Intelligent Endpoint-based Ransomware Detection Framework

by

Okpongete Faith Douglas
B.Eng., Covenant University, Ota, Nigeria, 2017
A Report Submitted in Partial Fulfillment of the Requirements for the Degree of
MASTER OF ENGINEERING
in the Department of Electrical and Computer Engineering

Intelligent Endpoint-based Ransomware Detection Framework

by

Okpongete Faith Douglas
B.Eng., Covenant University, Ota, Nigeria, 2017

Supervisory Committee

Dr. Issa Traore, Supervisor

(Department of Electrical and Computer Engineering)

Dr. Tao Lu, Departmental Member

(Department of Electrical and Computer Engineering)

ABSTRACT

Over the past couple of decades, ransomware attacks have increased significantly and that calls for more aggressive efforts in building robust detection models to detect and reduce the impact of the attacks. Once attacked, the malware takes over the victims' machines and files by locking or encrypting them. These attacks have also led to huge global financial loss for people, businesses, and government of nations. The cybercriminals who perpetrate these attacks always demand for payment of some ransom in cryptocurrency. Presently, there are three common methods for detecting these ransomware attacks viz static, dynamic, and hybrid detections. Static detection is known to evade detection easily by cryptographic techniques and that is why the dynamic detection was adopted for this project. We trained and tested offline a detection model using the ISOT Ransomware dataset and implemented the proposed model as a standalone endpoint detector. The detector was deployed and evaluated online using new samples from the wild, whereby Cuckoo Sandbox was used to execute and extract the malware features during the experiment. The online evaluation confirmed the offline performance results, which were very encouraging.

Contents

List of Tables

List of Figures

Acknowledgement

I am thankful to God, as well as my parent and siblings. Thank you to Dr. Issa Traore for his support and guidance throughout the program. To the University of Victoria for making me career ready and a platform to grow.

I would like to thank my committee members, Dr Amirali Baniasadi and Dr Tao Lu for all their valuable advice and critical feedback.

To Emem Okpongete, Thank you for supporting and accommodating me when I first came to Canada.

I would like to also extend my gratitude to my friends, Ola Olayokun, Onyekachi Nwamuo, Janwari Adnan, Isung Nsikak, and Chibuike Onuigwe for their guidance and support throughout my master's program as it was their valuable insight and perspective that kept me going in the program.

Dedication

I dedicate this report to my late father.

Chapter 1: Introduction

1.1 Context

The broad use of technology is increasing, along with the threats to current digital devices. Some of the malicious programs which are harmful to the current digital systems utilized in today's world include but are not limited to botnets, trojan horses, ransomware, viruses, and worms. Research has shown that ransomware is of utmost concern hence, the focus of this project.

In recent times, organizations have witnessed an increase in ransomware attacks as cybercriminals constantly innovate by creating new attack skills and methods. Cloud adoptions and delivering a complex mix of web applications have helped to fuel these cyber attacks. Ransomware is among the latest in the evolution of advanced malware and exploits. As a result, companies are closely examining existing infrastructures to ensure these vulnerabilities are not becoming a threat to their daily operations. In addition to this, companies are also working to build more resilient cyber defence mechanisms to protect themselves against these cyberattacks.

Since 2016, the United States has witnessed over 4000 ransomware attacks [1]. Healthcare and financial service industries lead the charts as the top targets for these cybercriminals. The first ransomware attack was carried out in 1989 by Joseph Popp. During the attack, Joseph distributed 20,000 floppy discs, which kept track of the frequency at which the infected machines rebooted. This ransomware conceals the directories and encrypts all the files in the root directory [2] when the reboot count reaches 90. In September 2013, there was a Crypto locker outbreak attack that affected more than 250,000 computer systems, and the victims were made to pay in cryptocurrencies to regain control of their devices [2]. In May 2017, there was also a ransomware attack called WannaCry, which exploited a weakness in the windows operating system thereby affecting hundreds of thousands of computers worldwide, resulting in roughly 4 billion dollars loss globally. The WannaCry vulnerability came to light as part of the National Security Agency (NSA) leaked documents. Till today, organizations are still falling victims to these WannaCry attacks. Three North Korean computer programmers were indicted by the US Department of Justice in 2021 for their alleged role in the WannaCry global attack.

The huge financial and reputational losses caused by these ransomware attacks serve as a wake-up call for organizations all around the world to intensify efforts to forestall further occurrences of these attacks. Undoubtedly, these attacks have costed users a lot of productivity hours, money, and business discontinuity over the past several decades. In another recent attack that happened in April 2021, cybercriminals gained unauthorized access to the Colonial Pipeline Company's network via a virtual private network account [1] allowing them to log into the compromised systems without requiring multi-factor authentication. The pipeline company had to pay a ransom of about 75 bitcoins equivalent to US$5M at the time of the attack [1].

It is no longer news that ransomware has been around for quite some time and in constantly increasing samples with the capacity to bypass detection, and encryption of files, and cause affected users to pay a high ransom. Ransomware falls into two categories namely crypto-ransomware and locker-ransomware [3]. The crypto ransomware encrypts files and denies legitimate users access to their files while the locker ransomware locks users out of their systems thereby preventing them from usage.

This project will contribute to building an endpoint detector to run samples on the trained model to optimally classify the samples as either yes or no. After the experimentation, we can say that these features of ransomware play a key role in building a good model which produced an accuracy of 95% with newly tested samples. Hence, we have been able to add to the cybersecurity collection for ransomware detection and analysis.

1.2 Project Objective

Different security teams have different approaches to ransomware analysis and detection but it all falls under the same general category. The various methods for malware analysis include static analysis, reverse engineering, and dynamic analysis [4].

Ransomware detection based on static analysis examines the source code for the presence of any malicious code that could give cybercriminals unauthorized access to the user data [4]. Any malicious routine flagged in the binary code will be detected by software such as antivirus and prevented from running. This is beneficial to cybersecurity experts because it provides information

about its functionality and provides tips that will help to generate simple network signatures. The signature-based analysis is the most common type of static analysis where specific code patterns are extracted from the application and compared against a database of known malicious signatures. As new malware evolves, the database is updated making signature-based detection the fastest approach to malware detection. However, it is largely ineffective against zero-day attacks, also called novel malware. Cybercriminals are continually developing sophisticated code to ensure every version is different making it hard for the static analysis to detect.

In recognition of the limitations of the static-based analysis, a reverse engineering of malware binary to assembly code can be used to help better understand the malicious programs and their design [4]. In reverse engineering of the malware, the source code is decompiled into code mnemonics which helps the security analyst/engineer to know what the program does and how it impacts the system. This approach helps to engineer solutions to mitigate the impact on the affected systems and to know which vulnerabilities the program intends to exploit. One of the success stories of the reverse engineering approach was the discovery of the WannaCry kill switch, which has helped curb WannaCry's spread.

Compared to the aforementioned two approaches to ransomware detection, the dynamic analysis uses a sandbox to run malware in a protected environment to monitor its behaviour. The sandbox ensures that the malware is confined and segmented to not compromise the design and further escape into the organization's network. Previous work on dynamic analysis for ransomware detection systems focused on training machine learning models using the different ransomware features [5] extracted from the samples such as API calls, embedded strings, mutex, windows DLLs, etc. In this project, unique features of the binary samples such as the API calls, embedded strings, and entropy were used to train the machine learning model.

This project builds on an earlier work conducted in the ISOT lab by Brijesh et al. [13] in which, unique features of the binary samples such as the API calls, embedded strings, and entropy were used to train, test, and compare different machine learning models. The purpose of the current project was twofold. First, to revisit and replicate the results obtained for the different classifiers explored in this earlier study. Second, to implement and evaluate in a sandbox environment an endpoint detector based on the best detection model.

Three major tasks were carried out in the project. The first task was to select the machine learning algorithm with the highest accuracy for binary classification by using the already processed ISOT dataset and taking into the work carried by Brijesh et al [13]. Three classification techniques were studied including logistic regression, support vector machine, and random forest. The logistic regression algorithm recorded the highest accuracy and hence its selection. This confirmed the outcome of the earlier study conducted by Brijesh et al [13]. The second task was to build a model with the selected machine algorithm having the best accuracy. The final task was to validate the model with new ransomware sample binaries (in addition to the existing dataset) and display the result in a client/server endpoint detector via a Graphical User Interface.

1.3 Report Outline

The report structure is as follows.

Chapter 1 outlines the context of the project, the history of ransomware attacks, the project objective, and a summary of the project.

Chapter 2 provides background information on ransomware, an overview of the detection model, and the ISOT ransomware dataset.

Chapter 3 presents the endpoint-based detection framework, the architectural design, and different machine learning algorithms and describes the GUI.

Chapter 4 presents the experiments conducted to evaluate the performance and the accuracy of the proposed detection scheme.

Chapter 5 gives the summary and recommendations for future work.

Chapter 2: Background

2.1 Background on Ransomware

Understanding ransomware behaviour is fundamental in building a client-server endpoint for detection. This chapter will not only introduce the basic concepts and theories on ransomware but also, the methods and techniques used in the detection of ransomware.

Ransomware involves the use of some technical and social engineering skills. Social engineering involves psychological manipulation, tricking users into making mistakes or providing sensitive information.

Also, ransomware being persistent in nature operates by adding a registry entry and copying itself to an operating system thereby interfering with the start-up process. Ransomware injects itself into a legitimate process and executes from the application data directory using a typical Windows executable name, making it hard to be detected by application security software.

In most situations, the attacker studies the victim, and the websites they visit frequently and sends an email to the user to click, but in a manipulating way. Most times the attacker carries out a form of attack called a clone phishing attack, which happens when a legitimate website or application is duplicated to fool the victim into thinking he is filling out a legitimate form on the website [6]. To avoid suspicion, the victim is forwarded to the legitimate page after the attack. The user is the vulnerability exploited in those attacks. When a user clicks the infected link, a payload is downloaded without the user's knowledge in the backend, and the malware begins its execution. And during the execution process, the ransomware hides its identity within a dropper code. The malicious code can be confined within a single stage to prevent detection by virus scanners or can be downloaded to the user system once it has been triggered (two stages).

One example of this staged ransomware operation is through a Windows operating system. Ransomware can hide in legitimate Windows operating processes such as svchost.exe and creates a fake svchost process in the backend while the user's computer boots up. While this happens in the backend, the ransomware reboots itself and executes in a safe mode. This operation creates registry keys in Windows and adds them to start-up processes to confirm their existence. One svchost may be responsible for network services while another can be responsible for remote

procedure calls. However, cybercriminals frequently hide harmful files by giving them similar names to standard programs. Figure 2.1 depicts a typical ransomware attack scenario.

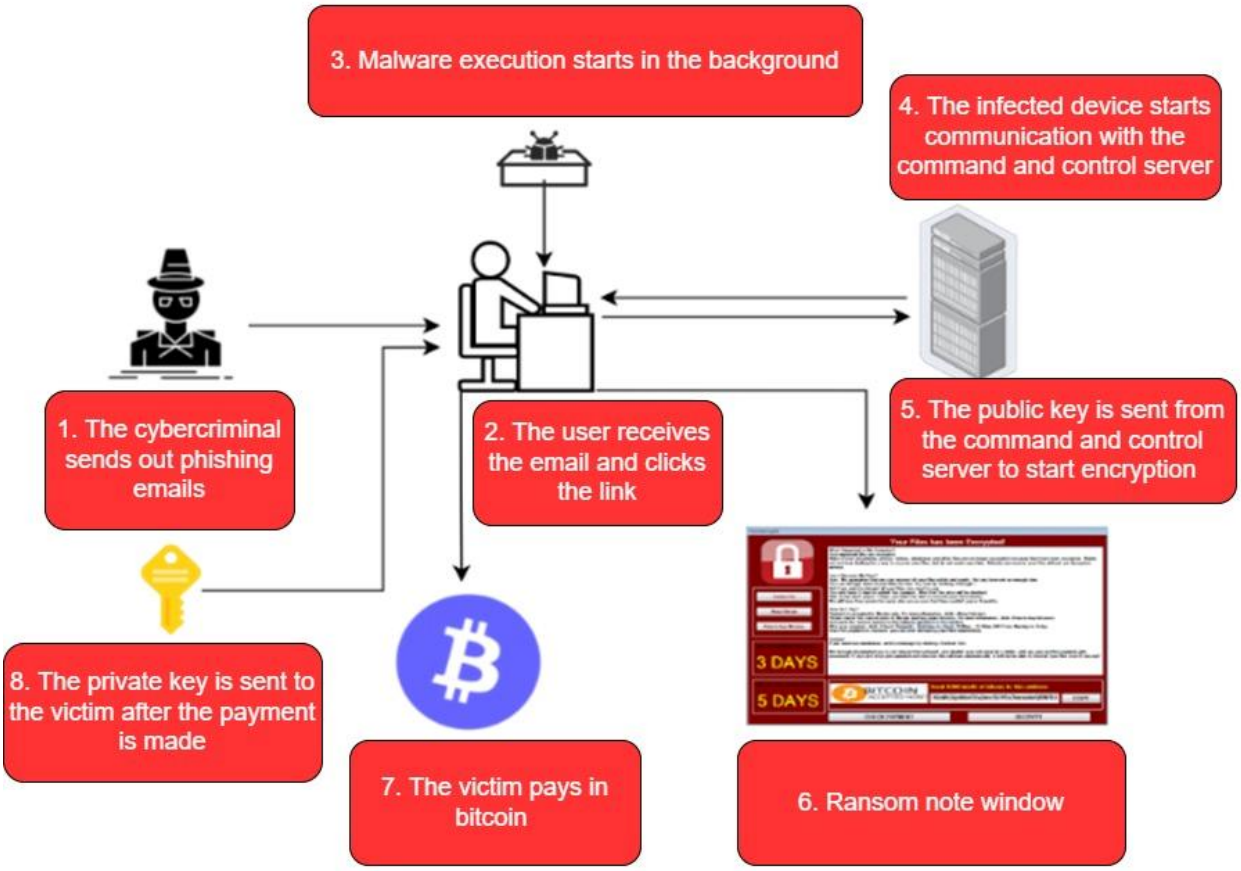

Figure 2.1 Ransomware Attack Scenario

As seen in fig 2.1, the attack scenario starts when the cybercriminal sends out phishing emails to the supposed victim. When the victim receives the email and clicks the link, the ransomware then makes its way to being deployed in the background of the victim's system after which a communication link will be established for communication with the command and control (C&C) server. The repercussions of this communication lead to the generation of pairs of encryption keys

(public and private key). The public key is sent from the C&C server to start the encryption process of the victim's device while the attacker withholds the private key which will be used for the decryption process once the victim pays the ransom amount. The communication with the C&C server is secured by a TOR browser which was downloaded and installed during the ransomware deployment, due to its anonymous nature.

Once the communication is successful, the ransomware is now live and ready to exploit the victim's resources. The malware code will begin encrypting all the files that the C&C programs have identified. The files can be in any format as seen in table 2.1 and the malicious code that carries out this encryption is frequently written in scripts or batch files to evade being recognized by signature scanners. Moreover, during this attack, anti-malware software installed on the victim's machines is disabled thereby rendering them ineffective in protecting the victim's machine during that period.

After the deployment of the ransomware and encryption of all the files, a ransom note is generated for the user as depicted in figure 2.2. The ransom note contains sets of instructions required for the user to regain control of the encrypted files and the information on the payment (currency) which is popularly known to be in bitcoin to regain control of the encrypted assets. During the extortion, the cybercriminal utilizes a variety of tactics to make the victim pay. A key can be provided for a victim to decrypt one file for demonstration or incremental fees detailing some amount you must pay before the key is erased by the server over time. After you pay the ransom, you will get the key to decrypt your files but not always guaranteed. Furthermore, there is no assurance that the system is ransomware free.

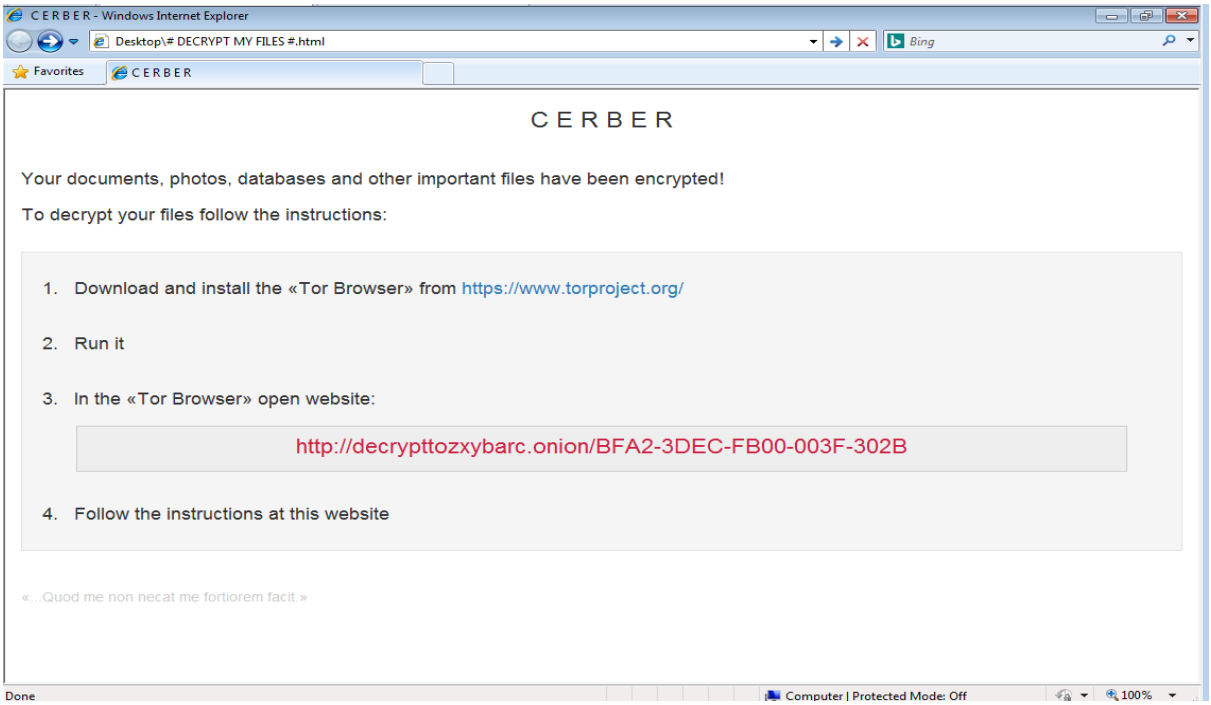

Figure 2.2 Cerber Ransom Note

2.2 Ransomware Detection Approaches

The detection of ransomware attacks relies heavily on ransomware analysis. As mentioned earlier, it is well-known that ransomware attacks occur on a wide range of devices. Ransomware analysis aids in deciphering a program's sequence of actions, determining its behaviour, and identifying the execution program.

Some researchers have elaborated on a different approach to consider and the discussions in this section will be centred around the growing amount of literature already published on ransomware detection techniques with an emphasis on static, dynamic, and hybrid methods. Some of the findings on these methods will be outlined and appraised.

2.2.1 Static Analysis

Static analysis detection method operates by matching or comparing the malicious code characteristics to existing features in a repository without executing the code.

Jerome and colleagues [7] retrieved program operational codes from malware binaries and converted them to an operational codes sequence. The analysis revealed some intriguing malware signature trends that helped the classifier improve its false positive and false negative rates. The authors selected valuable features from the information gained and utilized the support vector machine algorithm to classify them. The result of the experiment produced a true positive rate (TPR) of 81.40% and a false positive rate (FPR) of 2.67% [7].

Schultz and colleagues were among the first researchers credited with the using a static approach to detect ransomware [8]. A classification system was used by these researchers to classify the virus utilizing portable executable string information features. In a similar manner, Kolter et al [9] classified malware binaries using n-gram byte sequences and made use of various supervised machine learning algorithms such as decision trees, and SG boost, in the experiment. The result of their experiment shows that the boosted decision tree algorithm outperformed the others by having a true positive rate (TPR) of 98% and a false positive rate (FPR) of 5% [9].

Typically, classifications depending on static analysis are not able to detect new malware strains. In a study of malware strain, the detection rate for new ransomware utilizing static analysis examined ten out of sixty strains [10]. This weakness was validated by Baig et al [36] as they were able to avoid static detection by changing packed portable executables. In addition, some programs can be written to escape detection from static analysis. More so, ransomware detection can evade static analysis if the attackers modify the actual programs and metadata, the process called code obfuscation.

The static analysis only gives a few details and as a result, dynamic analysis has been taken into consideration. Researchers have conducted some studies on dynamic ransomware techniques to aid ransomware detection.

2.2.2 Dynamic Analysis

Dynamic analysis involves running malicious programs in a closed or controlled environment. The controlled environment records the displayed features of ransomware and how it can be executed. To improve the outcome of dynamic analysis, machine learning and deep learning algorithms can be applied to build a detection model [11].

In the past, researchers have carried out ransomware detection using machine learning techniques and process mining technology. Bahrani, A. and Bidgly, A.J., used Disco, a fuzzy algorithm-based tool, in mining features [12]. After each sample runs on a virtual machine during the analysis some event logs were generated keeping track of the registry and the file system activities. This study demonstrated a synergic combination of machine and deep learning technologies which could produce a more robust and accurate ransomware detection system [12]. Another contribution of their research was to provide a user manual that can help other researchers work with various technologies in ransomware detection while considering existing solutions. However, their study did not go into much detail about the dynamic analysis approach.

Jethva et al. used machine learning algorithms such as a support vector machine, random forest, and logistic regression to build a ransomware detection model. Their model was built by running ransomware binaries in a cuckoo sandbox to extract their behavioural features. Their experiment produced a true positive rate (TPR) of 98.7% and a false-positive rate of 1.41% [13]. The resulting model can accurately detect high survivability ransomware at an early stage. A forward-thinking monitoring system explored and compared the feature space with machine learning techniques. On the Windows system, this solution also offered file backup. One of the objectives of this project is to build a ransomware detection model using the current ISOT ransomware dataset and to explore more compelling ransomware features.

In another research carried out by Sayan Sinha et al, the authors developed a detection scheme called ransomware prevention via performance counters (RAPPER) that uses Artificial Neural Network and Fast Fourier Transformation [14]. Their solution was fast and successful in identifying disk encryption processes from potential ransomware operations. Their solution also provided insight into tackling most crypto-ransomware attacks.

Shield FS, developed by Continella et al [16] involved carrying out a study on an automated ransomware detection system that has the potential of rolling back malicious changes. The solution-focused on analyzing the entropy of read/write operations to internally monitor low-level file system operations. The study also looked out for block cipher key schedules in the memory areas of any process judged "possibly dangerous." In one peripheral driver, the technology combines automatic detection and easy file recovery. One of the drawbacks of this study is that file recovery is nearly impossible because novel ransomware attacks operate by encrypting or deleting all the Windows Shadow copies of file systems. Another drawback is that the study was only for file-related features and operations.

CryptoDrop [17] is, another ransomware detection system developed to alert users when questionable file activity was detected to have taken place. This system is designed to track changes in user data. Depending on the encryption of the user files, the researchers categorized ransomware into three categories viz class A, class B, and C. Similarity functions were adopted to determine the level of dissimilarity between the original and the content of the file which has been encrypted by the ransomware. However, the CryptoDrop system could not explain the root cause of the changes as logged in the audit report. As an example, the system was unable to show the difference between ransomware-triggered encryption and normal user-triggered encryption.

In another study, EldeRan [18], a machine learning approach for analyzing and detecting ransomware was proposed by Daniele and colleagues. EldeRan approach was in phases starting with monitoring a collection of application activities and the identification of the characteristics exhibited by the ransomware. The second phase involves making use of features like API calls, registry keys, dropped files, directory enumerations, and registry keys on a regularized machine learning model to discover patterns that can distinguish ransomware from innocuous programmes. To evaluate the model, a dataset having 582 ransomwares from 11 different families was used. Using dynamic analysis on a small number of characteristics, produced an accuracy of 96.3%. Nevertheless, the EldeRan solution was unable to extract the attributes of ransomware that remained silent for a tangible amount of time. More so, the solution was based on the binaries of these samples with emphasis on the availability or unavailability of features like registry key operations, dll operations, mutex, etc. This was also another drawback of the solution as it will not effectively detect new malware strains due to the lack of these specific processes [18].

As seen in the literature, the dynamic analysis technique has proven to be a better approach for ransomware detection and hence the focus of this project.

2.2.3 Hybrid Analysis

The hybrid analysis is a combination of static and dynamic analysis to deliver more accurate and acceptable results. Machine learning and deep learning models are vital in the hybrid analysis and have already been used in many ransomwares' detection studies till date.

Hasan and Rahman [20] carried out research on ransomware detection using hybrid analysis and titled their work "RansHunt". Their framework which was evaluated using a dataset with 1,283 different ransomware binaries returned a 97.1 percent accuracy [20]. The dataset contained 923 benign and 360 ransomware binaries from 21 different families [20]. New network-related features were included in the dynamic analysis part but there was no significant impact on the detection rate. The model was not successful in detecting new ransomware variants even though the features were similar to EldeRan [18].

In another study, Kashif and Riberio [21], presented a tiered defence mechanism for crypto-ransomware security. They created a hybrid system by combining static and dynamic analyses. The dynamic detection layer keeps track of file system operations and entropy changes linked to large-scale encryption. Files which have been altered by these ransomware processes were backed up in a safe location to offer temporary protection pending when they are evaluated and validated as either ransomware or benign. They evaluated their solution with a dataset consisting of 574 ransomware samples from 12 distinct ransomware families and achieved an accuracy of 98.25 percent. Dynamic analysis, like other systems, is heavily reliant on API calls and file system operations and this makes it difficult to detect ransomware programmes that use proprietary routines instead of standard Windows APIs.

Another research offered a pre-encryption ransomware detection method that used a hybrid analysis approach [22]. Two analyses utilize the system, which worked in two stages. The static analysis supported pre-execution in the first step, whereas dynamic analysis supported pre-encryption in the second stage. This research also developed a signature repository for evaluating crypto ransomware.

2.3 ISOT Ransomware Dataset Overview

A good reference dataset is needed for the development of a practical machine learning detection model to deal with most network security issues, including malware attacks, phishing, etc. The ISOT ransomware dataset was collected from the Information Security and object technology (ISOT) lab, and it is publicly available to the research community [13].

The ISOT ransomware dataset contains data from 669 ransomware programs from well-known ransomware families and 103 benign programs from frequently used windows software applications. The dataset is about 428 GB in size. Table 2.1 contains the family names and numbers of ransomware variants in the ISOT ransomware dataset.

In the ISOT ransomware dataset experiment setup, the Cuckoo sandbox was used to execute the ransomware and the benign samples [23]. Cuckoo was installed on a host machine running Ubuntu 14.04 and the analysis machine which is also the guest was running Windows 7 64 bit. Cuckoo sandbox requires a host and an analysis machine to function. The host is configured to use a static IP address of 192.168.50.11 and a host-only adapter network layout.

Table 2.1 Ransomware Families in ISOT Dataset

Family Name	Sample (Variant) Count
Mole	4
Jaff	3
Teslacrypt	348
Locky	129
Zeta	2
Satan	2
Cerber	122
Petya	2
Win32.Blocker	18
Crysis	8
Xorist	2
Sage	5
Flawed	1
Spora	5
Striked	1
CryptoShield	4
WannaCry	1
GlobeImposter	4
Unlock26	3
Total	669

During the analysis of the sample, the behavioural data is stored in different directories on the host machine as seen in figure 2.3 which contains Cuckoo analysis directory structure [23].

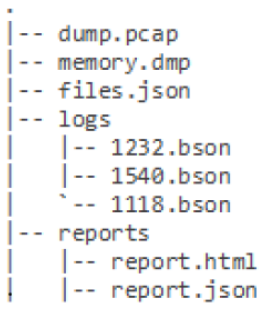

Figure 2.3 Cuckoo Analysis Directory Structure

The content of the sample directory is explained below.

dump.pcap

This directory contains all the traffic of the network dump generated during each analysis performed on the ransomware and benign samples.

memory.dmp

This directory contains the information of the memory dump of the analysis conducted.

file.json

This includes all the metadata information of the analysis in the form of JSON encoded format.

logs

The reported logs are stored in this directory with the extension.bson.

reports

All the analyses are stored in the reports folder using JSON format. It includes information about the analysis and its duration during the analysis. All the operations are stored in the top-level structure as seen in figure 2.4

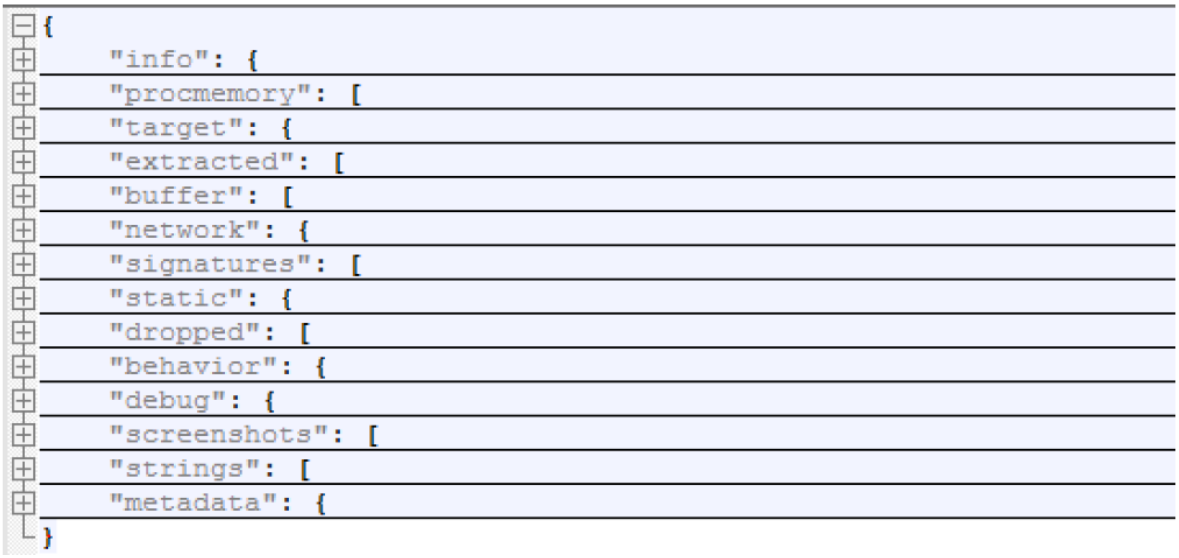

Figure 2.4 JSON Report Structure

Chapter 3: Endpoint-based Detection Framework

In the face of an ever-increasing threat landscape on the internet, effective cybersecurity measures are crucial to the success of digital transformation efforts in addition to keeping company and client information safe. The endpoint detection framework is an endpoint security solution that integrates client and server features to identify malicious software from a network's devices [24]. At the core of our endpoint detection framework, we are using a machine learning model to classify between ransomware and benign.

3.1 Architecture Design

An end-to-end detection framework solution consists of the following steps:

- Data sourcing
- Feature Engineering
- Model Development
- ML Deployment

Our proposed architectural design for the detection system is made up of 4 basic building blocks to predict between ransomware and benign samples as shown in figure 3.1.

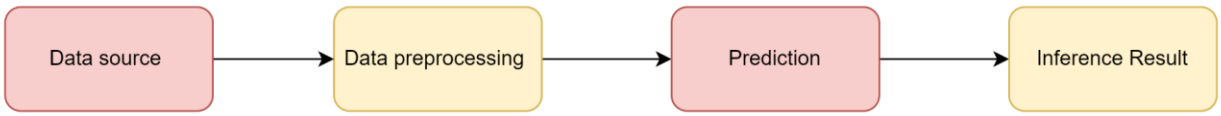

Figure 3.1 Architecture Design

- Box1 – Data Source/Ingestion: The API accepts data in JSON format.

- Box 2 – Data processing: The data collected is passed through the preprocessing pipeline where feature engineering steps such as scaling numerical features, encoding categorical variables etc. are applied. This transforms the data into the format to be fed into a machine learning algorithm and the output is saved as a pickle file.

- Box 3 - Prediction: In this step, we load the saved trained model weights from a pickle file, data from the preprocessing step is then fed into the model which then generates the inference results.

- Box 4 - Inference result: The model result (either Ransomware (1) or Benign (0)) is then processed and returned as a JSON payload.

3.2 Detector Implementation

The detector implementation consists of two stages viz model development and the backend server-side code.

Model development - (Data Science part):

- Data source/ingestion
- Feature Engineering - data preprocessing
- Model selection and training
- Model evaluation (Performance Metrics)

Backend server-side code - (Software Engineering part):

- API development

3.3 Model Development

3.3.1 Model Evaluation

It is always a good practice to evaluate a machine learning model using some performance metrics to validate how good the model performed with the new samples. To evaluate the performance of the trained model, we used a confusion matrix.

A confusion matrix is a table-like representation which gives you a better idea of what your classification model is getting right and what types of error it is making. Each cell in table 3.1 is an evaluation factor of the true positive, false positive, true negative, and false negative. Table 3.1 is a representation of the confusion matrix.

Table 3.1 Representation of a two-class confusion matrix

		Reference	
		Attack	Normal
Prediction	Benign	TN	FP
	Malware	FN	TP

True negative (TN) is when a benign program is correctly predicted.

False-positive (FP) is the incorrectly predicted event values such that, the benign program has been misclassified as a ransomware.

False negative (FN) is when ransomware has been categorized as benign.

True positive (TP) is the correctly predicted occurrence of ransomware program.

The second model evaluation we applied is the classification report. A classification report is a measure of the quality of the predictions gotten from the selected classification algorithm. The classification report consists of the precision, recall, f1 score, support row and accuracy, macro average and weighted average on the columns. The next couple of paragraphs gives a little bit of the rows and columns of this report.

Precision – Precision is the ratio of true positives to the total of false and true positives, where true positive (TP) is the number of attacks classified correctly and false positive (FP) is the number of incorrect classifications of benign as an attack.

$$\text{Precision} = 100\ \text{X}\frac{(\text{TP})}{(\text{TP}\ +\ \text{FP}\ )}$$

Recall - Recall is the ratio of true positive to the sum of a false-negative and true positive, where false negative (FN) is the number of incorrect classifications of an attack as benign.

$$\text{Recall} = 100\ \text{X}\frac{(\text{TP})}{(\text{TP}\ +\ \text{FN}\ )}$$

F1-score – F1-score is the weighted harmonic mean of the recall and precision. The weighted harmonic mean computes the average of precision and recall. It is calculated to provide a single metric that accurately weighs the precision-to-recall ratio. It is important to note that very low precision or recall will result in lower overall scores, hence aids in balancing the two key metrics. Also, F1 score will balance the metrics of both the positive and negative samples if you choose your positive class as the one with the fewest samples.

$$\text{F1 score}\ = 2\ \text{X}\frac{(\text{Precision} * \text{Recall})}{(\text{Precision}\ +\ \text{Recall}\ )}$$

Support – Support is the number of actual occurrences of the class.

Accuracy is the number of correct classifications either malware or benign out of all instances in the dataset.

$$\text{Accuracy} = 100\ \text{X}\frac{(\text{TP} + \text{TN})}{(\text{TP} + \text{TN} + \text{FP} + \text{FN})}$$

True Positive Rate (TPR) – True Positive Rate is the probability that an actual positive will test positive.

$$\text{TPR} = 100\ \text{X}\frac{(\text{TP})}{(\text{TP} + \text{FN})}$$

False Positive Rate (FPR) – False Positive Rate is the probability that a false alarm will be triggered because a positive result is given to negative result.

$$\text{FPR} = 100\ \text{X}\frac{(\text{FP})}{(\text{FP} + \text{TN})}$$

3.3.2 Data Source

The dataset used for the project was gotten from the University of Victoria ISOT lab. The format of the training data is a JSON dump extracted from the dynamic analysis conducted on a Cuckoo sandbox. Python panda's library was used for the loading of the data into a table-like structure called data frame.

Some of the features extracted included but not limited to the command line operations, the windows dynamic link library, embedded strings, registry key operations, file entropy, file operations and API calls. The next couple of paragraphs gives a few explanations of the features gotten from the binary samples.

Command-line operations: The command prompt is a line interpreter standard in every Windows operating system. It comes as a text interface for your computer system which you can use to navigate through different files and folders in your computer system through a command-line interface. A cybercriminal can leverage the backend of the command-line interface to delete write and run scripts that will automatically delete windows shadow copy and the master boot record.

Windows dynamic link library: Windows DLLs (Dynamic Link Library files) help to promote code reusability and the efficiency of memory usage. They contain resources for an application to run successfully. They also include images and libraries of executable functions that are not typically accessed by end-users but only the application when it starts up. DLL attacks mainly occur in Microsoft operating systems when the victim loads an infected DLL file in the same directory as the targeted application. Once a contaminated DLL file loads up during start-up, cybercriminals will be able to access the infected machine anytime it boots up.

Embedded strings: Typically, embedded strings are Unicode and ASCII character sequences embedded in binaries. Embedded strings from malware binaries can provide clue on a program's functionality. For example, a string can contain a reference to the filenames or domain names. Although strings may not provide a complete picture of a file's purpose and capabilities, they can provide insight into what malware can do.

Registry key operations: The registry key is a hierarchical database in a Windows operating system to manage system configurations and settings. Performing a malware attack using a registry key starts when we boot up our P.C. by modifying the parameters of the services. A wrong permission configuration can lead to changing registry keys. This allows the malware to launch at Windows start-up and run under a local system account with elevated privileges. Notwithstanding, this remains an excellent spot for malicious software to attack the endpoints.

File entropy: File entropy measures the degree of randomness in the file. The lower the entropy in a file, the less likely it is encrypted or compressed, but the higher the entropy, the more likely the file is encrypted. On the other hand, cybercriminals can use file signatures by corrupting the bytes, rendering the file useless, knowing they are essential to open the files.

Directory Enumerated: The DirectoryInfo and FileSystemInfo classes help search for and return various properties of directories and files. We have some directory enumerations converted to

numeric features in each binary execution in our given dataset, the ISOT ransomware dataset. Cybercriminals use brute force attacks to search the domain directory for unlinked contents such as temporary directories, files, obsolete backup, and configuration files. Intruders may consider these resources necessary since they store sensitive information about online applications and operational systems, such as source code, credentials etc.

API calls: Amongst the identified features, ransomware also exhibits random behaviour such as stealing browser information and changing booting sequences. Packer entropy, BCDEdit and deletion of the windows shadow copy are among these behaviours.

3.3.3 Feature Engineering - Data Preprocessing

In this project we applied data preprocessing steps such as data standardization and data encoding to prepare the data for the machine learning model.

Data standardization is one of the crucial steps in data preprocessing because it helps to ensure that all the features are on the same scale to be used by the model. This is one of the requirements for a machine learning model to perform well, also standardization transforms features by subtracting from the mean and dividing by standard deviation. Two of the notable data transformation method are min-max scaling and standardization [25]. Data standardization does not get affected by outliers because there is no predefined range of transformed features compared to min-max scaling. Also, data standardization does not bound to a certain range,as it translates the data to the mean vector of the original data to the origin and expands accordingly when exploring features. Compared to the min-max scaling method, which is affected by the presence of outliers, they are often used when the features are of very different scales and when we do not know about the distribution of the features. More information on data standardization can be found in Dhairya Kumar's work on the introduction to data preprocessing in machine learning [25].

However, because not all features in a dataset will be used in developing our machine learning model, we applied feature selection to select the most significant features. Adding more features to a model enhances the model's prediction. As a result, feature selection becomes an essential part of developing machine learning models. Feature selection consists of three different methods namely, filter, embedded and wrapper methods. The adopted method, filter, and apply statical measures to the features in the dataset and assign scores to each feature following their target class.

Filter selection takes note of the important feature by adding high score values and low values to the least important features. We adopted the filter method for the feature selection because it is fast and inexpensive [26]. Unlike the other two methods, wrapper methods discard unimportant features by using the recursive feature elimination method after comparing all the different features and embedded methods discover the right features while the model is being built and developed.

The features of a feature space influence the approach choice, some feature selection method for computing resources include principal component analysis and autoencoder when there is a need for dimensionality reduction in machine learning. Dimensionality reduction is when the number of features is not proportional to the number of data records, leading to overfitting and poor generalization. Having a dimensionality issue can include long training periods as well. Principal component analysis when compared to autoencoder is fundamentally a linear transformation process, unlike the latter which is a non-linear process. In addition, there should be a low-dimensional structure in the feature space if one needs to apply dimensionality reduction [26].

While conducting data preprocessing on the dataset, the Chi-square test was used to test the relationship between the features. The variables must be categorical, sampled independently, and the predicted frequency of the values should be more than 5 [27]. In general, a high chi-square value means that the feature is more dependent on the response and will be a candidate for selection by the training model.

3.3.4 Model Selection and Training

Model selection was applied to identify the model that will produce the lowest generalization error and the best performing algorithm. For this project, three different machine learning algorithms were used in developing the model and they include Logistic Regression, support vector machine (SVM), and random Forest.

With the help of the Chi-square feature selection, features ranging from 50 to 500 were explored to build the training model. The next step was to train the different classifiers mentioned above to determine the best model.

The next couple of paragraphs gives a little bit of background on the three supervised classification algorithm used in the preliminary model development.

Logistic regression (LR) is a classification algorithm that uses supervised learning to estimate the likelihood of a target variable of two classes [28]. Logistic regression is reasonable to estimate the best fitting model able to predict whether a payload is ransomware or benign.

SVM's primary purpose is to divide datasets into classes to find a maximum marginal hyperplane in two steps. First, SVM will iteratively construct hyperplanes that best separate the classes. Then, the hyperplane that correctly separates the classes will be selected [28]. SVM involves a unique implementation method which can handle both continuous and categorical variables.

Random forest (RF) is a type of machine learning algorithm commonly used to solve classification problems. The random forest algorithm builds decision trees from data samples, extracts predictions, and votes on the best prediction [28]. The random forest algorithm can be seen as ensemble-based learning because it aggregates many decision trees thereby reducing the variance as compared to individual trees. Each tree will categorize a new object based on attributes, and the class with the most votes will be chosen [28].

Based on a previous work conducted using 50 to 500 features in training a classifier for the detection of ransomware, Jethva et al. [13] chose three different machine learning classifiers namely logistic regression, support vector machine, and random forest to train the machine learning model. In their experiment, the support vector machine had an average accuracy of 94% while the logistic regression, and random forest accuracy ranged from 94% to 97%. During the model training, the authors observed, as depicted by Figure 3.2, that as the number of features increases, the accuracy also increases as was the case for the logistic regression, and random forest. In effect this also means that increase in the number of features could lead to a high probability of overfitting. Also worthy of note is that logistic regression, and random forest are easy to work with compared to the support vector machine. Logistic regression on the other hand is less prone to over fitting when trained with model regularization compared to random forest. Hence, the logistic regression was selected by the authors as the best classifier to train the model. In a like manner, logistic regression was also adopted to train the classifier for our endpoint detector.

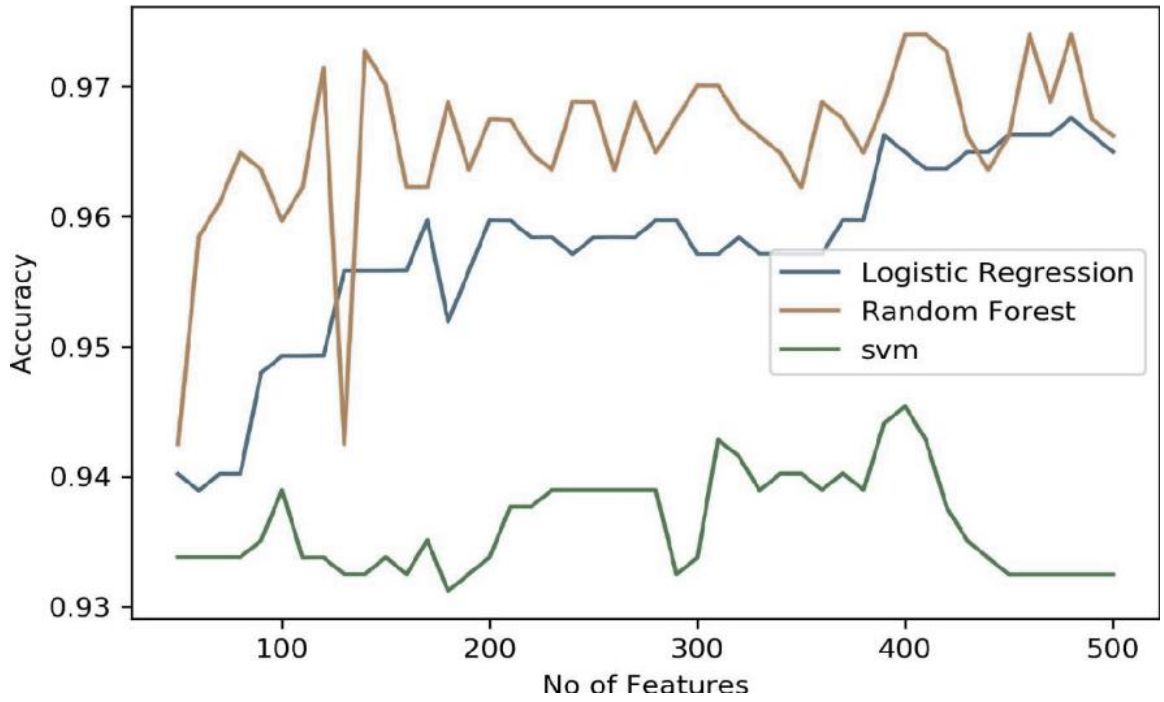

Figure 3.2 Accuracy when Training the Number of Features [13]

3.3.5 Trained Model and Evaluation

One of the crucial steps in building any machine learning model is the splitting of the dataset into a training set and a test set. While the training set is used for training the model, the test set is used to validate the model so it will be able to predict any unknown dataset. This splitting is usually done in a certain ratio, and in this project, we randomly allotted 75% of the dataset to the training set, keeping the remaining 25% for the test set.

During the testing of the model on the ISOT ransomware dataset, the logistic regression achieved a true positive rate (TPR) of 100% and false positive rate of 5% (FPR). Figure 3.3 and Table 3.2 show, respectively, the confusion matrix and the classification report obtained from the evaluation experiments for logistic regression.

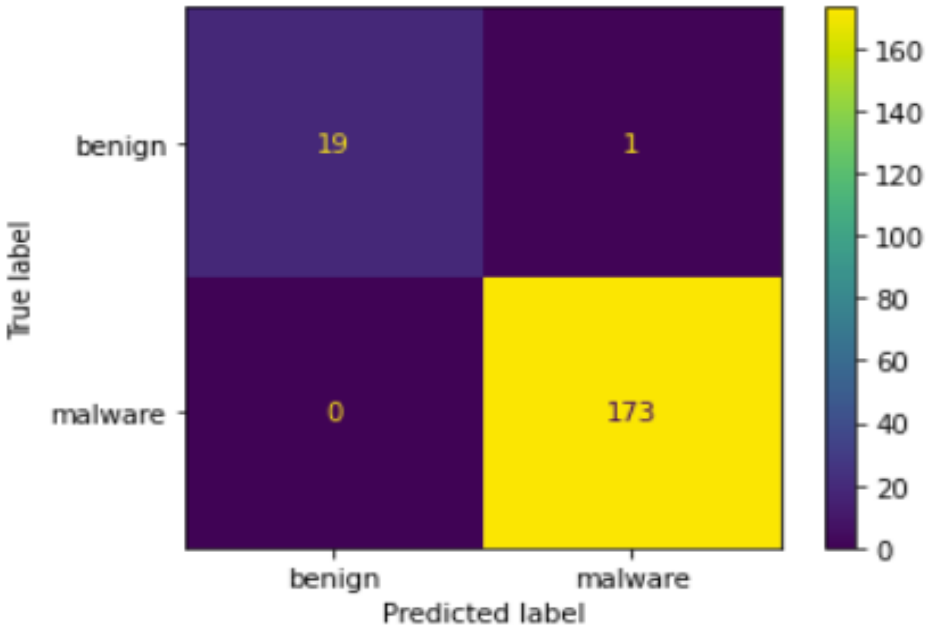

Figure 3.3 Confusion Matrix of Test Accuracy

Table 3.2 Classification Report for Logistic Regression (Test Accuracy)

	Precision (%)	**recall (%)**	**f1-score (%)**	**support**
Benign	100	95	97	20
Ransomware	99	100	100	173
Accuracy			99	193
Macro average	100	97	99	193
Weighted Average	99	99	99	193

Another metrics used to evaluate a classification model is the ROC (Receiver operating characteristics curve). The ROC curve provides insight into how well the model performed across all thresholds. It is a two-dimensional graph where the y-axis is the true positive rate, and the x-axis is the false positive rate. Figure 3.4 depicts the ROC curve obtained from the evaluation of the proposed logistic regression model.

A good model is expected to have an area under curve (AUC) value in the range of 0.7 to 1 which will indicate its level of separability as seen in figure 3.4. AUC values in the range of 0 - 0.6 indicate that the performance is either satisfactory or unsatisfactory as also seen in figure 3.4.

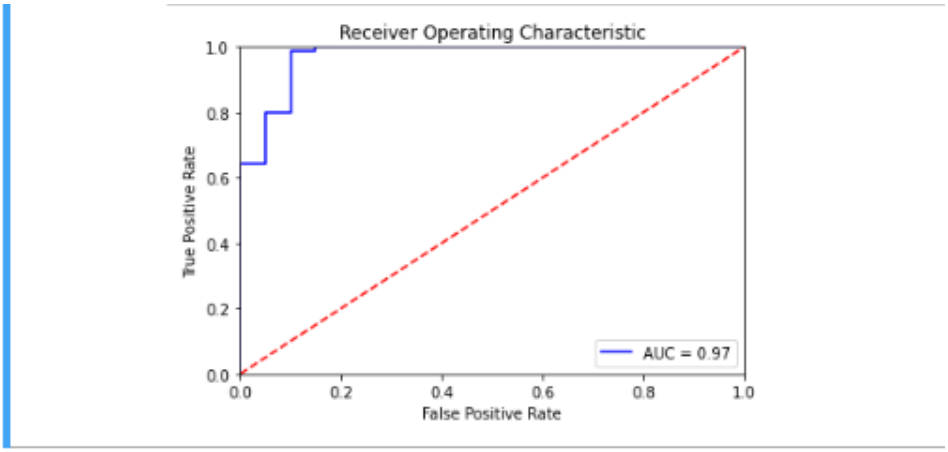

Figure 3.4 Proposed Logistic Regression Model ROC Curve and AUC

Our model performed very well with an AUC value of 0.97 as displayed in figure 3.4, which is in the excellent performance band in the reference figure 3.4.

Asides using a confusion matrix and classification report, there are other methods that can check for the test accuracy against a trained model. For training different models that fit poorly or require data to improve, one can use a learning curve theory. It is a theory that uses a graph to compare the performance of a model on training and testing data over a varying number of training models. With learning curve theory, one could see performance as the number of training points increases. The learning curve can be used for the case of different models to check for high bias and high variance if a model is not performing properly.

3.4 Backend Server-Side Code

The REST API for this project was developed with python using the Flask micro-framework. Flask is an excellent choice because it is lightweight and can be used to build scalable APIs. The API accepts JSON input as the request payload and uses the payload for prediction. The prediction result is also returned as a JSON response. Some of the features expected as payload includes but are not limited to RegQueryValue, HttpOpenRequest, embedded strings etc.

When the API receives the JSON payload, it uses it for prediction and loads the model into a pickle file. A Pickle is a python tool which allows you to save your model to avoid lengthy retraining and it can always be reloaded or deserialized for prediction purposes. The loaded model is then used to make predictions of either malware or benign, and the result is returned in the API response as JSON.

3.5 GUI

The graphical user interface (GUI) was built to enable users to interact with the application in a user-friendly graphical manner [29]. Additionally, users can also interact with the application via a command-line interface. The GUI is more convenient to use compared to the command-line interface. Figure 3.5 shows a picture of the GUI with a choose file and submit button. The “Choose File” enables a user to upload a test file in JSON format while the “Submit Button” performs an API call to analyze the JSON payload for classification as either ransomware or benign.

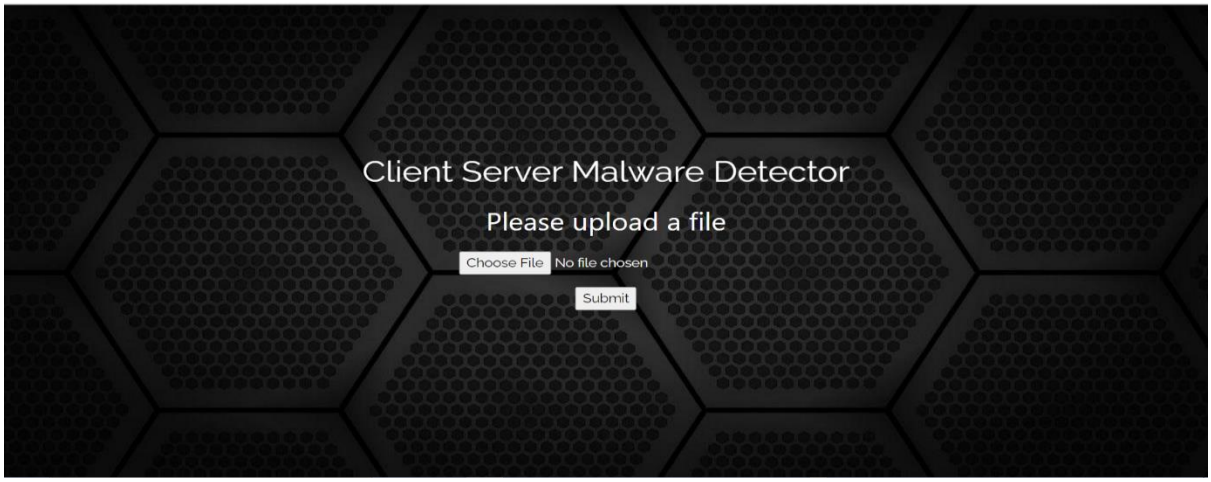

Figure 3.5 GUI for the Endpoint Detector

Chapter 4: Deployment and Experimentation

In this chapter, we would be setting up the environment for our experiment which involves the deployment of a Cuckoo sandbox into an Ubuntu Operating System (OS). The objective is to be able to extract ransomware features from any binary sample to be used for the validation of our trained model. The expectation is that our trained model should be able to return an ultimate prediction from the ransomware binary features. The prediction result should be either “yes” or “no”.

4.1 Experiment Environment

When setting up an experiment environment to analyze a malware sample, it is important to note that the host system could be at risk of being infected by the malware and because of this we deployed a Cuckoo sandbox on an Ubuntu host machine. The full experiment design is as shown in figure 4.1. It is also important to note that malware could spread through the network and hence, having a secure setup is very important to protect the local network to which other machines are connected to curb the spread.

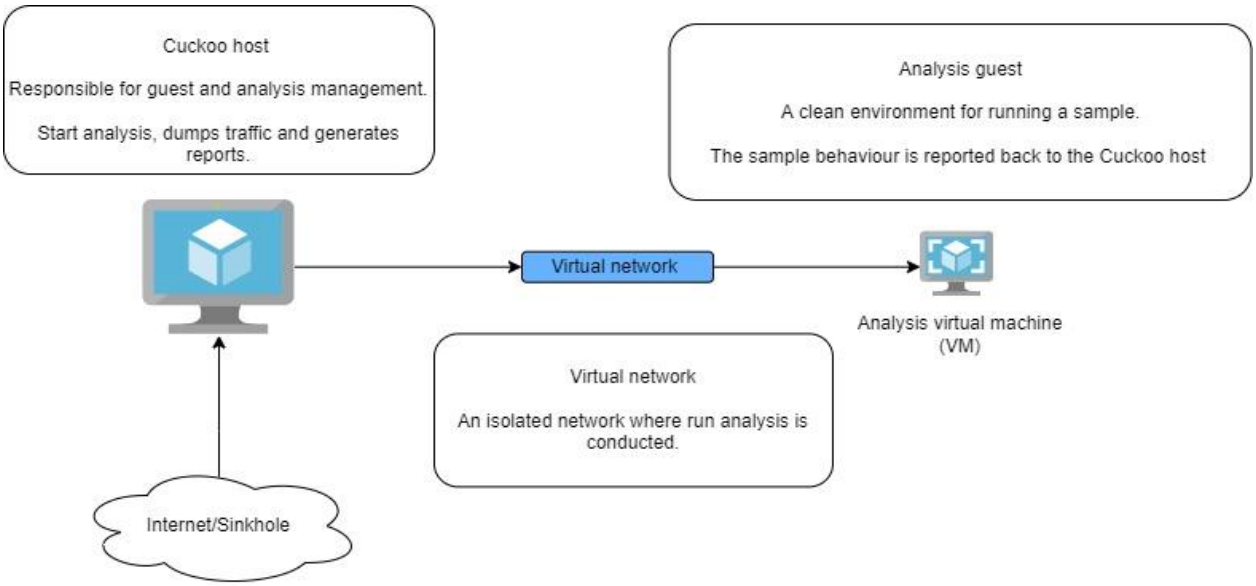

Figure 4.1 An Experiment Setup using Cuckoo Sandbox

The experiment setup as seen in figure 4.1 has Cuckoo sandbox which is responsible for the guest and analysis management. The Cuckoo sandbox comprises of two important machines viz the host machine and the guest machines. The host machine is the machine housing the installation of the Cuckoo sandbox and in our experiment setup, this host machine is an Ubuntu 20.4 OS. The guest machines enable the user to deploy virtual operation systems for analysis purposes. In a like manner these guest machines are running on windows 7 OS. By leveraging the native applications running in the host machine and the python command line utility, which is available in the Cuckoo sandbox, we were able to execute suspicious files downloaded from GitHub in the guest machine for both the static and dynamic analysis.

More so, the outbound traffic to the LAN network must be restricted to protect ransomware from spreading to other devices in the network. As a result, we assigned a static IP address to the guest machines to ensure they are segmented from the Cuckoo sandbox. We also configured a firewall to filter inbound and outbound traffic to and from the guest machine. This helps to filter the communication between a malware sample and its command-and-control server. The virtual network in the setup helps to isolate the network.

During the suspicious file execution, the Cuckoo result server runs continuously and logs the report of all operations that took place in the guest machine during the execution. For each binary file, multiple report files are logged and stored in different files and directories.

Additionally, snapshots were taken before conducting the analysis to record the state of the guest OS. The essence of the snapshots was to get the status of the machine before the analysis so that it would serve as a baseline for post experiment analysis especially when we compare the modifications to the Guest OS because of the ransomware sample.

The Cuckoo User Interface is as shown in figure 4.2.

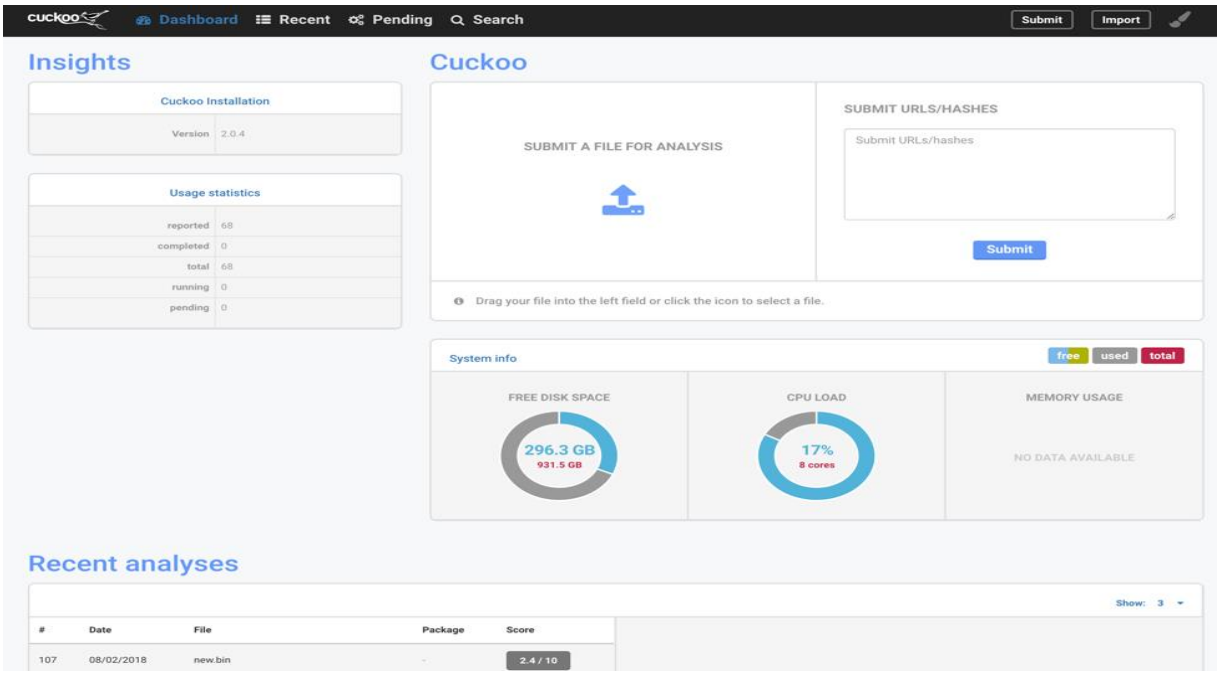

Figure 4.2 Cuckoo User Interface

When all the experiment setups have been completed, the next step will be to make some configuration changes to the Cuckoo configuration file. The configuration file can be accessed using the bash command cd ~/. cuckoo/conf [30]. The configuration changes take place in the following config files:

- cuckoo.conf: In cuckoo conf, we will ensure the machinery settings is pointing to the virtual box. We assigned a static IP address of 192.168.56.1 to the server and set the memory dump to yes. The rest of the configurations in the cuckoo.conf happens during installation.

- auxiliary.conf: In this configuration file, we configured and enabled the auxiliary modules.

- memory.conf: Memory.conf is the summary of the volatility configuration. We ensured the Guest_profile was set to the guest profile of the machine and set the delete_memdump to "no".

- processing.conf: In this configuration file we enabled, disabled, and configured all the processing modules which defines how to digest the raw data collected during the analysis.

- reporting.conf: In the reporting.conf, we enabled how we want the reports to be rendered after the malware analysis run.

- virtualbox.conf: It is important to name the virtual machine and the type of platform it is being hosted on. Also, we selected the GUI option to be able to view the Cuckoo results in a more user-friendly interface.

4.2 Experiment Procedure and Test Samples

After successfully setting up the environment for the experiment, we proceeded to the next phase which was to upload each of the binary samples unto the Cuckoo GUI and run them on the host for malware extraction. Figure 4.3 shows the entire workflow of the malware analysis.

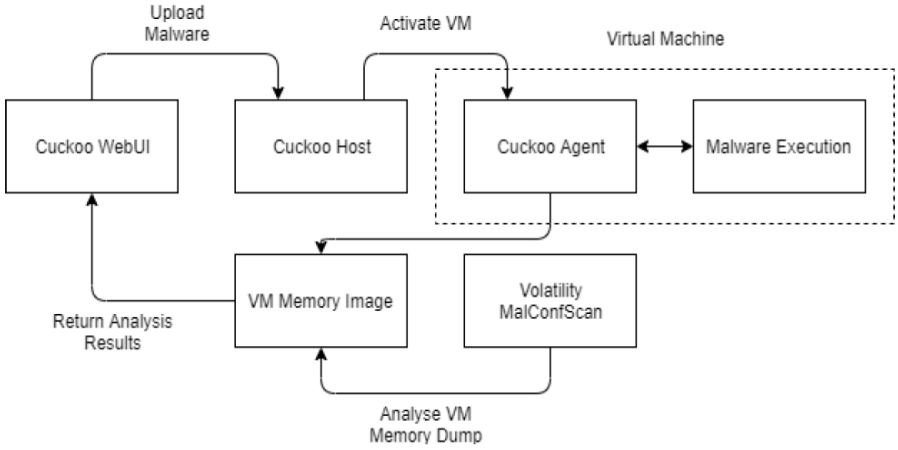

Figure 4.3 Malware Analysis Process

As seen in figure 4.3 the analysis process starts with the uploading of a binary sample file using the Cuckoo web user interface after which the Cuckoo host activates the Cuckoo agent running inside the virtual machine. The Cuckoo agent in turn commences the malware execution in the

agent.py terminal. The agent.py runs on a command line which acts as the reporting agent that informs the cuckoo host about the behaviour of the samples such as network or task activities. After the sample run is completed, the virtual machine automatically stops running and the analysed VM memory dump is logged on the host. The memory dump will show changes that occurred before and after the malware execution and at the same time, the volatility scans are also added to the return analysis result to show what happened in the virtual machine while the malware analysis was taking place. The logs are displayed on the Cuckoo terminal which also provides information on the IP address, the running task, and other changes that took place during the malware execution. Figure 4.4 is a screenshot of the running analysis.

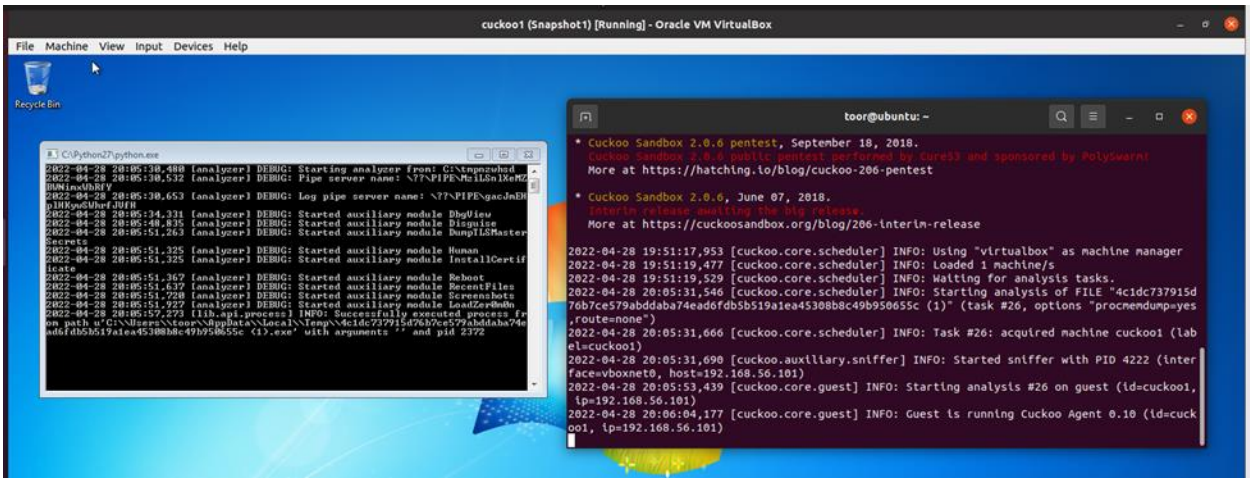

Figure 4.4 The Output of Cuckoo during an Analysis

Figure 4.5 shows the outlook of the user interface after the malware execution.

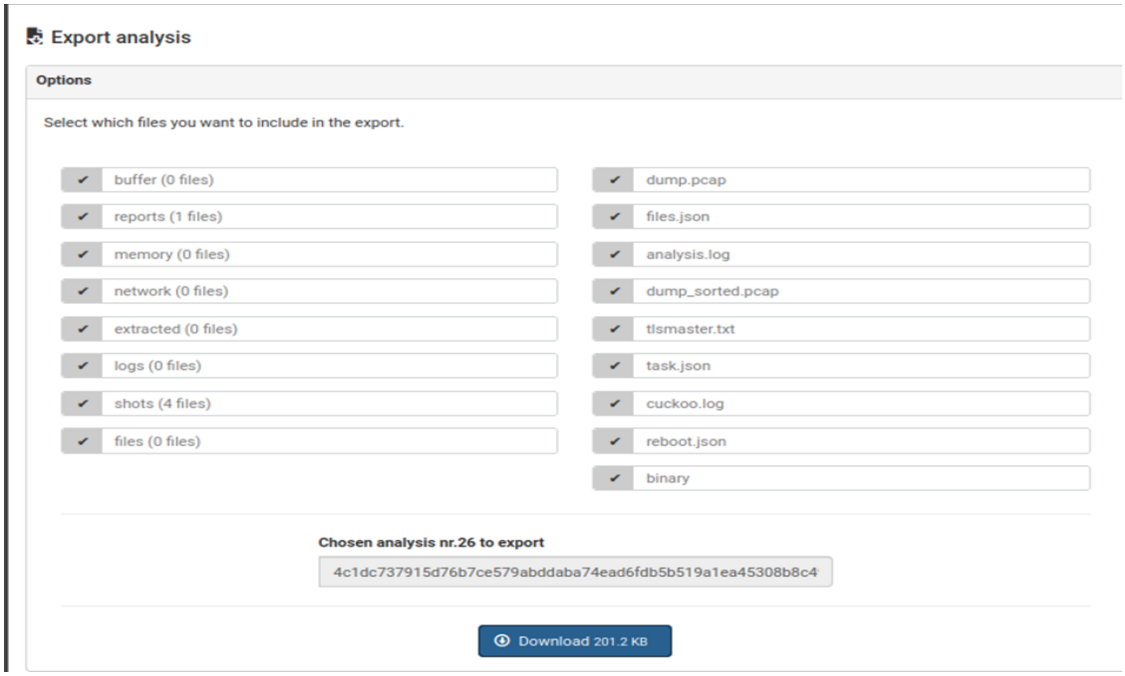

Figure 4.5 Export Analysis User Interface

The directory of the dump files from Cuckoo is shown in figure 4.6

```
.
|-- dump.pcap
|-- memory.dmp
|-- files.json
|-- logs
|   |-- 1232.bson
|   |-- 1540.bson
|   `-- 1118.bson
|-- reports
|   |-- report.html
|   |-- report.json
```

Figure 4.6 An Image of Cuckoo Directory

dump.pcap

This directory contains all the traffic of the network dump generated during each analysis performed on the ransomware and benign samples.

memory.dump

This directory contains the information of the memory dump of the analysis conducted.

file.json

This includes all the metadata information of the analysis in the form of JSON encoded format.

logs

The reported logs are stored in this directory with the extension.bson.

reports

All the analyses are stored in the reports folder using JSON format. It includes information about the analysis and its duration during the task.

Ten of the malware samples extracted from the experiment procedure and test samples are presented in table 4.1 [31].

Table 4.1 Ransomware Families Extracted

Ransomware samples	**Benign samples**
Petya	7-zip
NotPetya	WinRAR
CTBLocker	WinZip
Satan	CSV (Comma-separated values)
Zeta	Adobe Acrobat
TeslaCrypt	GIMP (GNU Image Manipulation Program)
Crysis	TrueCrypt
Locky	Ms Excel (Microsoft Excel)
Jaff	Ms Doc (Microsoft Word Document)
WannaCry	.txt (text file)

Listing of the Petya ransomware family analysis result files is shown in figure 4.7 as an example of the content when drilled down.

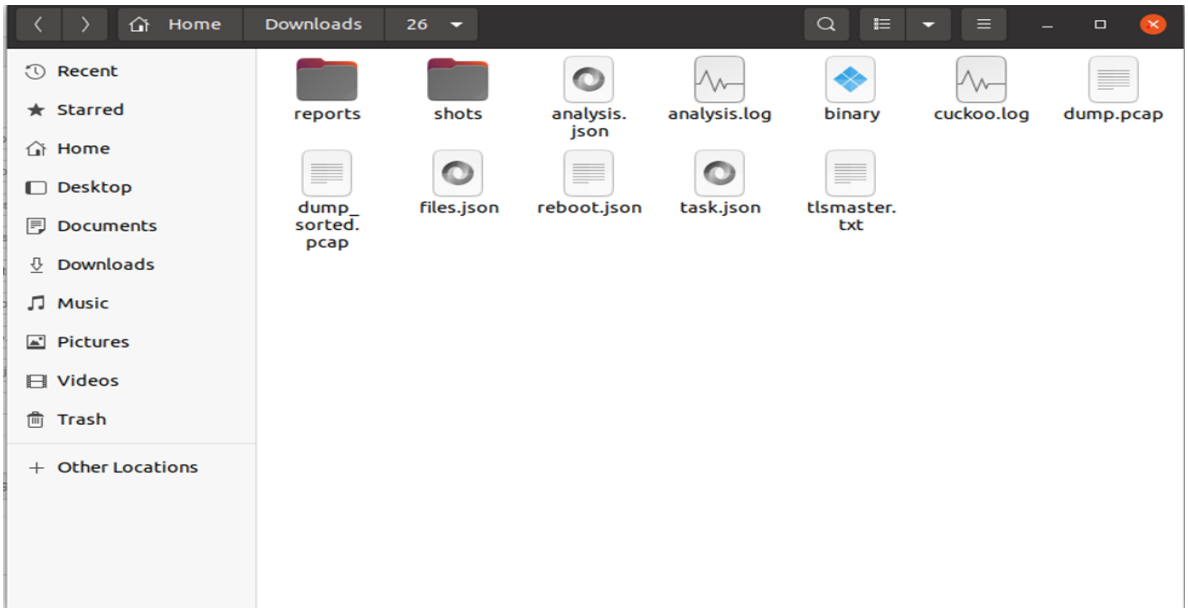

Figure 4.7 An Image of the Cuckoo Directory for Petya

4.3 Experiment Results

One of the capabilities of Cuckoo sandbox is to generate a report of both the static and dynamic analysis of the analyzed malware samples. We exported the extracted binary programs from the report.json file which contains some of the key features needed for us to validate the earlier trained endpoint detector model.

We used a script written in python programming language for the extraction of the features to be used for the validation of the detection model. We collected some of the dynamic features from the ransomware family samples listed in table 4.1. All the ransomware binaries used in this project were analysed to get a detailed understanding of the behaviour of each variant.

Running these features against our trained model will help to predict whether they are ransomware or benign sample. Like the experiment conducted by Jethva et al [13], our trained model contained 51,556 features which we processed on the built machine learning model.

For the detection result, we selected ten different families of ransomware binaries and ten benign samples. Table 4.2 shows the detection result we got when we ran the new samples to validate the trained model [31] [32].

Table 4.2 Detection Results

Family	**Detected as Ransomware**
Petya	Yes
NotPetya	Yes
CTBLocker	Yes
Satan	Yes
Zeta	Yes
TeslaCrypt	Yes
Crysis	Yes
Locky	Yes
Jaff	Yes
WannaCry	Yes
Benign (7-zip)	No
Benign (WinRAR)	No
Benign (WinZip)	No
Benign (CSV)	No
Benign (Adobe Acrobat)	No
Benign (GIMP)	No
Benign (TrueCrypt)	Yes
Benign (Ms Excel)	No
Benign (Ms Doc)	No
Benign (.txt)	No

4.3.1 Evaluation of the New Samples

Figure 4.8 shows the confusion matrix from running the new ransomware and benign binary samples.

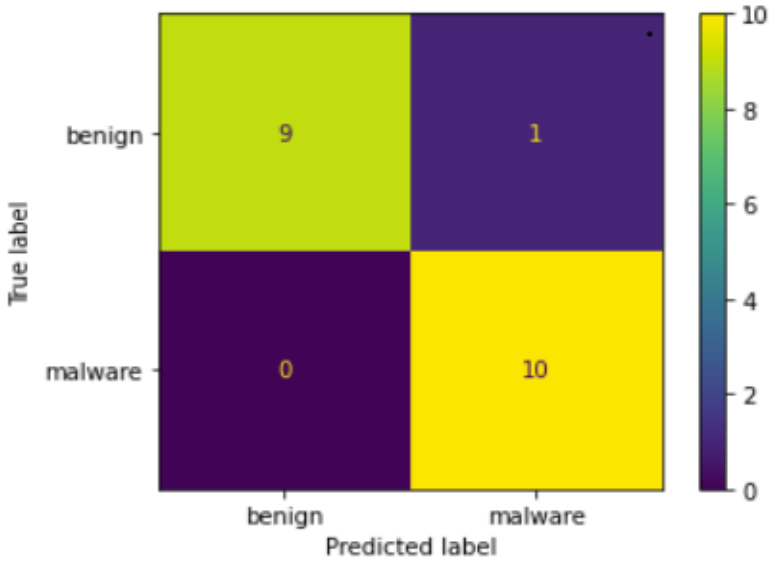

Figure 4.8 Confusion Matrix of the Detection

The classification report is represented in table 4.3.

Table 4.3 Classification Report for the Detection Result

	Precision (%)	**recall (%)**	**f1-score (%)**	**support**
Benign	100	90	95	10
Ransomware	91	100	95	10
Accuracy			95	20
Macro average	95	95	95	20
Weighted Average	95	95	95	20

It can be observed that the experiment produced an overall accuracy of 95% and a false positive rate of 5% which indicates that the model gave a good performance of the newly tested samples. The model only misclassified one of the benign samples as a ransomware application. Also as seen in table 4.3, the true positive rate or recall is 100% which implies that the model detected all the new binary test samples.

Chapter 5: Conclusion

5.1 Contribution Summary

There has been a significant increase in the endpoint detectors that can curb and detect the spread of ransomware today. In the proposed endpoint detector, we were able to have a model to do a detection out of the samples we got from the Cuckoo sandbox. Over the years, most experiments have focused on dynamic analysis because it has proven to give a better detection, especially considering its mode of operations.

In this project, we started by reporting the trends of ransomware over the past decades and how it has impacted different businesses and organizations worldwide. We also explored different types of malware analysis conducted by different researchers and their findings.

To build the detector, we started by carrying out model selection for the best machine learning algorithm for the binary classifications. In the process, Logistic Regression was selected out of the three tested classification machine learning algorithms for the building the trained model. To evaluate the trained model, we designed an experiment set up using the Cuckoo Sandbox to run some malware execution to extract samples for the model validation. We then used python scripts for the feature extraction and then ran them on the trained model.

In addition to the trained model, we built an endpoint detector to run samples on the trained model to optimally classify the samples as either yes or no. After the experimentation, we can say that these features of ransomware play a key role in building a good model which produced an accuracy of 95% with newly tested samples. Hence, we have been able to add to the cybersecurity collection for ransomware detection and analysis.

5.2 Future Work

Although this project was a success in detecting ransomware and benign binary samples from the wild, with an accuracy of 95% and a false positive rate of 5% there is still room for improvement. Our recommendation is that researchers should take this research to the next level by leveraging the following improvement strategies:

- Collection of more ransomware samples: The follow-up experiment in this project was conducted using a relatively limited number ransomware and benign binary samples from GitHub which is certainly not enough to validate a good model. It will be interesting to see the performance of a good model when used with more ransomware samples. The outcome of the analysis using more ransomware and benign binary samples will strengthen the adoption of the purposed detection system.

- Leveraging Deep Learning techniques and other machine learning techniques: During the model development we selected the machine learning algorithm with the highest accuracy which was logistic regression. However, as a future work, it will be good if researchers can expand the scope for model selection and leverage deep learning techniques which has proven to be successful in research such as speech recognition, robotics, and even ransomware analysis. Some of the useful deep learning algorithms to consider may include but not limited to recurrent neural networks, and feed-forward networks. This approach in-turn could lead to an improvement in ransomware detection by generating a higher model accuracy and a lower false-positive rate.

- Multiclass classification: One of the future works of this project will be to explore multiclass classification; it is a machine learning method that classifies instances into one of two or more classes. Unlike binary classification where instances are classified into exactly one of two classes. It will be good for researchers to expand the scope from feature engineering to modelling and evaluation of the metrics using multi class classification. The aim would not only focus on ransomware but also explore other malware variants which is of high concern. Mutliclass classification will enable assigning a malware sample to a specific ransomware family, rather than just categorinzing the sample as ransomware or benign.

- Integration of MLOps: MLOps can be used for the continuous integration, delivery, and automation for better model delivery. Continuous monitoring is one of the MLOps processes and if included in the pipeline build, can store samples of request-response payloads in a serving log store. This process will help keep track of important metrics of the model, such as accuracy, precision, f1 score, etc. By doing so, once there is a drift in

any of the metrics, a newly trained model can be pushed into production in the pipeline, preprocessed, and re-trained to get a better result from new ransomware samples. Adopting this approach could lead to a reduction in the false positive rate and increase in the model accuracy.

- Complementing existing multi-layer security solutions: Additionally, our approach can further be complemented by integration with existing multi-layer security solutions to protect crucial data from ransomware attacks. This can be done by the integration of the endpoint detector in a sandbox system or a SIEM (security information and event management) via API (Application Programming Interface). This approach will support compliance, threat detection, and robust security incident management.

- Cloud Hosting: Future work in this project can also include the expansion of the analysis by deploying the API on a cloud-hosted platform such as Azure, AWS, Google for real-time inferencing or scoring.

Reference

[1] *81 Ransomware Statistics, Data, Trends and Facts for 2021*
https://www.varonis.com/blog/ransomware-statistics-2021

[2] *A Timeline of The Biggest Ransomware Attacks*
https://www.cnet.com/personal-finance/crypto/a-timeline-of-the-biggest-ransomware-attacks/

[3] *Common Types of Ransomwares*
https://www.datto.com/blog/common-types-of-ransomware

[4] Yvan Labiche *Combining Static and Dynamic Analyses to ReverseEngineer Scenario Diagrams* Department of Systems and Computer Engineering Carleton University Ottawa, Canada

[5] Chen, Q.; Islam, S.R.; Haswell, H.; Bridges, R.A. "*Automated ransomware behavior analysis: Pattern extraction and early detection"*. In Proceedings of the International Conference on Science of Cyber Security, Nanjing, China, 9–11 August 2019; pp. 199–214.

[6] *The best brief guide to clone phishing available on the web*

https://cybertalk.org/2022/01/14/what-is-clone-phishing-and-why-it-matters

[7] Q. Jerome, K. Allix, R. State, and T. Engel, *"Using opcode-sequences to detect malicious Android applications,"* 2014 IEEE Int. Conf. Commun. ICC 2014, pp. 914–919, 2014

[8] M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo, *"Data mining methods for detection of new malicious executables,"* pp. 38–49, 2002

[9] J. Z. Kolter and M. A. Maloof, *"Learning to Detect Malicious Executables,"* Machine Learning Data Mining. Comput. Secur., vol. 1, no. 212, pp. 47–63, 2006.

[10] A. Kharraz, W. Robertson, D. Balzarotti, L. Bilge, and E. Kirda, *"Cutting the gordian knot: A look under the hood of ransomware attacks,"* Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 9148, pp. 3–24, 2015.

[11] Chen, Q.; Islam, S.R.; Haswell, H.; Bridges, R.A. *"Automated ransomware behavior analysis: Pattern extraction and early detection."* In Proceedings of the International Conference on Science of Cyber Security, Nanjing, China, 9–11 August 2019; pp. 199–214.

[12] Bahrani, A.; Bidgly, A.J. "*Ransomware detection using process mining and classification algorithms."* In Proceedings of the 2019 16th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC), Mashhad, Iran, 28–29 August 2019; pp. 73–77.

[13] Jethva, B.; Traoré, I.; Ghaleb, A.; Ganame, K.; Ahmed, S. "*Multilayer ransomware detection using grouped registry key operations, file entropy and file signature monitoring*." J. Comput. Secur. **2020**, 28, 337–373

[14] Alam, M.; Sinha, S.; Bhattacharya, S.; Dutta, S.; Mukhopadhyay, D.; Chattopadhyay, A. RAPPER: "*Ransomware prevention via performance counters*." arXiv 2020, arXiv:2004.01712.

[15] Al-Hawawreh, M.; Sitnikova, E. *"Industrial Internet of Things based ransomware detection using stacked variational neural network."* In Proceedings of the 3rd International Conference on Big Data and Internet of Things, Melbourn, Australia, 22–24 August 2019; pp. 126–130.

[16] A. Continella et al., "ShieldFS," Proc. 32nd Annu. Conf. Comput. Secur. Appl. - ACSAC '16, pp. 336–347, 2016.

[17] N. Scaife, H. Carter, P. Traynor, and K. R. B. Butler, *"CryptoLock (and Drop It): Stopping Ransomware Attacks on User Data,"* Proc. - Int. Conf. Distrib. Comput. Syst., vol. 2016-Augus, pp. 303–312, 2016.

[18] D. Sgandurra, L. Muñoz-González, R. Mohsen, and E. C. Lupu, *"Automated Dynamic Analysis of Ransomware: Benefits, Limitations and Use for Detection,"* arXiv 2016, arXiv:1609.03020.

[19] Yang, C.-Y.; Sahita, R."*Towards a Resilient Machine Learning Classifier-a Case Study of Ransomware Detection*." arXiv 2020, arXiv:2003.06428.

[20] M. Cantonment, *"RansHunt_ A support vector machines-based ransomware analysis framework with integrated feature set"* - IEEE Conference Publication., pp. 22–24, 2017.

[21] N. Hampton, Z. Baig, and S. Zeadally, *"Ransomware behavioural analysis on windows platforms,"* J. Inf. Secur. Appl., vol. 40, pp. 44–51, 2018.

[22] Kok, S.; Abdullah, A.; Jhanjhi, N.; Supramaniam, M. "*Prevention of crypto-ransomware using a pre-encryption detection algorithm"*. Computers 2019, 8, 79.

[23] *ISOT Ransomware Dataset Overview*
https://www.uvic.ca/ecs/ece/isot/assets/docs/isot_ransomware-dataset-readme.pdf

[24] *Workflow of a Machine Learning project*

https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94

[25] Dhairya Kum. "*Introduction to Data Preprocessing in Machine Learning".* published in Towards Data Science: https://towardsdatascience.com/introduction-to-data-preprocessing-in-machine-learning-a9fa83a5dc9d

[26] Artur J. Ferreira, Mário A.T. Figueiredo "*Efficient feature selection filters for high-dimensional data."*

[27] Urszula Stańczyk and Lakhmi C. Jain. *"Feature Selection for Data and Pattern Recognition"*

[28] *Evolution of machine learning:* https://www.sas.com/en_ca/insights/analytics/machine-learning.html

[29] *Graphical User Interface:* https://en.wikipedia.org/wiki/Graphical_user_interface

[30] *Cuckoo sandbox:* https://cuckoosandbox.org/

[31] *Ransomware samples:* https://github.com/fabricmagic72/malware-samples/tree/master

[32] *Ransomware samples:* https://github.com/iosifache/DikeDataset/ree/main/labelsransom