for ransomware threat hunting and intelligence. *IEEE transactions on emerging topics in computing*, *8*(2), 341-351.

https://ke-cirt.go.ke/wp-content/uploads/2020/11/Press-Statement-Update-by-the-Communications-Authority-of-Kenya-on-the-Wannacrypt0r-Ransomware-Cyber-Attack.pdf

Humayun, M., Jhanjhi, N. Z., Alsayat, A., & Ponnusamy, V. (2021). Internet of things and ransomware: Evolution, mitigation and prevention. *Egyptian Informatics Journal*, *22*(1), 105-117.

Hwang, J., Kim, J., Lee, S., & Kim, K. (2020). Two-stage ransomware detection using dynamic analysis and machine learning techniques. *Wireless Personal Communications*, *112*(4), 2597-2609.

Karbalaie, F., Sami, A., & Ahmadi, M. (2012). Semantic malware detection by deploying graph mining. *International Journal of Computer Science Issues (IJCSI)*, *9*(1), 373.

Kardile, A. B. (2017). *Crypto ransomware analysis and detection using process monitor* (Doctoral dissertation).

Khammas, B. M. (2020). Ransomware Detection Using Random Forest Technique. *ICT Express*, *6*(4), 325-331.

Kharaz, A., Arshad, S., Mulliner, C., Robertson, W., & Kirda, E. (2016). {UNVEIL}: A large-scale, automated approach to detecting ransomware. In *25th {USENIX} Security Symposium ({USENIX} Security 16)* (pp. 757-772).

Kharraz, A., Robertson, W., Balzarotti, D., Bilge, L., & Kirda, E. (2015, July). Cutting the gordian knot: A look under the hood of ransomware attacks. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (pp. 3-24). Springer, Cham.

Kok, S. H., Azween, A., & Jhanjhi, N. Z. (2020). Evaluation metric for crypto-ransomware detection using machine learning. *Journal of Information Security and Applications*, *55*, 102646.

Kok, S., Abdullah, A., Jhanjhi, N., & Supramaniam, M. (2019). Ransomware, threat and detection techniques: A review. *International Journal of Computer Science and Network Security*, *19*(2), 136.

Kolodenker, E., Koch, W., Stringhini, G., & Egele, M. (2017, April). Paybreak: Defense against cryptographic ransomware. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security* (pp. 599-611).

Kotov, V., & Massacci, F. (2013, February). Anatomy of exploit kits. In *International symposium on engineering secure software and systems* (pp. 181-196). Springer, Berlin, Heidelberg.

Kotov, V., & Rajpal, M. S. (2014). Understanding crypto-ransomware. *Bromium whitepaper*.

Kumar, A., Kuppusamy, K. S., & Aghila, G. (2019). A learning model to detect maliciousness of portable executable using integrated feature set. *Journal of King Saud University-Computer and Information Sciences*, *31*(2), 252-265.

Lee, J., Lee, J., & Hong, J. (2017, September). How to make efficient decoy files for ransomware detection?. In *Proceedings of the International Conference on Research in Adaptive and Convergent Systems* (pp. 208-212).

Lin, C. H., Pao, H. K., & Liao, J. W. (2018). Efficient dynamic malware analysis using virtual time control mechanics. *Computers & Security*, *73*, 359-373.

McIntosh, T., Jang-Jaccard, J., Watters, P., & Susnjak, T. (2019, December). The inadequacy of entropy-based ransomware detection. In *International Conference on Neural Information Processing* (pp. 181-189). Springer, Cham.

Mehnaz, S., Mudgerikar, A., & Bertino, E. (2018, September). Rwguard: A real-time detection system against cryptographic ransomware. In *International Symposium on Research in Attacks, Intrusions, and Defenses* (pp. 114-136). Springer, Cham.

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

Naik, N., Jenkins, P., & Savage, N. (2019, October). A ransomware detection method using fuzzy hashing for mitigating the risk of occlusion of information systems. In *2019 International Symposium on Systems Engineering (ISSE)* (pp. 1-6). IEEE.

Nieuwenhuizen, D. (2017). A behavioural-based approach to ransomware detection. *Whitepaper. MWR Labs Whitepaper*.

Osterman Research, Inc, August 2016. Understanding the Depth of the Global Ransomware Problem. Tech. rep. Osterman Research, Inc., https://www.malwarebytes.com/pdf/white-papers/UnderstandingTheDepthOfRansomwareIntheUS.pdf.

Oz, H., Aris, A., Levi, A., & Uluagac, A. S. (2021). A Survey on Ransomware: Evolution, Taxonomy, and Defense Solutions. *arXiv preprint arXiv:2102.06249*.

Paquet-Clouston, M., Haslhofer, B., & Dupont, B. (2019). Ransomware payments in the bitcoin ecosystem. *Journal of Cybersecurity*, *5*(1), tyz003.

Pham, D. V., Halgamuge, M. N., Syed, A., & Mendis, P. (2010, July). Optimizing windows security features to block malware and hack tools on USB storage devices. In *Progress in electromagnetics research symposium*.

Poudyal, S., Subedi, K. P., & Dasgupta, D. (2018, November). A framework for analyzing ransomware using machine learning. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 1692-1699). IEEE.

Rhode, M. (2021). *Racing demons: Malware detection in early execution* (Doctoral dissertation, Cardiff University).

Rhode, M., Burnap, P., & Jones, K. (2018). Early-stage malware prediction using recurrent neural networks. *computers & security*, *77*, 578-594.

Rhode, M., Tuson, L., Burnap, P., & Jones, K. (2019, June). Lab to soc: robust features for dynamic malware detection. In *2019 49th annual IEEE/IFIP international conference on dependable systems and networks–industry track* (pp. 13-16). IEEE.

Rieck, K., Holz, T., Willems, C., Düssel, P., & Laskov, P. (2008, July). Learning and classification of malware behavior. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (pp. 108-125). Springer, Berlin, Heidelberg.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386.

Sahay, S. K., Sharma, A., & Rathore, H. (2020). Evolution of malware and its detection techniques. In *Information and Communication Technology for Sustainable Development* (pp. 139-150). Springer, Singapore.

Salton, G., & Buckley, C. (1990). Improving retrieval performance by relevance feedback. *Journal of the American society for information science*, *41*(4), 288-297.

Scaife, N., Carter, H., Traynor, P., & Butler, K. R. (2016, June). Cryptolock (and drop it): stopping ransomware attacks on user data. In *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)* (pp. 303-312). IEEE.

Scalas, M., Maiorca, D., Mercaldo, F., Visaggio, C. A., Martinelli, F., & Giacinto, G. (2019). On the effectiveness of system API-related information for Android ransomware detection. *Computers & Security*, *86*, 168-182.

Selvaraj, K., Florio, E., Lelli, A., & Ganacharya, T. (2017). Wannacrypt ransomware worm targets out-of-date systems. *Microsoft Technet*.

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*.

Shaukat, S. K., & Ribeiro, V. J. (2018, January). RansomWall: A layered defense system against cryptographic ransomware attacks using machine learning. In *2018 10th International Conference on Communication Systems & Networks (COMSNETS)* (pp. 356-363). IEEE.

Shukla, M., Mondal, S., & Lodha, S. (2016, October). Poster: Locally virtualized environment for mitigating ransomware threat. In *proceedings of the 2016 ACM SIGSAC conference on computer and communications security* (pp. 1784-1786).

Singh, K., Guntuku, S. C., Thakur, A., & Hota, C. (2014). Big data analytics framework for peer-to-peer botnet detection using random forests. *Information Sciences*, *278*, 488-497.

Sjouwerman, S. (2017). New Ransomware CryptoFortress Encrypts Unmapped Network Shares.

Sophos: Sophos state of ransomware 2021. Sophos Technical report (2021). https://secure2.sophos.com/en-us/medialibrary/pdfs/whitepaper/sophos-state-of-ransomware-2021-wp.pdf

Stiborek, J., Pevný, T., & Rehák, M. (2018). Probabilistic analysis of dynamic malware traces. *Computers & Security*, *74*, 221-239.

Subedi, K. P., Budhathoki, D. R., & Dasgupta, D. (2018, May). Forensic analysis of ransomware families using static and dynamic analysis. In *2018 IEEE Security and Privacy Workshops (SPW)* (pp. 180-185). IEEE.

Takeuchi, Y., Sakai, K., & Fukumoto, S. (2018, August). Detecting ransomware using support vector machines. In *Proceedings of the 47th International Conference on Parallel Processing Companion* (pp. 1-6).

Taylor, M. (2017). *Ransomware Detection Using Machine Learning and Physical Sensor Data*. Southern Methodist University.

Thakkar, S. (2014). Ransomware-exploring the electronic form of extortion. *Department of Computer Applications*, *2*.

Wainwright, R. (2016). Internet organised crime thread assessment (IOCTA). *Europol European Police Office, Hague, The Netherlands, Tech. Rep*.

Ward, M. (2014). Cryptolocker victims to get files back for free. *BBC News, August*, *6*.

Weckstén, M., Frick, J., Sjöström, A., & Järpe, E. (2016, October). A novel method for recovery from Crypto Ransomware infections. In *2016 2nd IEEE International Conference on Computer and Communications (ICCC)* (pp. 1354-1358). IEEE.

Young, A., & Yung, M. (2005, February). Malicious cryptography: Kleptographic aspects. In *Cryptographers' Track at the RSA Conference* (pp. 7-18). Springer, Berlin, Heidelberg.

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.

APPENDICES

Appendix 1: Research Schedule

The Gantt chart below shows the research study's schedule.

	Jan '21	**May '21**	**June '21**	**July '21**	**August '21**	**August '22**	**Sept '22**	**Oct '22**
Preparation of thesis proposal								
Concept paper Submission								
Research proposal development								
Proposal defence								
Research phase								
Data collection								
Dynamic Analysis								
Model development								
Model testing & validation								
Work in progress document defence								
Report writing								
Submission of thesis								
Final thesis defence								

Table A.1 : Research schedule

Appendix 1I: Resources and Budget

Resource	Unit/Capacity	Value/ cost ($)
Analysis computer (Laptop)	1	800
External Hard Drive	500 GB	60
Anti-virus software	1	50
Internet bundles	-	200
Printing & binding costs	-	150
Miscellaneous	-	100
Total	-	1360

Table B.2 : Resources and Budget