Imperial College
London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Automatic Detection of Ransomware Attacks on Linux Machines

Author:
Bianca Tazlauanu

Supervisor:
Sergio Maffeis

Submitted in partial fulfillment of the requirements for the MSc degree in Advanced Computing of Imperial College London

September 2021

Abstract

It is well known that security breaches have been on a rise in the recent years, with hackers constantly adopting more sophisticated methods to deploy in their attacks. Among the different types of security attacks, ransomware has seen a particular surge in incidence. While historically it was a type of attack mostly directed towards Windows machines, the trend is now shifting. More and more Linux machines are becoming victims of ransomware attacks with devastating repercussions. Businesses are rendered incapable of running, suffering not only a huge loss in information, but also being financially impacted when paying ransom in an attempt at recovering their data.

Existing work shows what indicators of compromise are most representative for Windows ransomware along with different methods of stopping these types of attacks from being successful. In comparison, considerably less work has been done for Linux machines. One of the main challenges for this task being the lack of a considerable amount of samples to analyse. The project aims to bridge this knowledge gap on Linux ransomware, determine which indicators are appropriate signals for the attack and propose a dynamic detection method of malicious applications which attempt to execute a ransomware attack.

Acknowledgments

I would like to thank my supervisor, Prof. Sergio Maffeis, for the guidance and support that he offered throughout the entirety of this project and also Prof. Emil Lupu for his advice and insights.

Last, but not least, many thanks to my family and friends for always being present and making everything possible.

Contents

Chapter 1

Introduction

1.1 Context

According to PwC's latest Annual Global CEO Survey for 2021, almost half of the interviewed CEOs are afraid of cyber threats impacting their organisation's growth [1]. In the report, PwC brings forth four main takeaway points, one of them being fortifying the cyberspace. In the current context in which global regulations for the use of the Internet are tentative, we are just now starting to realize the potential danger of online platforms and lack of data privacy.

Moreover, the year 2020 was met with a surge in security breaches. Contributing to the ever rising risk of security attacks was also the socio-economic situation brought by the COVID-19 pandemic which resulted in an increase in the use of e-commerce and Software-as-a-Service (SaaS) solutions [2]. These trends opened up more opportunities for malicious actors to take advantage of. In 2020 we have also seen one of the largest software companies, Software AG, being the target of a ransomware attack [3]. Alongside it, companies like Sopra Steria [4] and Seyfarth Shaw LLP [5] were also affected by ransomware attacks at the end of the year which brought about millions of dollars in losses. Even more, in the 2021 Mid Year Report released by cyber security company Check Point [6] it shows that ransomware attacks have increased by 93% in the first half of the year compared to the previous year.

1.2 Cyber Attacks

There is no such thing as a safe computer if it is connected to the Internet. Once becoming part of a network, it automatically gets exposed to a myriad of possible attacks like malware (Trojan horses, spyware, ransomware, viruses), phishing attacks, drive-by downloads, network attacks (Denial-of-Service, Man-In-The-Middle) and web attacks (Cross-Site-Request-Forgery, Cross-Site-Scripting). Moreover, people are generally not well equipped with the knowledge to recognise even less sophisticated attacks. According to the Verizon Data Breach Investigations Report [7], simple phishing attacks account for 22% of all security incidents reported in 2020.

Often, the triggering factor for an attack to take place successfully is human error. Wrongly configured cloud infrastructure, misplaced credentials or credentials that can be easily brute-forced are just some of the problems that leave individuals and enterprises alike at risk. There is also the social engineering aspect which enables phishing attacks to be very profitable for hackers. As mentioned by Verizon, the COVID-19 pandemic not only increased the attack surface, but also the usage of mobile phones and tablets — platforms where this type of attack is more likely to succeed [8, 9].

Additionally, the growing risks presented by Advanced Persistent Threats (APTs) have also been widely observed. As reported by the security company Mandiant, a considerate part of security breaches that are affecting companies and large organisations are caused by such APTs [10]. One of such actors which has been monitored by Mandiant since 2006 managed to extract "hundreds of terabytes of data" from more than 141 organisations.

1.3 Objectives

The project's objective consists in finding a novel approach to detecting ransomware attacks on Linux systems. Since most of the attacks previously happened on Windows machines, very little work has been done in examining and providing detection mechanisms for Linux ransomware. The current work attempts to bridge that knowledge gap by looking at how to identify a ransomware attack on Linux machines. Additionally, more information will be provided about the specific indicators of a malicious application compared to a benign one.

In order to detect a ransomware infected application, a machine learning model will be built. One of the main challenges of the project is collecting appropriate data for the task since the available Linux ransomware samples are very scarce. As a result, one of the first tasks is finding alternative data sources needed to train the machine learning model.

Chapter 2

Background Information and Literature Review

2.1 Ransomware Attacks

A ransomware is a type of malware which encrypts data on the host machine and then proceeds to ask a ransom from the victim to recover the data. In a ransomware attack, the malicious actors would demand a ransom in exchange for the decryption key, without which the files generally cannot be recovered. Even so, organizations like Certified Information Systems Auditor (CISA) and FBI discourage paying the ransom as there is no guarantee that the victim will recover their data and the money might also be funneled into other criminal activities [11].

In one of his recent publications, security researcher Chad Anderson analysed the ransomware landscape and which variants are the most prolific [12]. He points out that with the multiplication of hacker organisations, it becomes harder and harder to keep track of ransomware strains that can be currently found in the wild. Moreover, security attacks started to evolve into more complex business models in which organisations might leverage each other's services. One such example is the partnership between botnet operations and ransomware attacks. In the article [12], Anderson presents the case of a Qakbot infection leading to a REvil ransomware attack.

Additionally, the report presents the top 3 most prolific ransomware families: Conti (16% of attacks), Maze (11.2% of attacks) and REvil (9.9% of attacks). The entire distribution of attacks based on the family of ransomware that was responsible for it is presented in Figure 2.1. Looking at the top 3 prevalent ransomware families is essential in understanding the general thinking that attackers have when designing their approach.

Starting with the most prolific family of ransomware at the moment, Conti has been seen since 2020 and it is a type of ransomware affecting Windows machines. Conti is known to first delete the Volume Shadow Copies, which is generally used on Windows machine to back up and restore lost data [13]. Once it is deleted, the ran-

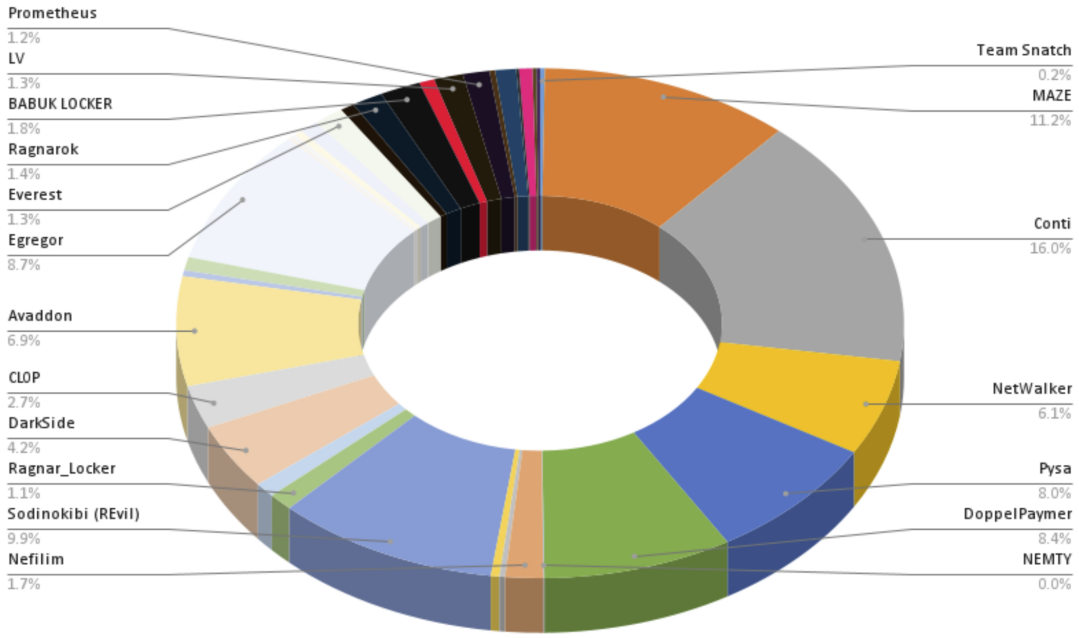

Figure 2.1: Attack frequency grouped by ransomware family [12].

somware proceeds to encrypt the data by using a custom made version of the AES-256 encryption algorithm which is significantly faster than the regular one. This speeds up the attack on the machine. The Conti ransomware first gains access by using TrickBot, Buer, BazarBackdoor or AnchorDNS and then proceeds in exploiting a vulnerability in the Atera registration process to gain trial access to an Atera agent. Atera is a remote monitoring system which can give the user shell execution needed by the ransomware to proceed with its attack.

In the beginning of 2021 at least 16 attacks targeted the United States' healthcare, some of which included 911 emergency call dispatchers [14]. Moreover, at least 400 institutions worldwide have been the victim of Conti ransomware according to a Federal Bureau of Investigation (FBI) Flash Advisory.

On second place there is the Maze ransomware group. It targets Windows systems and has been discovered in 2019 [15]. Maze has become well known for threatening their victims in disclosing their data if the ransom is not paid. They would typically infiltrate the target computers by taking advantage of vulnerable virtual private network (VPN) and remote desktop (RDP) servers. One of their most famous attack is on the technology firm Cognizant which resulted in loses of tens of millions of dollars for the company.

On third place is REvil, a ransomware which is based in Russia. Similarly to Maze ransomware, the REvil organisation would also threaten their victims to disclose their private information [16]. They would do so on their page Happy Blog. Since the publishing of the report, REvil vanished from the internet and they stopped their

activity [17]. Compared to other ransomware strains, REvil uses particular methods of social engineering to convince the user to gain elevated privileges. Another approach by the ransomware is to use Windows Safe Mode to encrypt the files.

2.2 Ransomware Types and Current Trends

Ransomware uses asymmetric encryption to encrypt the files on a victim machine. Unless the encryption has a flaw in its implementation, the encryption is irreversible. Ransomware families can be split into two main categories: Crypto-Ransomware and Locker-Ransomware.

Locker ransomware is characterised by locking the infected host, making the user lose access to the machine. This type of ransomware does not encrypt the data, but it is rather focused on denying access to the entire computer. One example of such ransomware is Petya which can restrict access the infected computer or the data on that computer. In contrast, crypto ransomware is responsible for directly encrypting the files on the machine, no matter the type of the file.

Crypto ransomware generally uses encryption methods like RSA, AES or other encryption libraries found on a system [18]. There are many examples of crypto ransomware, e.g.: Locky, WannaCry, REvil, RansomEXX. Some crypto ransomware are targeting Windows operating systems while others are focusing on Linux. Some ransomware types, like RansomEXX started as a Windows ransomware variants and later moved on to Linux.

There are a couple of additional types of ransomware that are less known. One of them is the hybrid mix of locker ransomware and crypto ransomware which inherits traits from both groups. Another one is a more recent type of ransomware which are called “crypto miners” — they infect the host and convert it into a mining resource for cryptocurrencies [19].

A new trend that has recently emerged in ransomware attacks consists of hackers not only encrypting the private data, but also collecting it and sending it to a Command&Control (C&C) server. After the data is successfully backed up on the attackers servers and encrypted on the victims machines, the attackers threaten to release the confidential information on the Internet. In August of 2021, the computer hardware manufacturer Gygabyte was the target of a such attack, falling victim to the RansomEXX ransomware strain. Subsequently, the attackers requested ransom in exchange for not making 112GB of private data public [20].

2.3 Attack Stages

A typical ransomware attack consists of multiple stages [21] in which the attacker is trying to get a foothold in the system and elevate their privileges in order to encrypt

the data and maintain its presence. The phases of a ransomware attack usually occur in several stages, as presented in Figure 2.2.

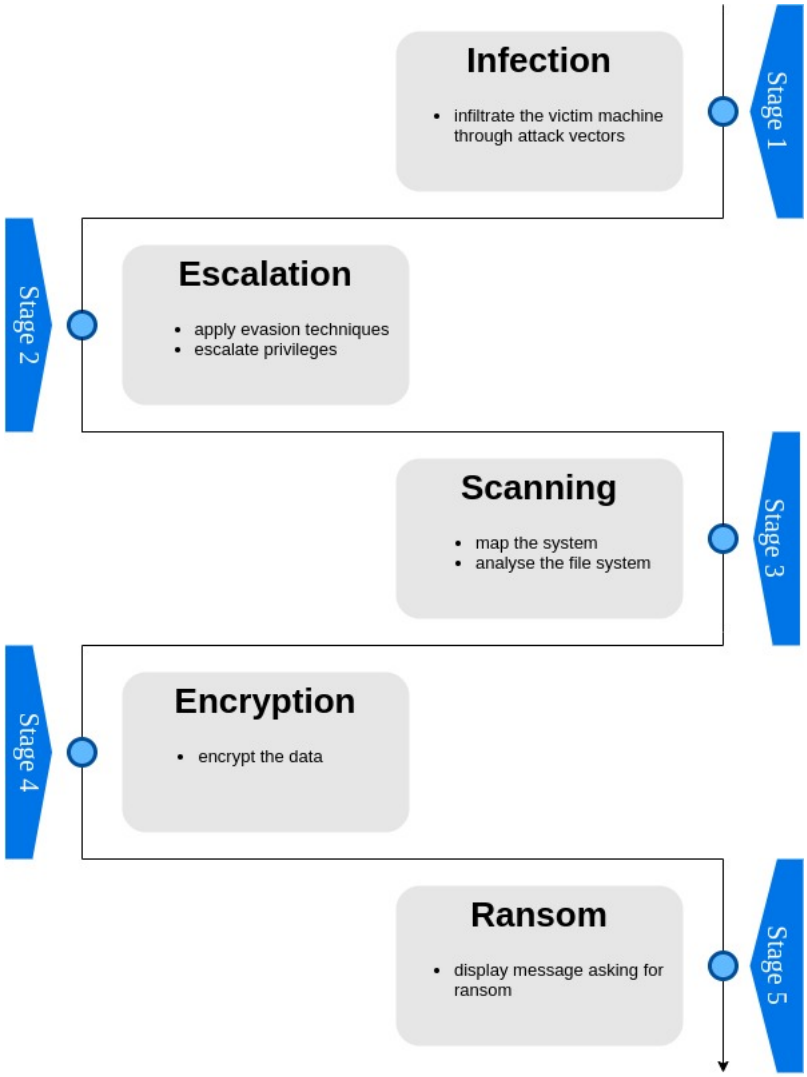

Figure 2.2: Stages of a ransomware attack.

Phase 1: Infecting the victim's machine. In the initial stage, a malicious application is delivered to the victim's machine through various methods. The methods are generally similar to the ones used in other types of attacks.

Oftentimes, a host becomes infected after the user accesses an untrusted website or opens a file delivered through email attachments. Phishing attacks are one of the most popular methods used by ransomware to gain a foothold in the system. Other methods include exploiting leaked credentials or easy to break credential by hackers to get access in the system. After successfully infiltrating, criminal organisations can proceed in placing malicious files in the system and start the ransomware attack. Another popular vector attack is represented by taking advantage of the Remote

Desktop Protocol (RDP) which is generally ignored by users from a security point of view making it an easy target for hackers. Exploiting weak points in the configuration of the RDP became even more popular in 2021 after more and more companies started to rely on it for remote work. Unpatched and not up-to-date software is yet another door that hackers frequently use for gaining access to systems.

Some examples of ransomware and their methods of infiltration are: Cerber, which uses spear-phising emails, or WannaCry. which uses the EternalBlue exploit which uses vulnerabilities in Microsoft's Server Message Block (SMB) protocol.

Phase 2: Anti-analysis and mapping: During this part of the attack, the malware collects information about the host system in order to detect if it, for example, runs in a virtual environment. This type of techniques are designed by the attackers in order to make investigation of the malware by security researchers more difficult.

Phase 3: After the malware finished collecting information about the host system, it attempts to discover how the local file system looks like, exploring storage volumes and partitions. Additionally, the malware might contact a C&C server in order to obtain the encryption keys or other configuration data.

Phase 4: During this stage, if the malware has not already obtained the encryption keys, it would proceed with that step first, followed by encrypting the file system or locking the system. Additionally, for some variants of ransomware, during this stage the malware might first transmit the data to the C&C server before encrypting it. Often times, the ransomware would also add specific extensions to the encrypted files like `.encrypted`, `.locky`, `.petya`, `.xyz` and so on. Each ransomware generally uses a specific file extension.

Phase 5: In the last stage, the attackers will make their demands, asking for ransom by displaying messages on the display along with the account details where the money should be sent to. Since cryptocurrencies have the property of being hard to trace, often times it is the preferred currency used for receiving ransom payments by the attackers. Additionally, at this stage the ransomware might clean its tracks by self-destructing.

2.4 Characteristics of Ransomware

2.4.1 Types of Encryption

The way data encryption is handled by each ransomware differs. One approach done by ransomware is to directly encrypt the files on the infected machine. Another class of ransomware instead prefers to copy the files to a different location, encrypt them there and then move them to the original location. The last category of ransomware would read the files, encrypt them and then proceed to copy the encrypted files to a new location and delete the original files.

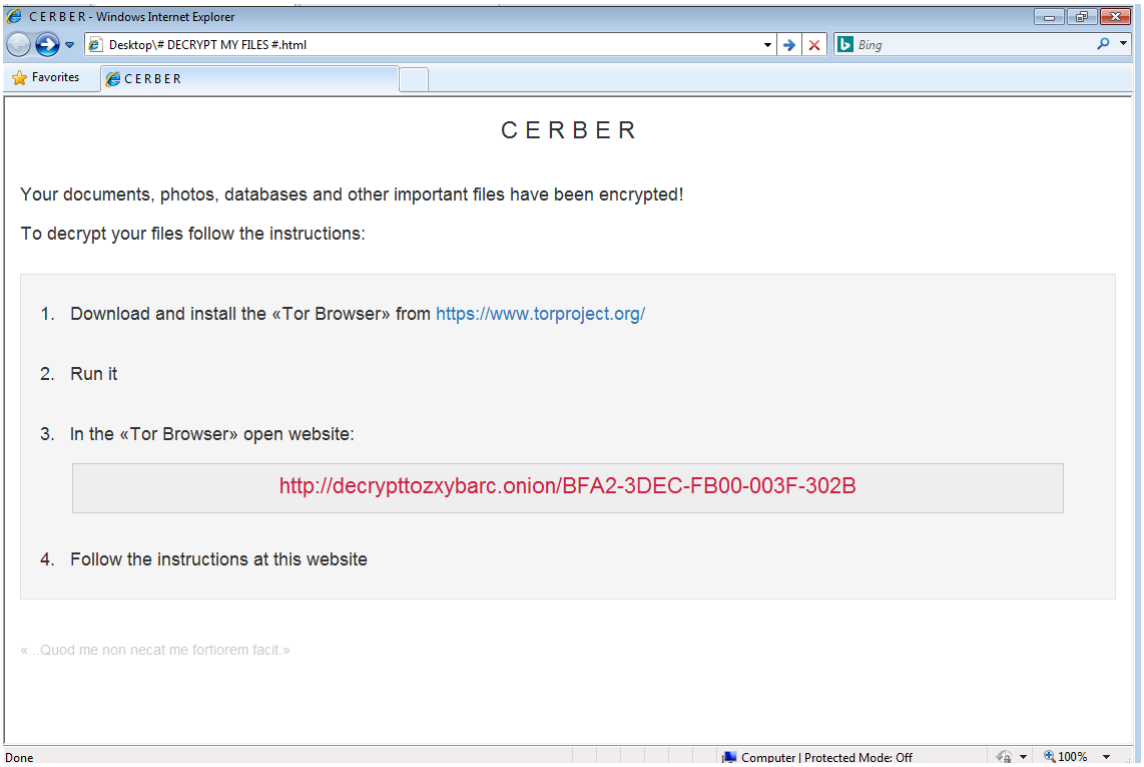

Figure 2.3: Cerber ransomware message.

Ransomware	Encryption Algorithm
Avadon	AES-256
DarkSide	Custom Salsa20
REvil	Curve25519 and Salsa20
MAZE	ChaCha 20 and RSA
Conti	custom AES-256
DoppelPaymer	RSA-2048 and AES-256
NetWalker	Salsa20
VaultCrypt	RSA-1024

Table 2.1: Encryption algorithms used by different ransomware variants.

Some of the most used encryption libraries are Microsoft Crypto Provider, OpenSSL, PolarSSL and GnuPG. In order to evade the detection by Host Intrusion Detection Systems (HIPS), some ransomware families would choose to develop their own implementation of the algorithms. After the ransomware finishes encryption, they would proceed to delete the encryption key. When it comes to the time needed to perform the encryption it varies from minutes to hours depending on the type of encryption, the amount of data that needs to be encrypted and, if the files are also send back to the C&C servers, on the network bandwidth.

2.4.2 Evasion techniques

Different ransomware vary in their behavior, even though they generally follow the same workflow. Oftentimes, some of the stages might be re-ordered or other obfuscation techniques might be applied. Moreover, depending on the variant and family of ransomware, some might delay the start of their execution once downloaded onto the host.

Obfuscation is one of the evasion techniques which varies across different ransomware, e.g. reordering of instructions, reordering of code blocks, dead code insertion, register renaming, etc. Families of ransomware that apply these techniques are CryptoWall v3, Cerber and Petya [22].

Another technique used by ransomware to avoid being analyzed by security researchers is by detecting if it is running in a virtual machine. Ransomware does this by looking for VM-specific processes and service, e.g. `vboxtray` in VirtualBox. In the case of Windows, they look for virtual system specific registry keys or checking for VM system installation. Some ransomware also take advantage of the common BIOS serial numbers and MAC addresses that are generally found on VM systems. If the ransomware detects that its running on a virtual machine, oftentimes it will not run or it will self-destruct.

As a result it becomes a problem when hackers design their ransomware software to specifically avoid frameworks which are known to be used by researchers, e.g.: `Cuckoo Sandbox`. Another technique used by ransomware belonging to, e.g., the Locky family, is to encrypt their network communication.

2.5 Ransomware-as-a-Service

Ransomware-as-a-Service (RaaS) is a new business model adopted by hackers which gained popularity in recent years [23]. It provides tools which enables attackers to deploy ransomware attacks easily on their chosen targets. As a result, attackers do not require advance knowledge in computer systems in order to conduct a ransomware attack and the criminal organisations that offer this type of service earn a percentage of the payments.

Nowadays, criminals rely more and more on business models like RaaS in which they use already built ransomware variants [24]. RaaS is in principal similar to Software-as-a-Service and it enables even inexperienced criminals to deploy ransomware attacks. Attacks coming from RaaS generally use phishing emails or exploit kits in order to infect the target host.

Typical prevention mechanisms include network segmentation, using multi-factor authentication for remote access and strong spam and network filtering. Furthermore, the employees in the organisations should be trained to prevent ransomware

attacks. Lack of training and awareness among companies and organisations makes ransomware attacks a very profitable business for criminals, as 40% of those infected end up paying the ransom, which totals over a billion dollars per year [25].

One such RaaS provider is Darkside which publicly announced their services in August 2020 [26]. They encrypt sensitive information and backups and generally choose their victims among big companies which can afford paying the requested ransom. The way of operation differentiated itself from other RaaS by using stealthy techniques especially in the initial stages of the attack in order to avoid detection.

Darkside was observed to use compromised contractor accounts in order to access Virtual Desktop Infrastructure (VDI) necessary for remote access. After gaining a foot in the system, Darkside would use Remote-Desktop-Protocol (RDP) client through TOR via HTTPS over port 443 for command and control purposes. Another element common for Darkside attacks were the usage of Cobalt Strike, a penetration testing toolkit that was hosting Command&Control (C&C) servers. Moreover, it was observed that malicious actors would use multiple accounts for Virtual Desktop environment of the infected host and create .lnk files to keep track of the compromised accounts.

2.6 Network Level Analysis

2.6.1 Intrusion Detection Systems

Intrusion Detection Systems (IDSs) are responsible for scanning and analysing the traffic in a network in order to detect signs of an on-going attack of an intrusion [27]. With the growing level of sophistication at which the attacks are being deployed, it is getting harder and harder for the IDSs to intercept them. Moreover, encrypted communication limits their capabilities even more.

Provenance tracing is another important aspect that can help in the detection of sophisticated cyber attacks as it keeps track of the origin, time of creation and relationship of data in a system. Such a system can offer information about various aspects of the system from network connections, processes and files. One example of a provenance system is ProTracer [28] which uses both system even logging and unit level taint propagation to remain lightweight and efficient. Other examples include BEEP [29] and LogGC [30] which also manage to avoid the dependency explosion problem — finding a small subset of events that are actually dependent on each other while most of the events have a casual dependency — which is commonly occurring in these types of systems.

2.6.2 Security Information and Event Management

The current state of the art in terms of alert and monitoring for security threats consists of using Security Information and Event Management (SIEM) [31]. Some examples of SIEM systems are Splunk, LogRythm and IMB QRadar. These are powerful tools that can monitor and log information about a given system. SIEM systems collect information from logs belonging to host application and monitors along with network monitors. Paired with a set of rules, SIEM systems can attacks like detect Denial-of-Service (DOS), brute force or unexpected file transfers.

Even so, SIEM systems often lack an understanding of the bigger picture and the fine grained knowledge needed to detect APTs or more sophisticated attacks which can take place over a longer period of time.

2.7 Host Level Analysis

2.7.1 Static Analysis

One method used for detecting ransomware is by using static analyses on every executable file on the host machine. One such technique is signature analysis, a technique commonly used by virus detectors [32]. Through this method the code base of the analysed application is scanned for specific code sequences (signatures) which have been previously been found in known malware samples. Virus detectors that use this method relay on large databases containing a multitude of signatures from different types of malware. In order to maintain its efficiency, the database must constantly updated.

This method has a series of flaws which do not make it ideal for malware detection. First of all, it can only recognise previously detected and documented malware. If a new family or a new strain of ransomware appears, it will evade the antivirus. Moreover, since ransomware are becoming more evolved, attackers take advantage of sophisticated evasion techniques in order to avoid detection. Methods like code obfuscation can render virus detection systems completely ineffective.

Most of the common anti-virus software relies of some method of static analysis and a preexisting list of malware signatures used to scan the system. Some of the more popular antivirus solutions for ransomware are Bitdefender Antivirus Plus, AVG Antivirus, Avast Antivirus and Webroot Antivirus [33]. While these options offer some level of protection, they can become expensive and also consume considerable resources when performing a scan of the system. Even though most of the anti-virus options offer additional functionalities like banning illegitimate websites or offer protection against a wider range of malware types, they need constant updates and as previously mentioned they will be ineffective to new strains of ransomware.

2.7.2 Dynamic Analysis

Another detection method which tries to solve some of the short coming of static analysis is dynamic analysis [32]. Dynamic of behavioural analysis monitors a running application in order to detect if it exhibits any suspicious activities. Compared to static analysis, behavioural analysis is capable of identifying previously unknown ransomware variants by looking for specific activities which are common among this type of malware. As the stages of an attack have been previously described, a ransomware sample might be expected to have an abnormal amount of read and write calls which can occur during file encryption compare to a benign application. Indicators like number of system calls, created files, opened sockets or destination IPs with which an application communicated can all be flagged during dynamic analysis and reported in order to establish a threat level for the monitored application [34].

Dynamic analysis has many applications and previous projects have explored different solutions for malware detection. In the following sections are detailed a few projects that detect malware on Linux and Windows machines.

2.8 Detection Methods

2.8.1 Automated Malware Detection

When trying to identify ransomware application, dynamic analysis has been proven highly effective in multiple studies. A number of detection mechanism have been proposed for Windows machines which explore common traits shared among different ransomware samples.

Asmitha and Vinod [35] proposed a machine learning approach to detect malware on Linux machines. In their solution, they are using system call logs generated by ELF files. The samples are collected from 226 malware samples belonging to different malware types (e.g.: virus, trojan, rootkit, worm, etc.) and 442 benign samples extracted from `/bin`, `/sbin`, `/usr/bin` from different Linux machines. After they collect the system call information, the samples are divided into training and testing datasets. Since the malware and benign dataset produce different sets of system calls, they train a model for multiple scenarios: when the features set is represented by the union or intersection of the malware and goodware features set. After training, for both cases they achieve an accuracy of 97.3% by using a Random Forest classifier.

Similar work by Shifu et al. [36] uses system call graphs for detecting malware on Android platforms. They propose a novel method of analysing Android applications called Component Traversal which uses dynamic analysis to automatically extract system calls while interacting with the application. They extract the system call logs from 3000 Android applications split equally between benign and malware. After the data is extracted, they proceed in constructing call graphs for each one of the

samples and feed them to a deep learning framework with very good results.

Yun-Chun et al. [37] introduced a different approach in detecting malware. Instead of using system call logs to classify if an application is malware or not, they use the network data from `pcap` files. Their malicious data samples include numerous Windows ransomware families like Locky, CryptoWall, CryptXXX and Cerber. The features that are being used in training are the port, IP, protocol, payload and flow behaviour. They achieve different detection rates depending on the class of malware. For Adylkuzz, Upatre and Crowti they achieved close to 100% recognition rates.

Vinayakumar et al. [38] is proposing a dynamic analysis approach for detecting Windows ransomware by leveraging API call graphs. The samples used in the project come from 7 different ransomware families and the logs were collected using `Cuckoo Sandbox`. With the collected data, they are compering the efficiency of shallow versus deep neural networks when it comes to classifying malware samples. They concluded that a multi-layer perceptron (MLP) had a high accuracy in both classifying ransomware and benign samples, but also in identifying the specific family a ransomware belongs to.

Sgandurra et al. [39] developed an automated ransomware detection for Windows platforms. They ran 582 ransomware samples belonging to 11 classes and 942 benign applications through `Cuckoo Sandbox` to collect API and network information, Registry Key operations, strings found in the executable and file operations. During feature selection they determined that some of the most impactful features are the Registry Keys Operations and API stats by using mutual information. They achieved a detection rate of 96.3%.

2.8.2 Linux Honeypots

Gomez-Hernandez et al. [40] proposed a novel way of detecting and stopping a ransomware attack on Linux through R-Locker. The tool is setting up a series of honeyfiles in the file system which act as a trap for the ransomware. Once the ransomware application tries to read the files, it will be blocked and a series of mechanisms will be triggered by R-Locker in order to completely stop the attack.

The honeyfile-based approach is efficient in detecting crypto ransomware. It has two main functionalities. One it is to block the ransomware when it attempts to access the honeyfile and the second is to notify the user about the breach in the system. The authors of the paper underline that the architecture that they are proposing can be implemented for other operating systems as well (like Windows, iOS, etc.), but they made a demonstrative implementation for Linux systems. The project uses names pipes or FIFOs to maintain both the effectivity and low consummations requirements. A named pipe has the advantage of acting in some ways similar to an entry in the filesystem and can be accessed as regular files by applications. Furthermore, named pipes also act as a channel between two processes, one being the

reader and the other being the writer. In the current use case, the anti-ransomware mechanism will create and write something to the pipe, while the ransomware will act initially as the reader. When that happens, R-Locker records the process ID (PID) of the process which accessed the pipe and notifies the user. After the user is notified, they can take the appropriate measures to stop the attack and remove the malware from the system.

The R-Locker was tested against a few samples of ransomware and while it presented successful at detecting those samples, the solution has a major drawback. The success of the solution highly depends on the placement of the honeyfiles. Some variants of ransomware might decide to encrypt only certain partitions of the filesystem in which case the ransomware would go undetected. Additionally, even if the ransomware is eventually discovered through the series of traps in the filesystem, there is no guarantee on the order in which the ransomware would access the files. There could be the case of R-Locker discovering the ransomware attack only after it finished encrypting a large percentage of the filesystem or all of it.

2.9 Data Source for Threat Detection

The MITRE organisation put together a list of attack patterns in order to help security specialists detect and prevent different types of attacks [41]. The list known as "Common Attack Pattern Enumeration and Classification" (CAPEC) has over 500 attack patterns that can be split by mechanisms of attack (e.g.: software, hardware, supply chain, etc.) or domains of attacks (e.g.: inject unexpected items, abuse existing functionality, employ probabilistic techniques).

Another source of data which is commonly used by researchers consists of multiple databases containing hundreds to thousands of malware samples of different types: Trojans, Worms, Ransomware, Bots and so on. Some of these databases are: VirusShare [42], Malshare [43], The Zoo [44], VirusBay [45], VirusSign [46], VirusSamples [47] and others. This is a valuable source of information as it helps security specialists analyse a multitude of malware binaries and keep up to date with emerging trends in security attacks.

Moreover, websites like VirusTotal [48] maintain a large database of indicators of compromise for different strains of malware. IPs or hosts which are known to belong to criminal organisations and used in performing cyber attacks are flagged accordingly and firewalls can use this information to perform sanity checks for incoming and ongoing transmissions.

All of these data sources can prove useful in building a detection mechanism for Linux ransomware, but as it will be explained in the following sections, while there has been an increase in Linux ransomware attacks, researchers have been able to capture only a very small amount of these new samples and catalog them.

2.10 Prevention Methods

After recognising the devastating impact ransomware attacks can have on both businesses and individuals alike, it is important to take precaution steps in order to preventing this type of attack from happening. A security report released at the beginning of 2021 by Google Cloud called "CISO's Guide to Cloud Security Transformation" [49] outlines some recommendations for best practises. One of the principles that they present in their report is the notion of being "risk-informed". That means taking the time to understand what assets a company or individual has that are potentially at risk to ransomware attacks. That may also include different systems a company might have, the data they are storing, their clients data and even their employees' data. In contrast to a "risk-avoidance" approach, a "risk-informed" approach realises and embraces the fact that companies can not isolate themselves from the potential of being the target of a security attack, but rather acknowledges the risks and prepares accordingly. This approach translates into knowing the most vulnerable points of a system and enforcing some security measures for them. One such example is offering security training to employees and individuals as often times they are the triggering factor in an attack chain. Opening suspicious email attachments, using unknown USB sticks and download files from suspicious sources are all common bad practices which should be avoided.

Moreover, ransomware attacks are regularly exploiting known vulnerabilities in the system. A good prevention method is to keep all the software up to date with the latest upgrades and security patches. Additionally, all data should be backed up and easy to recover in case an attack does succeed and it is advised to not pay the ransom as it might sponsor more illegal activities in the future.

2.10.1 Recovering from a Ransomware Attack

In order to completely recover from a ransomware attack there are two sides of the problem that need to be addressed. On one hand, the system needs to be cleaned up and freed from any trace of ransomware and on the other hand the data needs to be restored.

In order to restore the lost data, it is only possible if back-up mechanisms already existed before the attack [21]. If the systems had back-up, the damage from the ransomware attack is considerably reduced, becoming a matter of restoring the systems to a clean state and reducing downtime. After the system has been restored, an investigation into the attack should be conducted. The investigation is necessary to uncover weak points in the system which allowed for the attack to take place and fortify them to prevent future attacks. If there are no backup systems, it is generally not possible to decrypt the files, assuming the encryption was done correctly.

2.11 Legal and Ethical Considerations

The project aims to build a self contained solution for ransomware detection. The detection mechanism would incorporate a trained classifier in order to detect malware applications. Once the model is build and shipped with the application to the users computer, the data about the analysed applications will not leave the user's machine, avoiding most potential privacy issues. The only privacy considerations that have to be made are in regards to the data used for training and evaluating the model. If the model is trained with data not coming from users' machines, the privacy implications are negligible. Moreover, even if the data used for training does belong to the user and has been collected from the applications installed on the user's machine for example, we have to consider where the training is taking place. If the training is done locally (on the users' machines), the data does not leave the machine and there are no privacy concerns. If the data is instead sent to a centralized server for training, the data will contain detailed information about the applications installed on the machines. If we wish to commercialize the solution, since the model is small, it can be trained in a federated learning approach, which means the model can be trained across a decentralized network of machines while the data stays only within that network.

During the development of the project, we required both benign and ransomware samples. The samples were run in a contained environment on a private machine and after running they do not expose any private information. The data processing and model training were also done locally.

Chapter 3

Design and Implementation

3.1 Proposed Solution

For detecting Linux ransomware, two approaches can be considered. One can be through static analysis, but as previously discussed, this method has significant disadvantages. Moreover, the lack of a substantial amount of ransomware samples will result in a very restrictive detection system which will only be able to identify a few types of ransomware. Another option consists of building a detection system based on dynamic analysis. While the problem of having only a few samples of Linux ransomware does not go away, dynamic analysis has the advantage of being more flexible in its detection. In particular, one solution to the lack of data problem is by taking advantage of the large database samples of Windows ransomware. As formerly presented, at run-time, most of the ransomware families exhibit a similar behaviour. By using dynamic analysis, the proposed detection method can monitor a running Linux application and signal if aspects of its behaviour are similar to the behaviour of ransomware.

Therefore, the solution will have the goal of extrapolating common traits of ransomware when run on a Linux platform and compare them to the behaviour of an application to determine if it is benign. In order to train a classifier we need a substantial amount of ransomware samples, therefore, Windows samples are being used. After the model is trained, it will be tested with a few Linux ransomware samples to evaluate its performance. More details about the process of obtaining the Windows and Linux ransomware samples are discussed in Section 3.2. To be able to run Windows ransomware samples on Linux system, the `Wine` framework was used. `Wine` is an open-source project which allows applications developed for Windows platforms to be run under Unix-like systems. Since the ransomware needs to be run in a contained environment, we used the `Cuckoo Sandbox` for this task.

3.1.1 Cuckoo Sandbox

`Cuckoo Sandbox` is a state-of-the-art sandbox used by researchers to analyse malware and collect information about it. `Cuckoo` is an open source project which is

compatible to multiple platforms. It has two main components: a Guest and a Host. The Host can be either Linux or Windows and is where Cuckoo software is installed. From there samples are submitted for analyses and reports are being collected. It has an easy to use web interface which runs on the localhost at port 8080. Once a sample is submitted for analyses, it is run inside a virtual machine (VM). The VM in this scenario is the Guest and can run under one of the following operating systems: Windows, Linux, iOS or Android. For our project, we will focus on Linux Guest architecture as we want to collect behavioral information about ransomware running on Linux platforms.

Inside the Guest VM there are 2 main components running which are responsible for starting the analysed application and collecting data about it. One of them is the Cuckoo Agent which exposes a web service on the localhost of the Guest on port 8000. The web service is used by the Host to communicate with the Guest. Therefore, it enables the Host to copy the analysed application onto the VM and collect back information once the analyses is done. The other component in the Guest is the Analyser. It starts the analysed application and different auxiliary systems for monitoring the behaviour of the application. In the case of Linux Guest machines, systemtap is used to collect system call information. Systemtap is a free profiling tool used for instrumenting Linux applications. It extracts system call information on the running processes.

3.1.2 Cuckoo Base Version

The developers recommend using the version of Cuckoo that comes with pip. In this case it is version 2.0.7 and as of today none of the versions are currently supported by the developers. Moreover, Cuckoo out of the box does not have stable support for a Linux Guest VM. First of all, the display output is not working so any application with a graphical user interface will fail to run. As a result, the profiling logs are also incorrect. The recommended network configuration is only capturing packages coming between the Host and the Guest VM, failing to capture the traffic between the Guest and outside network. Moreover, the recommended installation setup for systemtap is unreliable. If the intention is to use systemtap, it has to be compiled from source and installed manually.

3.1.3 Architecture

The project consists of two parts. The first stage is the data collection which is then used for the second part, building the classifier. The proposed data collection architecture consists of adapting the Cuckoo Sandbox to allow it to run Windows ransomware samples on a Linux Guest with Wine. After the data is collected, it is passed through a pipeline and a set of features is being extracted. The features are then used to train the classifier for detecting which executable is a ransomware and which one is benign. Figure 3.1 presents a high-level overview of how the data collection architecture looks, along with its components.

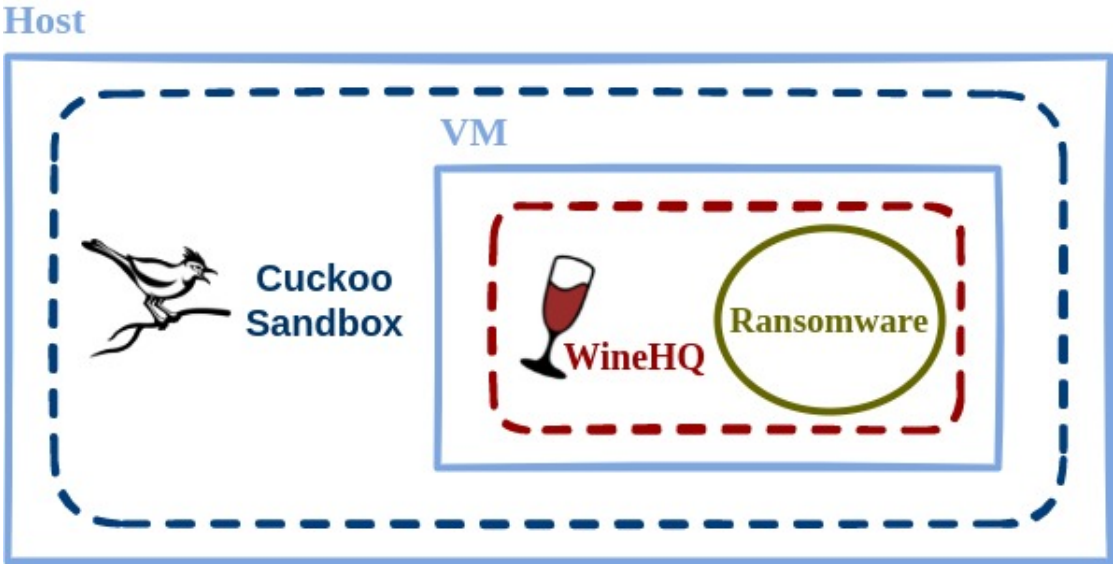

Figure 3.1: High-level overview of the data collection architecture.

The following sections offer more detail about the process of collecting and processing the data, along with the methodologies used in building the model.

3.1.4 Data Sources

The samples used for analysing (Table 3.1) are split in two categories: samples used for training and evaluation, and samples used for testing. For training, we need both benign applications and ransomware. As we need a considerable amount of ransomware samples for training, we used Windows ransomware samples. These samples were obtained from `VirusShare` [42], a malware database which requires permission from the maintainers to access. Out of the total Windows ransomware samples, we randomly selected 127 samples. For benign applications, we used 110 samples from a Windows application center [50]. These applications are a mix of applications that can be generally found on a computer: video players, music players, photo viewers, text editors, text editors, document readers, messengers, browsers, etc. The samples were verified to not be malicious using `VirusTotal`. If the samples were not identified as malicious by multiple anti-virus vendors out of the 65 vendors used for checking, then the sample is considered benign. All the Windows samples are 32-bit `exe` files.

For testing, we used 10 Linux benign applications and 3 Linux ransomware samples. The benign applications were selected from the Linux application center. The entire list of Linux applications can be find in the appendix section. For Linux ransomware samples we used 3 different samples. One of them is Bash Ransomware, a ransomware developed for research purposes [51]. Another ransomware sample that was used is Erebus ransomware, a type of crypto ransomware which targets Linux

	Number of Samples	Training	Testing
Windows Ransomware Samples	127	Yes	No
Windows Benign Applications	110	Yes	No
Linux Ransomware Samples	3	No	Yes
Linux Benign Applications	10	No	Yes
Total:			**250**

Table 3.1: Samples distribution per usage.

machines. The sample was obtained from VirusShare. The third ransomware sample that was used was a custom bash ransomware (Listing 3.1).

```
#!/bin/bash

URL="https://gist.githubusercontent.com/tazlauanubianca/
d68251f55816850ed11d742fcfb30355/raw/
9d0db33111d62a1fea30b0f8e2057ffd23cca2b5/key"
wget $URL

KEY=$(cat key | tr -d '\n')
if [ "$#" -ne 1 ]; then
  # encode
  args=""
else
  # decode
  args="-d"
fi

for f in $(find $HOME/Downloads -type f); do
  openssl aes-256-cbc $args -k $KEY -in $f -out $f.enc
  mv -f $f.enc $f
done
```

Listing 3.1: Bash ransomware for Linux.

The ransomware is using an encryption key which is obtained from the server to simulate possible calls a ransomware might make to a C&C server. It uses the AES-256 encryption algorithm provided by the OpenSSL library.

3.2 Data Collection

3.2.1 System Calls Data

As mentioned before, the first step was data collection. While the default `Cuckoo` implementation uses `systemtap`, it proved inefficient when the application is run through the `Wine` framework because of the huge amount of information that it generated compared to when it is run without `Wine`. `Systemtap` fails with an error for

exceededing the size of the storage buffer. As a result, `strace` was used to collect system call information, as an alternative to `systemtap`.

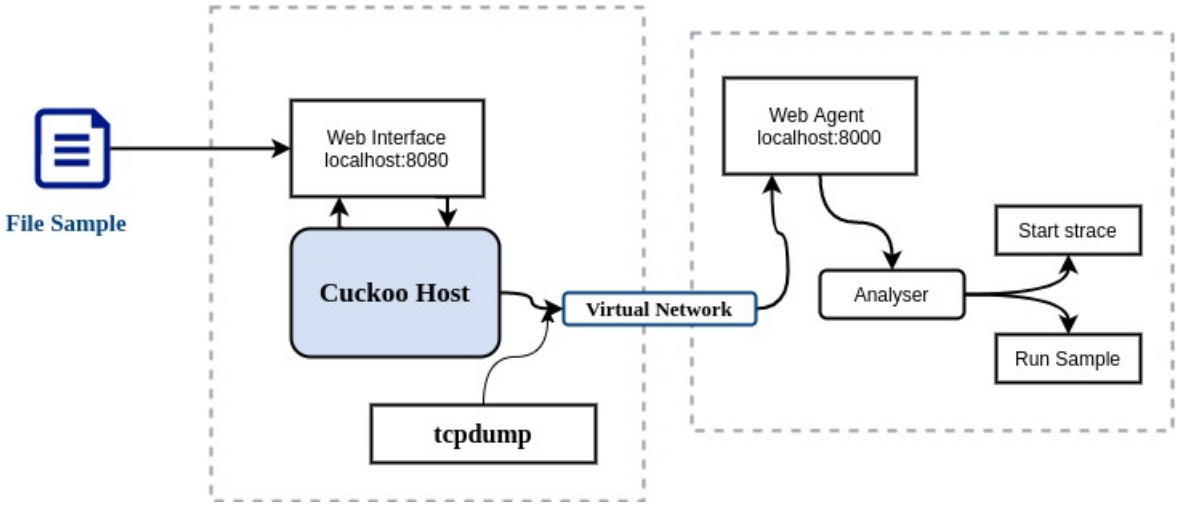

Figure 3.2: Cuckoo architecture with Wine.

Since the `Cuckoo` architecture for Linux Guests is designed to run Linux applications and we wanted to run Windows applications through `Wine`, all the applications were wrapped in a `bash` script. The code in Listing 3.2 is an example for a `bash` script used for running `WinTabber.exe`, which is a Windows benign application. On an initial run, it was observed that during the execution, no visual output was present in the Guest VM. Since the VM is started under `root` user, but the execution happens under the `cuckoo` user, the `DISPLAY` environment variable is not set. As a result, the wrapper script had to have an additional first line for exporting the value of the `DISPLAY` variable. Without it, not only would the application not be displayed, but if the application has a graphical user interface (GUI), it would not execute correctly. One example is the `Firefox` browser. If the process starting `Firefox` can not access the display, the execution will terminate with an error.

```
export DISPLAY:=0.0
wine /home/cuckoo/Desktop/goodware/WinTabber.exe
```

Listing 3.2: Bash script for executing Windows ransomware through Wine.

Another `bash` script is responsible for iterating through all the samples of executable files, both malware and goodware, create the wrapper scripts and submit them for analysis. Listing 3.3 is a code snippet of the `Cuckoo Sandbox` which was modified to allow to profile the wrapper scripts inside the Guest VM using `strace`. All the samples were copied to the VM along with multiple files (pictures, archives, documents) to simulate a normal working machine. The state of the VM was reset every time between runs and each sample was analysed for 2 minutes.

```
def start(self, path):
```

```
  os.chmod(path, 0o755)
  return self.execute([
    "su", "cuckoo", "-c",
    "strace -y -D -o  /tmp/strace.log -f sh " + path.strip()
  ])
```

Listing 3.3: Code for running the application.

The `strace` command was used with the `-y` flag to enable the display of file path names in the logs, instead of file descriptors. Additionally, the `-D` flag was set to run the tracer as a grandparent process of the profiled process and not as a parent. This impacts the effects `strace` has in polluting the logs, reducing the noise in the data. Lastly, `strace` was given a file path where to store the output logs. The given file will then be collected by the Host through the web service.

After the samples were run, the Host collects the `strace` information. The format of the output produced by `strace` can be seen in Listing 3.4. The sample logs belong to the `ZoomInstaller.exe` application.

```
epoll_wait(12<anon_inode:[eventpoll]>,
    [{EPOLLIN, {u32=90, u64=90}}], 128, 760) = 1
read(32<pipe:[41064]>
    "\25\0\0\0\0\0\0\0\0\0\0\0\210\0\0\0\0\0\0\0
    \0\0\0\0\0\0\0\0\0\0\0\0"..., 64) = 64
close(113</home/cuckoo/.wine/drive_c/users/cuckoo/
    ApplicationData/Zoom/zoom_install_src/nydus.dll>) = 0
stat("/home/cuckoo/.wine/dosdevices/c:/users/cuckoo/
    ApplicationData/Zoom/zoom_install_src/nydus.dll",
    {st_mode=S_IFREG|0664, st_size=1586472, ...}) = 0
unlink("/home/cuckoo/.wine/dosdevices/c:/users/cuckoo/
    ApplicationData/Zoom/zoom_install_src/nydus.dll") = 0
write(37<pipe:[41995]>,
    "\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
    \0\0\0\0\0\0\0\0\0\0\0\0"..., 64 <unfinished ...>
    <... read resumed>
    "\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0
    \0\0\0\0\0\0\0\0\0\0\0\0"..., 64) = 64
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
```

Listing 3.4: Strace output for ZoomInstaller.

In the case of `ZoomInstaller` we can notice for example how it is accessing its own installation files. If we take a look at the `strace` logs produced by one of the ransomware samples instead (Listing 3.5), it can be observed how it is accessing files that are not related to the process. In this case, it is accessing the file `1984 - George Orwell.pdf` located in `/home/cuckoo/Downloads/`. The ransomware is opening the file and is performing a series of `write` operations.

```
openat(AT_FDCWD, "/home/cuckoo/.wine/dosdevices/z:
    /home/cuckoo/Downloads/1984_-_George_Orwell.pdf",
    O_RDONLY|O_NONBLOCK) =
    106</home/cuckoo/Downloads/1984_-_George_Orwell.pdf>
fstat(106</home/cuckoo/Downloads/1984_-_George_Orwell.pdf>,
    {st_mode=S_IFREG|0664, st_size=2671659, ...}) = 0
write(23<pipe:[38725]>,
    "\0\0\0\0\0\0\0\01\0\0\0\0\0\0\0\0\0\0\0
    \0\0\0\0\0\0\0\0\0\0\0\0"..., 64) = 64
read(8<pipe:[38725]>,
    "\0\0\0\0\0\0\0\01\0\0\0\0\0\0\0\0\0\0\0
    \0\0\0\0\0\0\0\0\0\0\0\0"..., 64) = 64
rt_sigprocmask(SIG_SETMASK, [], NULL, 8) = 0
stat64("/home/cuckoo/.wine/dosdevices/z:
    /home/cuckoo/Downloads/1984_-_George_Orwell.pdf",
    {st_mode=S_IFREG|0664, st_size=2671659, ...}) = 0
        . . .
write(7<pipe:[37300]>,
    "-\0\0\0\0\0\0\0\0\0\0\01\0\0\0\0\0\0\0\0\0\0\0
    \0\0\0\0\0\0\0\0"..., 64 <unfinished ...>
    <... epoll_wait resumed>
    [{EPOLLIN, {u32=7, u64=7}}], 128, 1854) = 1
```

Listing 3.5: Strace output for ZoomInstaller.

Moreover, the ransomware is making a call to the `sendmsg` system call, as can be seen in Listing 3.6. During the analysis, the ransomware samples generally made multiple communication calls (possibly for exchanging encryption keys) with C&C servers.

```
sendmsg(21<socket:[37299]>, {msg_name=NULL, msg_namelen=0,
    msg_iov=[{iov_base="l\0\0\0", iov_len=4}],
    msg_iovlen=1, msg_control=[{cmsg_len=20,
    cmsg_level=SOL_SOCKET,
    cmsg_type=SCM_RIGHTS,
    cmsg_data=[106<
    /home/cuckoo/Downloads/1984_-_George_Orwell.pdf>]}],
    msg_controllen=20, msg_flags=0}, 0) = 4
write(23<pipe:[38725]>,
    "\0\0\0\0\0\0\0\0\1\0\0\0\1\0\0\0\0\0\20\0
    \0\0\0\0\0\0\0\0\0\0\0\0"..., 64) = 64
    . . .
read(8<pipe:[38725]>,
    "\0\0\0\0\0\0\0\0\1\0\0\0\1\0\0\0\0\0\20\0
    \0\0\0\0\0\0\0\0\0\0\0\0"..., 64) = 64
rt_sigprocmask(SIG_SETMASK,
```

```
    [HUP INT USR1 USR2 ALRM CHLD IO], NULL, 8) = 0
recvmsg(6<socket:[38641]>,
    {msg_name=NULL, msg_namelen=0,
    msg_iov=[{iov_base="1\0\0\0", iov_len=4}],
    msg_iovlen=1, msg_control=[{cmsg_len=16,
    cmsg_level=SOL_SOCKET, cmsg_type=SCM_RIGHTS,
    cmsg_data=[17<
        /home/cuckoo/Downloads/1984_-_George_Orwell.pdf>]}],
    msg_controllen=16, msg_flags=MSG_CMSG_CLOEXEC},
    MSG_CMSG_CLOEXEC) = 4
fcntl64(17</home/cuckoo/Downloads/1984_-_George_Orwell.pdf>,
    F_SETFD, FD_CLOEXEC) = 0
```

Listing 3.6: Strace output for ZoomInstaller.

In other logs it can also be noticed that the ransomware is changing the extension of the file after it is done encrypting. In the current example (Listing 3.7), the file name is changed from `1984 - George Orwell.pdf` to `1984 - George Orwell.pdf.EnCiPhErEd`.

```
rename("/home/cuckoo/.wine/dosdevices/z:
    /home/cuckoo/Downloads/1984_-_George_Orwell.pdf",
    "/home/cuckoo/.wine/dosdevices/z:
    /home/cuckoo/Downloads/
    1984_-_George_Orwell.pdf.EnCiPhErEd") = 0
```

Listing 3.7: Ransomware changes file extension after encryption.

3.2.2 Network Traffic

One of the aspects that is taken into account when building the model is the network traffic. The usage of network communication by a ransomware can vary depending on the family variant it belongs to. Generally, ransomware needs little communication to the C&C servers, mostly for receiving encryption keys and some configuration files. Moreover, most of the ransomware strains will encrypt their data, so extracting the payload of the communications initiated by the ransomware might not yield interesting results. As a result, the more relevant information for ransomware detection can be the number of `TCP/UDP` communications and the destination of the communication.

By default, `Cuckoo Sandbox` offers the functionality of collecting network data, but for the Linux Guest setup, under the indications provided by developers, it is only possible to collect the communication between the Host and the Guest during the analyses process. For this reason, `Cuckoo`'s network configuration between Host and Guest had to be adapted. The Guest was assigned a static IP and a gateway to be able to communicate with the Host and with the outside network. The `VirtualBox` VM was configured with a bridge interface through which the network traffic was routed. On the Host machine, `Cuckoo` was configured to sniff on all the incoming

and outgoing traffic routed through the bridge interface of the VM. The configuration excluded all communication that was just between the Host and Guest as those represent communication initiated by Cuckoo during analysis and it is not needed for the data collection.

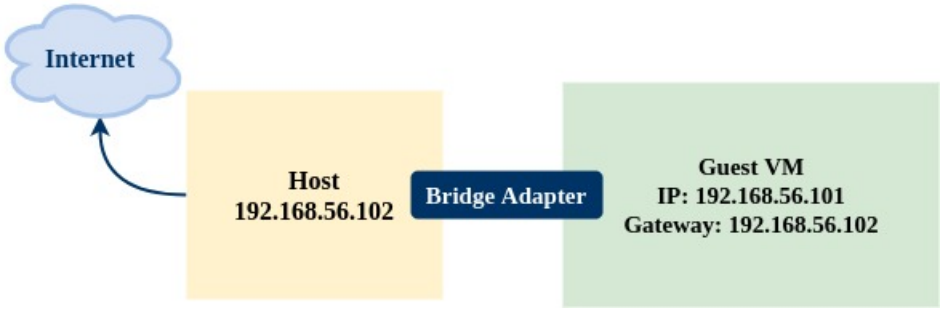

Figure 3.3: Network architecture of the Guest VM and Host.

It is important to mention that Cuckoo intercepts all communications coming from the VM, not only the one generated by the application that is being analysed. As a result, some noise is considered to be present in the collected traffic which might come from the network traffic generated by the Linux system. Cuckoo uses tcpdump to collect the pcap files for each sample submitted for evaluation. After the analysis is completed, Cuckoo parses the files to generate a report in which it also includes the TCP and UDP transmissions. That data was collected and represented different features in the final dataset. Therefore, the extracted features from the pcap files are the number of TCP connection, the number of UDP connections and then for each unique IP that was ever encountered in a transmission during any analysis, a value of 1 if a specific sample had a communication with that IP or a value of 0 otherwise.

On average, each goodware sample contacted on average 10 different IP address, while the ransomware samples contacted a similar average number of 11 unique IP. In the following tables present the top 10 contacted IP addresses for both goodware and badware.

It is important to notice that among the IP addresses that have been contacted, not all of them are relevant. For example, the top 4 contacted IP address for both malware and goodware are 8.8.8.8, 91.189.89.199, 185.125.188.12, 109.202.202.202 and 224.0.0.251. This is to be expected since 8.8.8.8 is the address of the Domain Name System (DNS) service which was set on the VM. The other 3 addresses belong to the Internet service provider and Linux system updates. As a result, during the process of feature selection, the features represented by these IP addresses will be removed.

3.3 Data Processing

After all the samples of ransomware and goodware were run through the architecture described in the previous sections, the data was filtered and parsed. At this

Destination IP	Frequency
91.189.89.199	246
8.8.8.8	246
109.202.202.202	244
185.125.188.12	244
224.0.0.251	239
255.255.255.255	211
151.101.194.49	194
8.43.85.29	150
91.189.91.38	131
91.189.91.39	114

Table 3.2: IP communication distribution.

stage we are only interested in extracting all the features that are potentially relevant for the model training, without doing any feature selection.

The network data which was stored in `pcap` files was automatically parsed by `Cuckoo` and the results were saved in `json` report files. The report files needed to be then parsed to extract only the information needed for the project. For each one of the samples it was extracted the number of `TCP` and `UDP` connections along with the IPs that the application communicated with. The parsing was done with a `Python` script and by using the `json` library offered by `Python3`.

From the `strace` files we extracted the rest of the features. The features that were of interest were: system calls, created files, deleted files, written files, read files and changed permissions. For each system call in the `strace` file we extracted the name and created a set with all the system calls. If the system call was `open_at` then the flags given as parameters were checked to see if the file was accessed for reading or writing operations. If the system call `unlink` was used, the resource sent as parameter to the call was recorded as deleted. A similar approach was done for the `chmod` and `create` system call. To create the set of features we created the union set composed of all the registered system calls, deleted files, created files, read files, written files, and resources that had their permissions changed. The total number of columns generated for the dataset is 7603 with the distribution among groups of features as presented in Table 3.3.

Once the features were extracted and saved in a `csv` file, we moved to the second part of the project which consists of analyzing the data and building a classifier.

3.4 Data Analysis and Feature Selection

Before building and training the machine learning model, we first look at the properties and distribution of the data. The input dataset used for training has a total of 7603 features and a total of 237 samples distributed between 127 ransomware

Group of Features	Type	Number of Features
Name	str	1
Label	int	1
Test Sample	int	1
System Calls	str	166
Number of UDP Transmissions	int	1
Number of TCP Transmissions	int	1
Destination IPs in UDP Transmissions	str	35
Destination IPs in TCP Transmissions	str	81
Deleted Files	str	363
Written Files	str	3082
Read Files	str	3861
Perms	str	10
Total:		**7603**

Table 3.3: Distribution of features.

	Average Variance
System Calls	60331457.94
Destination IPs	0.449
Written Files	0.00438
Read Files	0.0090
Deleted Files	0.0023
Permission Changes	0.0020

Table 3.4: Average variance of features split by class.

samples and 110 benign samples.

First, we looked at the most variant features in the training dataset. To compute the variance we first compute the mean of the given feature $\bar{x}$. The variance is given by squaring the difference between the individual feature values and the mean summed across the whole feature, divided by the number of feature values:

$$S^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}$$

The only features that are used from the total feature are the ones with a variance above $0.1 \cdot (1-0.1)$ (variance of Bernoulli distribution with $p = 0.1$), as the remaining ones do not carry meaningful information across multiple samples for the model. Out of the total 7601 total initial features, only 168 are used in the final dataset for training. The numbers here are computed across the whole dataset – at training time, this is computed for each cross-validation fold, so it most likely differs slightly per fold. Features with low variance indicate that the specific feature is found in very few samples. As a result, if features with low variance are kept in the training process, there are greater chances of the model overfitting on those features. Across the whole dataset, the first 90 features sorted by biggest variance consist of system

calls, with the number of UDP transmission on position 91 and for TCP on 93.

Ranking	Variance	Feature
1	3679477790.132707	read
2	3167661738.114227	rt_sigprocmask
3	1577313961.7366161	write
4	547948899.9293917	stat64
5	532258494.56874794	epoll_wait
6	184376270.18586767	recvmsg
7	184344099.6971995	writev
8	47432726.85200021	mmap2
9	20334514.491623484	getdents64
10	18565467.530203488	openat
11	10073733.618472822	close
12	9241755.965532588	getpid
13	8691817.748865034	mprotect
14	6485285.72675319	fstat64
15	6476471.023553919	lstat64
16	4485466.817746443	statfs64
17	3715479.8398760892	sched_yield
18	2174529.602734605	poll
19	1864965.2939521798	futex
20	466569.9504352935	epoll_ctl
21	395486.6325197172	_llseek
22	206015.145364881	fstat
23	191277.31221848348	stat
24	141889.8586764942	_newselect
25	122019.21437091635	sendmsg

Table 3.5: Top 25 features based on their variance score.

Next, we looked at the Principal Component Analysis (PCA) and T-Distributed Stochastic Neighbouring Entities (t-SNE) plots to better understand the distribution of the data.

PCA is a method used to reduce the number of dimensions of a dataset while preserving the most information. The number of dimensions is given in a dataset by the number of features. In our case, this results in 168 dimensions which is not easily representable for human inspection.

In Figure 3.4 we can see the training data with the system calls features projected on the first 3 components of PCA. We can see a separation between the classes across several axes, however, the classes still have large overlap which will make them hard to classify. This is explained by the fact that a lot of applications, benign or malware, have similar traits. One example is the case of network communication which uses encryption libraries, even though they are not used for encrypting files like in the

case of ransomware. Further more, all the training samples were run through `Wine`, introducing more similarities between them.

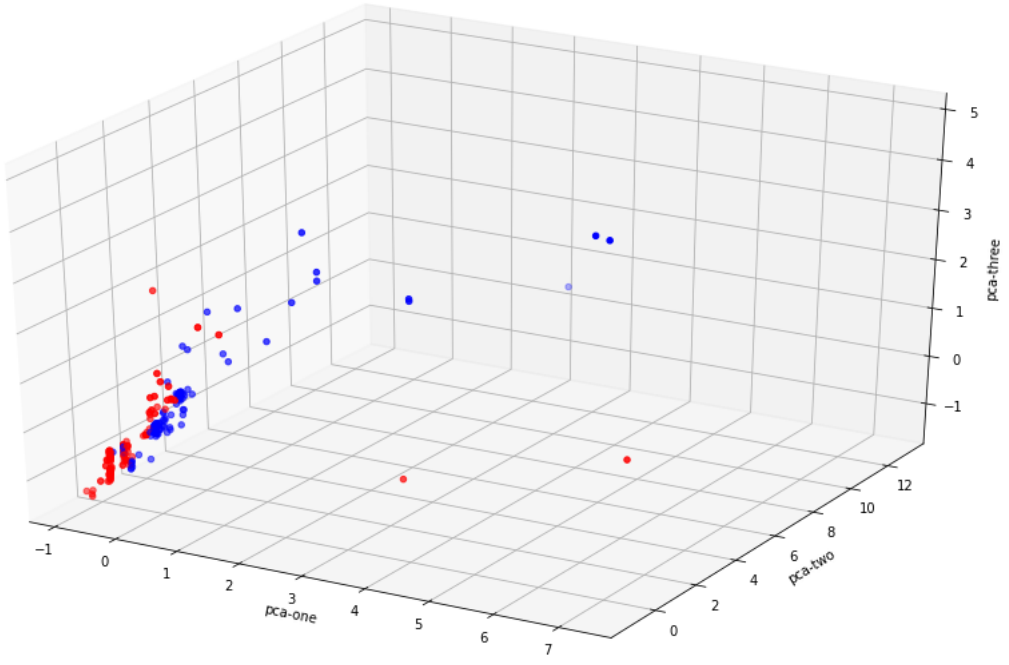

Figure 3.4: 3D plot of the first three components of PCA.

t-SNE is another method used for data visualisation which can give us more information about the distribution of the samples. In the Figure 3.5 we can see a projection of the training data set onto the first 2 dimensions of the `TSNE` space computed from the data. The points belonging to two different classes which are close to each other indicate that the points are close in the original space. Similar to the information offered by the `PCA` plot, we can see how some of the data samples are closely related. This can be explained, for example, by the fact that the samples did not execute properly. In the case of ransomware, some of them might have detected that they are being run in a VM and not executed. Another problem that might have occurred during analysis is that the samples could not be run through `Wine` which resulted in similar trace logs.

After examining the variance and distribution of the data, in order to avoid overfitting, only the system calls information and number of `UDP` and `TCP` transmissions was used for training.

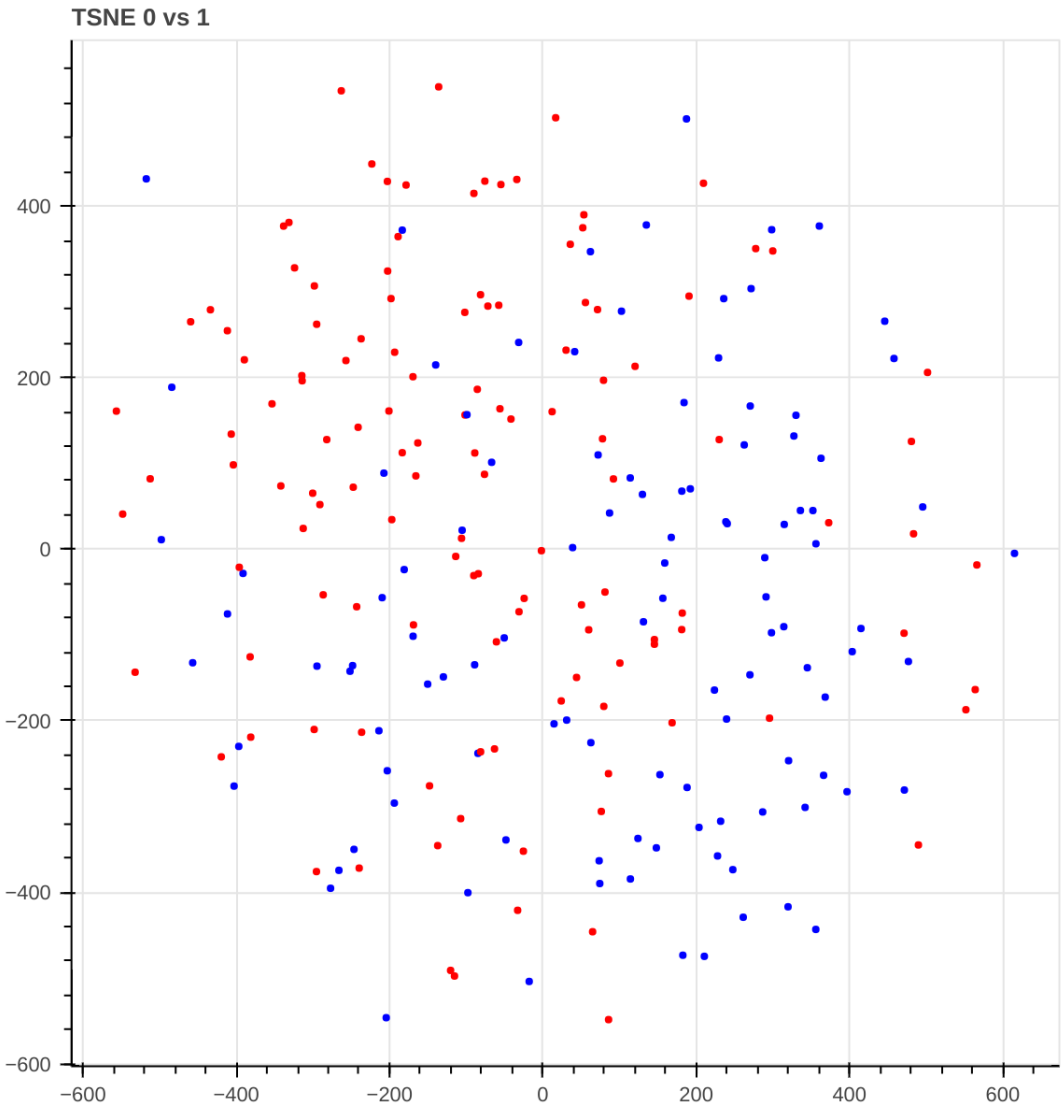

Figure 3.5: TSNE plot across 2 axes.

3.5 Building the Machine Learning Model

For training, the first step was to remove the name of the files from the input data as it is not one of the relevant features used in training. Then, the label column in the input dataset is split off. At this stage the data is split between the training and testing dataset based on the testing label. As previously mentioned, only the system calls information and number of `TCP` and `UDP` transmissions was kept as features for training. This resulted in 168 features: 166 system calls and 2 numbers representing the number of `UDP` and `TCP` communications. When referring to training data, we refer to the system calls and number of `UDP` and `TCP` features of the training set.

The model is built using the training dataset using `Python 3` and `scikit-learn` library. We chose the following models for training: Logistic Regression, Linear Kernel Support Vector Machine, RBF Kernel Vector Machine, K-Nearest Neighbors (KNN)

Classifier and Random Forest Classifier.

The Logistic Regression model has the advantage that it is simple and can be used to determine for each feature what coefficient it was assigned and as a result, its relevance when making the classification. It has a linear number of parameters with respect to the number of features. It creates a linear separation hyper-plane in the dataset.

As opposed to Logistic Regression, SVMs try to maximise the minimum distance to the separated hyper-plane. With an RBF Kernel, conceptually differs from the Linear SVM by first applying an exponential feature transformation. In this way it can represent nonlinear separation hyper-planes.

In the case of KNN, for each data point we want to predict the label of, it looks at the K nearest neighbors and the majority label of those neighbours will determine the label of the data point.

The Random Forest classifier is a more complex algorithm that builds multiple decision trees to perform the classification and picks features at random.

Because of the small training dataset, a 5 fold cross validation approach was used. This also helps in having a better estimate of the test error. The performance of each model is discussed and compared in the following chapter.

Chapter 4

Evaluation and Experiment Results

4.1 Metrics Used in Evaluation

During the evaluation stage, multiple metrics were computed to determine the performance of the models. The metrics that were computed are accuracy, precision, recall and F1-score.

	Positive Label (Ransomware)	Negative Label (Benign)
Positive Prediction	True Positive (TP)	False Positive (FP)
Negative Prediction	False Negative (FN)	True Negative (TN)

Table 4.1: Confusion matrix.

To better understand how the metrics are computed, we need to define the False Positive Rate (FPR) or false alarm ratio and True Positive Rate (TPR). The False Positive Rate is defined as the ratio between the false samples that were predicted as positive and the total number of samples predicted as negative. The True Positive Rate is computed as the ratio between the true positive samples, samples that were predicted as true and are actually true, and the sum of true positive samples and false negative, so the total amount of positive samples in the dataset, regardless of how they were predicted. It is important to mention that in our current problem, a sample is positive if it is a ransomware sample and negative if it is benign.

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

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

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

$$Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

Recall is equivalent to the TPR and it is a good estimator for the number of correctly predicted samples the model predicted. In the given context, it lets us know how many ransomware samples we have not identified. On the other hand, precision is the ratio of the true positive samples over the sum of true positive and false positive samples.

Considering the high cost of not detecting a ransomware application, we are more interested in having a low false negative count even at the expanse of having more false positives. That is why for the given task, recall plays a bigger role in determining the performance of the model that precision. Oftentimes, when tuning the machine learning model, improving recall might come at the expense of the precision score [52]. One method to avoid this is by using the `F1 score`. The `F1 score` is an indicator of accuracy based on the `recall` and `precision` scores.

The `Macro F1-score` or `macro-averaged F1 score` is computed by computing the `F1 score` for every individual class and then averaging them. This method is proffered when we have unbalanced classes. In our case, for the testing dataset the benign class has 3 times more samples than the ransomware class, making the dataset unbalanced.

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$\text{Macro-averaged F1-score} = \frac{1}{N} \sum_{i=0}^{N} \text{F1-score}_i$$

4.2 Models Results and Comparison

We used 168 features consisting of the extracted system calls logs and number of `TCP` connections and number of `UDP` connections. The file based features and the

IP features have proven to be too sparse and models trained on them seemed to give too much importance to one specific IP or one specific file access. This leads to overfitting and as a result we reduced the number of features based on the variance threshold.

4.3 Experiments on Linux Ransomware Samples

The training for all the classifiers was done using a 5-fold cross validation approach. It is a technique which facilitates training on a small number of samples. We trained the classifier on different models and the evaluated them on the testing dataset. Out of all the classifiers, the Logistic Regression model performed the best on the testing set identifying all the ransomware samples.

	Logistic Regression	**Linear SVC**	**Random Forest**
Accuracy	92.81% ± 2.21%	91.98% ± 2.83%	95.78% ± 2.68%
Precision	93.13% ± 2.12%	92.48% ± 2.60%	95.91% ± 2.77%
Recall	92.70% ± 2.25%	91.97% ± 2.67%	95.76% ± 2.51%
F1-Score	92.75% ± 2.22%	91.92% ± 2.83%	95.76% ± 2.67%

Table 4.2: Classifier model cross-validation metrics.

	RBF SVC	**3-NN**
Accuracy	94.08% ± 3.67%	86.91% ± 4.57%
Precision	94.23% ± 3.74%	87.03% ± 4.58%
Recall	94.05% ± 3.52%	87.09% ± 4.58%
F1-Score	94.05% ± 3.66%	86.88% ± 4.57%

Table 4.3: Classifier model cross-validation metrics.

	Precision	**Recall**	**F1-score**
Macro Avg.	0.75	0.85	0.75
Weighted Avg.	0.88	0.77	0.79

Table 4.4: Logistic regression average scores in the test dataset.

The Logistic Regression model had some problems identifying benign samples and misclassifying them as malware. As seen in Table 4.5, 3 benign samples were identified as ransomware. This can be explained by the fact that these samples had a higher network traffic than the rest of the submitted samples and during training the percentage of benign application with high network traffic was small. In comparison, the percentage of ransomware with higher network traffic was bigger, which might have led the model to classify these samples as ransomware. Even so, Logistic Regression outperformed the classification capabilities of all the other models.

Sample	Correct Label	Predicted Label
nautilus	0	0
ubuntu-software	0	1
transmission-gtk	0	0
software-update-gtk	0	1
cheese	0	0
rhythmbox	0	0
thunderbird	0	0
evince	0	0
eog	0	0
firefox	0	1
hagz	1	1
erebus	1	1
bash-ransomware	1	1

Table 4.5: Prediction labels for testing dataset with logistic regression.

We can see from the results in Table 4.2 and Table 4.3 that other classifiers have a better accuracy score, but that does not result in a better performance. For example, if we look at the macro average F1-score in the Table 4.6 of the Random Forest we can see it has a very low score. That is because the model highly classified the samples as malware, missing almost all benign applications. Similarly, k-Nearest Neighbors model for k = 3 also tended to classify samples as malware, having a very high false positive rate. In the case of RBF SVC, it classified all the samples as benign, the model not managing to learn how to identify ransomware samples.

	Macro Average F1-Score	Macro Average Recall
Logistic Regression	0.75	0.85
Linear SVC	0.61	0.75
Random Forest	0.38	0.60
RBF SVC	0.43	0.50
3-NN	0.54	0.55

Table 4.6: Macro average F1-score and recall for every classifier.

After training, it is interesting to examine which system calls the models considered to be the most relevant in the classification process. Figure 4.1 presents the top 50 system calls according to the best performing model, the logistic regression.

4.4 Limitations

The main limiting factor in developing an automatic detection for Linux ransomware is the lack of enough samples to train the classifier. While the Linux ransomware attacks are becoming more frequent and we are aware of more strains that are targeting Linux systems, researchers have yet to capture enough samples for analysis.

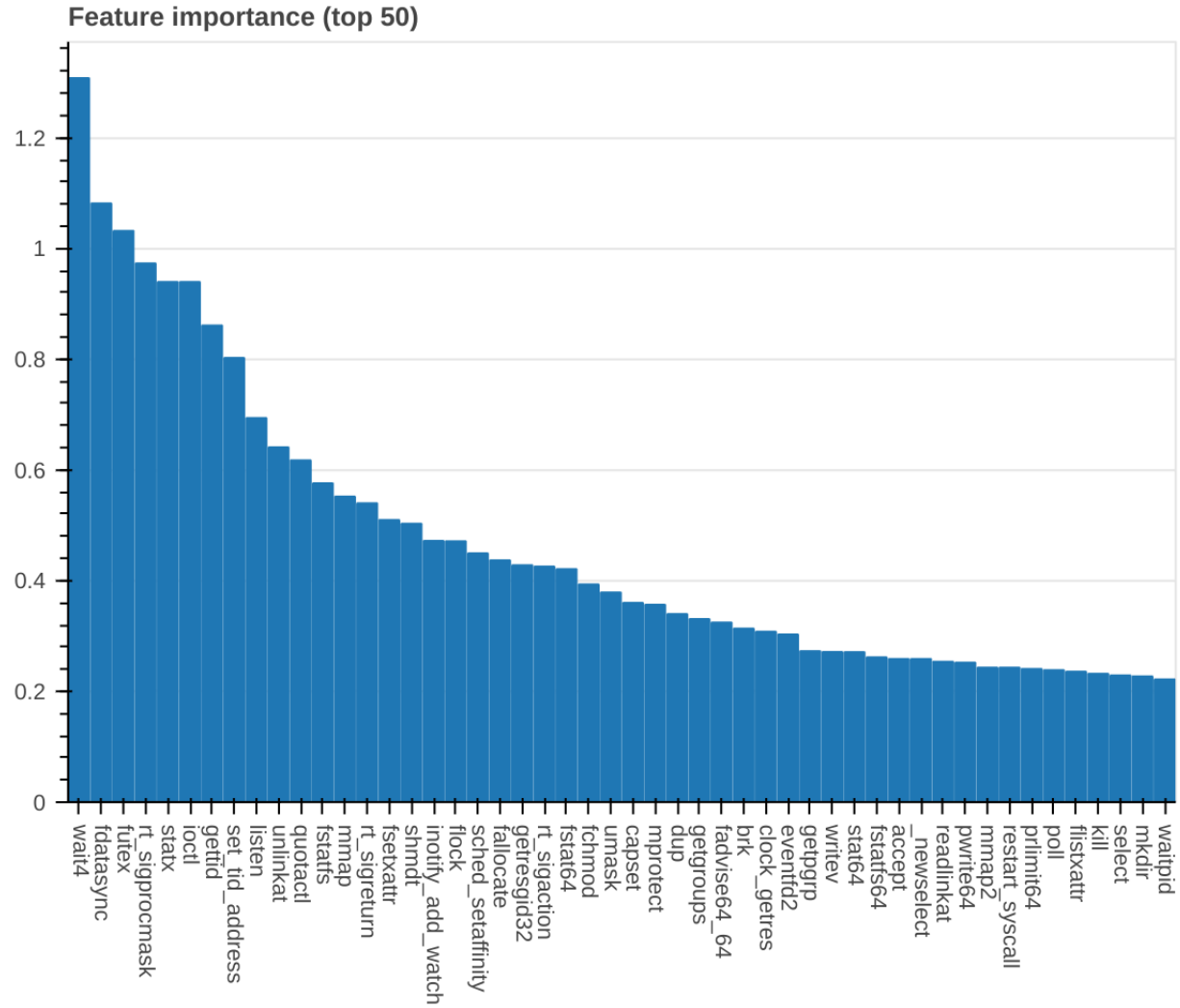

Figure 4.1: Top 50 system calls according to the coefficients given by logistic regression model.

This process is strenuous and takes a considerable amount of resources. Honeypot machines might be used for this task [53], but there is still the problem of identifying the malware type. The model's performance is strongly related to the amount of data it gets for training. Too few data samples might result in overfitting, the model memorizing only the training set and not being able to learn to meaningfully distinguish between ransomware and goodware.

The quality of data is also a big limiting factor. If ransomware families evolve and branch away too much from the ransomware strains used in training, the classifier will no longer be able to correctly identify those new ransomware strains. With the hackers constantly changing and adapting their attack techniques, new ransomware samples will need to be periodically collected and the model retrained.

By trying to use Windows ransomware instead of Linux ransomware, we try to over-

come the problem of sample scarcity. Even so, since the training is done on a different type of samples than the one used for testing, it increases the risk of overfitting and misclassifying the testing samples. Moreover, we still require Linux ransomware samples for properly testing the solution. With more samples, we can get a better view of the performance of the model.

Another aspect that can limit the performance of the model is the methodology used for feature extraction. In the current implementation, the features were extracted according to domain knowledge on general traits of ransomware gained from background reading. The features can be represented in a different form or different features might be extracted which will influence the performance of the model. The method of doing feature selection is also impacting the classification capabilities of the model.

4.5 Improvements and Future Work

The first improvement that can be considered for the project is create a bigger training dataset considering there are more Windows ransomware sample and Windows applications available. Moreover, we need to consider the noise that is added in the `strace` logs by using `Wine`. A further improvement that can be made is by trying to reduce this noise by filtering the log data. One approach might be running `Wine` and profiling it with `strace`. We can then attempt to filter the sample log data based on the "baseline" data.

As mentioned before, the `systemtap` profiling tool offered by default in `Cuckoo` proved ineffective in our current setup and `strace` was chosen instead. Part of the future work can be experimenting with other profiling tools as well, like `perf_events` and `eBPF`. This will impact the type and quality of data used for training and as a result, the performance of the classifier [54]. Moreover, all sample applications need to be run with a profiling tool, which reduces the user-perceived "speed" at which they run. An additional improvement benefiting the project would be to find more efficient ways of profiling the applications at inference time.

The natural next step in the project in developing an end-to-end detection system which will be capable of recording `strace` logs, parse them and run the sample through the classifier. Currently these components are independent and need to be run manually.

Additionally, a different method for creating the dataset can be used. One example of such approach is considering creating a synthetic dataset based on statistical data about Windows ransomware. This approach has the disadvantage that Windows system calls would have to be mapped to their Linux counterparts which might not be possible. Another approach would be to continuously collecting Linux samples and then using the rest of the pipeline provided by this project to build a new classifier.

Chapter 5

Conclusion

5.1 Summary and Contributions

Criminal organisations are trying time and time again to create more evolved attacks targeting both big corporations and individuals. With ransomware attacks doubling in numbers in the last year, researchers continue their race in building superior detection tools.

Historically, Unix systems were not the main target of ransomware attacks, since their percentage of total running systems was small. Instead, hackers would focus their efforts on more profitable systems like Windows. Once Linux and other Unix operating systems have increased in popularity, so did the interest of criminal organisations to attack them. Not only ransomware, but other types of malware, are starting now to migrate from other other operating systems to Linux, making it essential for security experts to develop detection systems for the Unix ecosystem.

Generally, no matter the approach used in detection, either by doing static or dynamic analysis, it still requires huge amounts of data samples. Even with increasing efforts to collect and analyse Linux malware, we are lacking sufficient data in order to build a proper detection system. Waiting for more Linux ransomware samples to be collected would take a long time, time in which more machines would fall victim to this type of attack. In its place, we propose a different approach which requires less samples and can be constantly improved when ransomware samples become available. With more malware families migrating from Windows systems to Linux, this approach might prove effective in detecting other types of attack as well. The project has the following main contributions:

- Building a system for collecting ransomware information for Linux platforms which leverages the high availability of Windows malware samples.
- A parsing pipeline which extracts different features out of the ransomware samples logs.
- An analysis of the most relevant features when detecting Linux ransomware.

- A dataset with information about different ransomware samples run on a Linux machine.
- A machine learning model capable of classifying ransomware and benign samples.

The project introduces a system for safely collecting and parsing malware data produced by Windows malware. The system analyses the behaviour of different classes of malware on a Linux system even with lack of Linux samples. This can be used to extrapolate common features of the analysed class and facilitate detection of previously unknown Linux malware strains.

We collected a training and testing dataset to be used for Linux ransomware detection. From the collected analyses, multiple features were extracted and interpreted. For each sample, we extracted information like number of invocation of each system call, contacted IP addresses, deleted, written and read files. This information was analysed to determine which features were the most relevant for the training process and determined that the distribution of the system calls has the biggest impact in labeling the samples. The dataset can be found on Google Drive [55].

With the newly created dataset, we trained classifier models and tested them for detecting Linux ransomware samples. The models were trained using different approaches and some yielded relatively good results on the testing dataset.

While all the ransomware samples were accurately predicted, the size of the testing set is too small to make a definitive conclusion, but it serves as an indicator that the proposed approach can prove effective. Considering the trade-offs of waiting for multiple Linux ransomware samples, the project offers a novel proof-of-concept solution which can be used until more samples become available.

Appendices

Appendix A

List of Windows Applications

- 7z1900.exe
- abinstall.exe
- AbiWordPortable_2.8.6.paf.exe
- Ambiyans_setup.exe
- audio_converter.exe
- avast_free_antivirus_setup_online.exe
- BMICalculator.exe
- BootVis-Tool.exe
- CalmeSetup.exe
- converber.exe
- CoolNovo_Setup.exe
- CoolPlayerPlusPortable_2.19.6.paf.exe
- copernicagentbasic.exe
- CrazyTaxiRacers.exe
- crosswordpuzzlemaker_setup.exe
- dfsetup222.exe
- diffractor-setup.exe
- draw.io-13.9.9-windows-installer.exe
- drivegleam_setup.exe
- dropbox_112.4.321_offline_installer.exe
- D-TPV_2_setup_32_64_bits.exe
- emailstripper.exe
- ERSetup440435_eng.exe
- EyeDefenderSetup.exe
- facelift_v075.exe
- FenrirFS245-setup.exe
- file.exe
- Free3gpVideoConverter_2.2.exe
- FreeArc-0.666-win32.exe
- free-avi-player.exe
- freeclef.exe
- freeexcelviewer.exe
- free_hard_drive_data_recovery.exe
- freeisocreator.exe
- Gassmann_1_3.exe
- GranolaPersonalInstall.exe
- homesecuritycamerasetup.exe
- HottNotes4.1Setup.exe
- Kamus_2.04.exe
- KeyBlazeTypingTutorSoftware.exe
- KindleForPC-installer-1.31.60170.exe

- kmplayer_4.1.5.8.exe
- Launchy2.5.exe
- LineInst.exe
- Listary.exe
- LostPhotosSetup1.0.1.exe
- manshutdown_1.0.exe
- map-explorer.exe
- MidiSheetMusic-1.3.exe
- mind-graph.exe
- miranda-im-v0-10-22-unicode.exe
- mp3PROAudioPlayer_v1_1_0.exe
- mpsetup.exe
- MSN_Polygamy_15.0.exe
- myCollections.exe
- MySpaceIM_Setup.exe
- OblyTile_v0.9.9.exe
- OddStormLive.exe
- OperaSetup.exe
- PDFMerger.exe
- PlagiarismCheckerX_Setup.exe
- plagiarismfindersetup.exe
- Pokedex.exe
- PopTray320.exe
- PowerPack171.exe
- qmp_5.0.121.exe
- RapidTyping_Setup_5.4_x32.exe
- reiboot.exe
- Scratch_3.23.1_Setup.exe
- servicesoptimizer.exe
- setup_avp.exe
- setup.exe
- setup_full.exe
- SetupSSuiteOmegaOfficeHD.exe
- setupvb.exe
- SHAREit-KCWEB.exe
- SI306SETUP.EXE
- SmartAssist_v147_IT.exe
- Sonma-Typing-Expert-2.01.0000.exe
- SwitchAudioFileConverter.exe
- Synkron-1.6.2-win32.exe
- SystemMechanicStd_DM.exe
- SystemMonitorIISetup.exe
- testgeneratorstandard.exe
- tomighty-0.6-install.exe
- trillian-v6.4.0.5.exe
- tsetup.2.7.4.exe
- TXCSetup_2.02Stable_Win32.exe
- TypeFaster-v0.4.2-install.exe
- TypingMaster10demo.exe
- UndeleteMyFilesSetup.exe
- UnitConverter.exe
- UnstopCpy_4_4_Win2K_UP.exe
- usb_network_gate.exe
- uTorrent.exe
- ViberSetup.exe
- VidCoder-5.21.exe

- vidshot-capturer.exe
- w32-473.exe
- WeatherEyeDownloader_4.0.exe
- weatherpulse-beta-2.10-build7-setup.exe
- WebFerretSetup.exe
- WeChatSetup.exe
- winguide-2.0-win32.exe
- WinMerge-2.16.12-Setup.exe
- WinTabber.exe
- WinToHDD_Free.exe
- xeoma_win64.exe
- xneat-windows-manager_setup.exe
- ZoomInstaller.exe

Appendix B

List of Windows Ransomware MD5 fingerprints

- 000b28e7a5905b5bf3b80c6c60a5f828
- 00a38bce6818521bed3c78b14ddef25f
- 00b26c8964bf6c20183d13867e6dbcb0
- 00b35ea8173b4f5116e599d9f80e3611
- 00ce1f799aa5dc2f43c366dc60dc6098
- 00ce45f73fa3bdbb862357983ce8989d
- 088208cefdeef7e6e2a1126e30228b85
- 0a06fbd7930744b8a6b24c92007913ce
- 0a07df0841c559bcb2fd3e5486130293
- 0a09a21c2f28c8c5b5d168feb668a80b
- 0a09c719f867439465c45bcb853a7be1
- 0a13a956e4b398aac91c736715e24030
- 0a14bbb02a94ebb47d1068ff2e71b282
- 0a33af27fc0239b5205067cf5a146352
- 0a33ca032358f29faff821136ed8fec0
- 0a34ca9df5dd56f8fed7610e74cf1c31
- 0a36ff9756610800eb13e0920b310e0a
- 0a6c2525634713bbc92d2fc61c43c882
- 0a7aca58836451c4f7843c43f4981d7f
- 0a7aeb15c02416a22d855a1a525f2596
- 0a7dc02826bf3c091a1a65efe5e5e7f2
- 0a7dda5e0482655a76fedb57622f7b81
- 0a8bde4a4ddddaaeaa06ce2db064feb2
- 0a9ce179ac1b183e01897938e83d3949
- 0b10e38a251c45f54ddeb750f5efcac9
- 0b13d9844d6b76adf1b528c73b56754f
- 0b16d47bf27428fdd9f7181306337487
- 0b21d2e272ffbbe95b747aa7c0c96659
- 0b22c54538c81b70de6fec49c2a0898a
- 0b22ed62c0b8e0d34e4e21006c662a76
- 0b28ceee1e738c5ac645053e1bbdd4a7
- 0b30eab2b989c9c0de4ce4bab6352f92
- 0b32c3e3bf18cbc418ec8f43dcea75b0
- 0b34f528cbbb780192d043b1127756ae
- 0b34ff052eac077c6464eb112e4ad0ab
- 0b35e10917f4ba647661f7f27b24ebb0
- 0b40e34dea8c34fbb75896ac4c1c42e4
- 0b6a0fe4aef0d2905b091102807a42ca
- 0b6a667eeb755ae61e3b01d6dc9cd812

- 0b6dfe55374313977ac33876011506e0
- 0b6fa82d31b88c7f6e24a32e431ff89e
- 0b6ff0b81d7d9576a4b1876d6854cad5
- 0b7a628509df4b6253cce75860f211f8
- 0b7ddf9bcb87bd9925dace367b85ab50
- 0b7ee60630737c58f5434a1b17c23469
- 0cd10be380dfe5b9adf88a2691d49d3e
- 0cd7aa9c367a8f0f65ba790f8e0a2223
- 0cd8c45d4da143a129fc75e7ccb24103
- 0cdc7f2318174f819872f169a6989d43
- 0cdd06bb2d12bcaadd16d126fdf19ae5
- 0cde55d6d3510f66c5180774c802d2a0
- 0ce4d65a9c7639303daf2e50b3883580
- 0ce52d5069ca6276a9c84fb4b22052cc
- 0ce63bfbfe3f60b7284d528ec1486981
- 0ce6b27487a28187dc49d6153d766fa0
- 0ce76aae5b1789baf99bfeb02b444e3b
- 0ceae1287931e467256574040157d378
- 0cec65a8a6e74497bf802125c96621e0
- 0cecfe58f0a6eb2060b764de7b5d4d25
- 0ceed71c5cfa44b988d70ddfd0d81dc4
- 0cef80bd43ef1860e33ce05e903d51d6
- 0cefbf1386781ea858485b4ccc390920
- 0cefce0dbbbedc5eb1febe4d85b23c71
- 0cf053cfd40eeae31ec034d793ec5f90
- 0cf08b5be2b33bb4a19c89deea07a724
- 0cf2e3c0cae4792231f345fc79c19e50
- 0cf35dbe0768d375031678ee14c9b7f0
- 0cf38aab9a8ea0b1eba32fab91ec645e
- 0cf3e91199fbef9cbbe3001d39c620d8
- 0cf46f706503926a159d4599f4370b90
- 0cf9d253ca7243737c3507f876f446b2
- 0cfa477d69952a90a512490f5b79274f
- 0cfab3f8ea799e75d1817bf171f306ae
- 3c0948c9f4ccc0346d59b39dfedab2d2
- 3c1849e69e4227a9fd4a4c6a1ff55a68
- 3c2215d04660208e39536a6d1ea7f680
- 3c650d803307de7902e62da1c02f3ca9
- 3c658c1945c067468d8f79db07643e90
- 3c663fad82be493ec0b0ecb044a57600
- 3c704a12c280b456e3359af78e33e206
- 3c786dd75daa0ddcd5d36607d6eeba4b
- 3c875d58d7b81642e0d7be512766c0d0
- 87644bd77a74a82d8ed2d01d08846c04
- 87743e5b89a8719a809774e2f2446380
- 87887dc9c1d09771a7fa05cee0827740
- 87887e5d5a288fe1bcae74e47df5141c
- 87957f64f0580b8192566172a550e643
- 87976dea8e494333bed3db1a0e950ac1
- 88086b6e605af87d5c8bca9913985fc6
- 88162c8d9a44917296423f48df81996f
- 88216f785a604ceae1ad4f4ba94652b3
- 88219d4f4fd890c06caa061eeb26dbf8
- 88329fd1cb04e1cf88296e8452db3940
- 88388acc273378b8bac2400d32781bea
- 88392d9fee377c417c8afadb75f5fb78

- 88452f1fcecd99e94a5e5fc6421d1149
- 88500c46b87a254c0b767103a2c6aed0
- 88505c57d47b5a74b5613585ec80bca9
- 88650d1b826f84a08bbfa1db34455caa
- 88660b93c0558738766d1a5c60773eb4
- 88722f4ca0a8be5da2fe25b4231f82f0
- 88734be7b84310a02850469ba0511205
- 88781a94b87e233e285c8bc597dfe971
- 88898c2cbaa83165b7033792b2f30cd2
- f08aaa9da49ba67afe8e891d365e41a9
- f7f69d63af1986fdda7096e693ec4190
- f7faa1ef5603afe7d461725613a951a7
- f7fac8f1d3b99b5f11f3872824690321
- f7fb92a71b96fc9f62d73bee797e8b13
- f7fd3e47bdc115db6183c090d85bad89
- f8a12e3606aec7788232569fcaa3e8f0
- f8a14950826702bfe6e7eb52eb20c2a6
- f8a150b0f86b349f7be0789c38bccc78
- f8a1bf1064eaba4a87a8a9535c4ddd80
- f8a3baf290cb52f88cfd44d6a5586326
- f8a3dfcd0a53e75643aef41e83b9a2af
- f8a4b7ab70cc847aedd6075a6a01589f
- f8a543c021d19c2233ea3b3dc20e1d47
- f8a78136c3ce5105ccca213190d68f90
- f8aee730635bc08336e4ac08c5a9163a
- f8b26df426b46725c7eac85e9af2d244
- f8b7b3a4dd59c2633e3b835a88da1d8e
- f8b826a37ea098f8541445e70825a147
- f8b8b25c5cff0d227d6a90fa23862ea6
- f8b93b4f00a5d95f9f4c102e419f5ee6
- f8b9590bcca5093d9a0830ed57f0e756
- f8ba55dc668e5a1a14e207c3db53c218

Appendix C

List of Linux Applications

- cheese
- eog
- evince
- firefox
- nautilus
- rhythmbox
- software-update-gtk
- thunderbird
- transmission-gtk
- ubuntu-software

Appendix D

List of System Calls

- _llseek
- _newselect
- accept
- access
- arch_prctl
- bind
- brk
- capset
- chdir
- chmod
- chroot
- clock_getres
- clone
- close
- connect
- dup
- dup2
- epoll_create
- epoll_create1
- epoll_ctl
- epoll_wait
- eventfd2
- execve
- exit
- exit_group
- faccessat
- fadvise64
- fadvise64_64
- fallocate
- fchdir
- fchmod
- fchmodat
- fcntl
- fcntl64
- fdatasync
- fgetxattr
- flistxattr
- flock
- fsetxattr
- fstat
- fstat64

- fstatfs
- fstatfs64
- fsync
- ftruncate
- ftruncate64
- futex
- getcwd
- getdents
- getdents64
- getegid
- getegid32
- geteuid
- geteuid32
- getgid
- getgid32
- getgroups
- getpeername
- getpgrp
- getpid
- getppid
- getpriority
- getrandom
- getresgid
- getresgid32
- getresuid
- getresuid32
- getrusage
- getsockname
- getsockopt
- gettid
- getuid
- getuid32
- inotify_add_watch
- inotify_init
- inotify_init1
- inotify_rm_watch
- ioctl
- kill
- link
- listen
- lseek
- lstat
- lstat64
- madvise
- memfd_create
- mincore
- mkdir
- mlock
- mmap
- mmap2
- mprotect
- mremap
- munlock
- munmap
- nanosleep
- newfstatat

- openat
- pipe
- pipe2
- poll
- prctl
- pread64
- prlimit64
- ptrace
- pwrite64
- quotactl
- read
- readahead
- readlink
- readlinkat
- recv
- recvfrom
- recvmsg
- rename
- restart_syscall
- rmdir
- rt_sigaction
- rt_sigprocmask
- rt_sigreturn
- sched_getaffinity
- sched_getparam
- sched_getscheduler
- sched_setaffinity
- sched_yield
- seccomp
- select
- send
- sendmmsg
- sendmsg
- sendto
- set_robust_list
- set_thread_area
- set_tid_address
- setitimer
- setpriority
- setsid
- setsockopt
- shmat
- shmctl
- shmdt
- shmget
- shutdown
- sigaltstack
- socket
- socketpair
- stat
- stat64
- statfs
- statfs64
- statx
- symlink
- sysinfo

- tgkill
- time
- times
- ugetrlimit
- umask
- uname
- unlink
- unlinkat
- utimensat
- wait4
- waitpid
- write
- writev

Bibliography

[1] PwC Team. 23rd Annual Global CEO Survey: Navigating the rising tide of uncertainty; 2020. Available at https://www.pwc.com/gx/en/ceo-survey/2020/reports/pwc-23rd-global-ceo-survey.pdf. pages 1

[2] Verizon Team. Analysing the COVID-19 data breach landscape; 2021. Available at https://enterprise.verizon.com/en-gb/resources/articles/analyzing-covid-19-data-breach-landscape/. pages 1

[3] Scott Ikeda. Clop Ransomware Attack Hits German Software Giant Software AG; Confidential Documents Stolen, $23 Million Ransom Demanded; 2020. Available at https://www.cpomagazine.com/cyber-security/clop-ransomware-attack-hits-german-software-giant-software-ag-confidential-documents-stolen-23-million-ransom-demanded/. pages 1

[4] Sopra Steria Team. Cyberattack: updated information; 2020. Available at https://www.soprasteria.com/newsroom/press-releases/details/cyberattack-updated-information. pages 1

[5] Alina Bizga. Law Firm Seyfarth Shaw Hit by Apparent Ransomware Attack; 2020. Available at https://securityboulevard.com/2020/10/law-firm-seyfarth-shaw-hit-by-apparent-ransomware-attack/. pages 1

[6] Check Point Team. 2021 Mid Year Report; 2021. Available at https://pages.checkpoint.com/cyber-attack-2021-trends.html. pages 1

[7] Verizon Team. 2021 Data Breach Investigations Report; 2021. Available at https://enterprise.verizon.com/en-gb/resources/reports/dbir/2020/summary-of-findings/. pages 1

[8] Chris Greco. Mobile Users Prove More Susceptible to Phishing Attacks; 2020. Available at https://blogs.blackberry.com/en/2020/02/mobile-users-prove-more-susceptible-to-phishing-attacks. pages 2

[9] Team C. Phishing attacks are ramping up on smartphones; 2020. Available at https://corrata.com/phishing-attacks-ramping-up-on-smartphones/. pages 2

[10] Mandiant Team. APT1: Exposing One of China's Cyber Espionage Units; 2020. Available at https://www.fireeye.com/content/dam/fireeye-www/services/pdfs/mandiant-apt1-report.pdf. pages 2

[11] CISA F. Joint Cybersecurity Advisory: DarkSide Ransomware: Best Practices for Preventing Business Disruption from Ransomware Attacks; 2020. pages 3

[12] Chad Anderson. The Most Prolific Ransomware Families: A Defenders Guide; 2021. Available at https://www.domaintools.com/resources/blog/the-most-prolific-ransomware-families-a-defenders-guide. pages 3, 4

[13] Richard Hickman. Conti Ransomware Gang: An Overview; 2021. Available at https://unit42.paloaltonetworks.com/conti-ransomware-gang/. pages 3

[14] Charlie Osborne . FBI identifies 16 Conti ransomware attacks striking US healthcare; 2021. Available at https://www.zdnet.com/article/fbi-identifies-16-conti-ransomware-attacks-striking-us-healthcare-first-responders/. pages 4

[15] Yaron Zinar. Maze Ransomware Analysis and Protection; 2021. Available at https://www.crowdstrike.com/blog/maze-ransomware-analysis-and-protection/. pages 4

[16] Dvir Sason. REvil Ransomware Attack on Kaseya VSA: What You Need to Know; 2021. Available at https://www.varonis.com/blog/revil-msp-supply-chain-attack/. pages 4

[17] Zack Whittaker. Maze, a notorious ransomware group, says it's shutting down; 2020. Available at https://techcrunch.com/2020/11/02/maze-ransomware-group-shutting-down/. pages 5

[18] Kaspersky Team . Ransomware Attacks and Types – How Encryption Trojans Differ; 2021. Available at https://www.kaspersky.com/resource-center/threats/ransomware-attacks-and-types. pages 5

[19] Justin Fier. Crypto-mining malware: Uncovering a cryptocurrency farm in a warehouse; 2021. Available at https://www.darktrace.com/en/blog/crypto-mining-malware-uncovering-a-cryptocurrency-farm-in-a-warehouse/. pages 5

[20] Lawrence Abrams. Computer hardware giant GIGABYTE hit by RansomEXX ransomware; 2021. Available at https://www.bleepingcomputer.com/news/security/computer-hardware-giant-gigabyte-hit-by-ransomexx-ransomware/. pages 5

[21] Brewer R. Ransomware attacks: detection, prevention and cure. Network Security. 2016;2016(9):5–9. Available from: https://www.sciencedirect.com/science/article/pii/S1353485816300861. pages 5, 15

[22] Olaimat MN, Aizaini Maarof M, Al-rimy BAS. Ransomware Anti-Analysis and Evasion Techniques: A Survey and Research Directions. In: 2021 3rd International Cyber Resilience Conference (CRC); 2021. p. 1–6. pages 9

[23] Edward Kost. What is Ransomware as a Service (RaaS)? The Dangerous Threat to World Security; 2021. Available at https://www.upguard.com/blog/what-is-ransomware-as-a-service. pages 9

[24] Crowdstrike Team. Ransomware-as-a-Service (RaaS) Explained; 2021. Available at https://www.crowdstrike.com/cybersecurity-101/ransomware/ransomware-as-a-service-raas/. pages 9

[25] Sam Ingalls. Ransomware Protection in 2021; 2021. Available at https://www.esecurityplanet.com/threats/ransomware-protection/. pages 10

[26] Snir Ben Shimol. Return of the Darkside: Analysis of a Large-Scale Data Theft Campaign; 2021. Available at https://www.varonis.com/blog/darkside-ransomware/. pages 10

[27] Ben Lutkevich. What is an Intrusion Detection System?; 2020. Available at https://searchsecurity.techtarget.com/definition/intrusion-detection-system. pages 10

[28] Ma S, Zhang X, Xu D. ProTracer: Towards Practical Provenance Tracing by Alternating Between Logging and Tainting. In: NDSS; 2016. . pages 10

[29] Lee KH, Zhang X, Xu D. High Accuracy Attack Provenance via Binary-based Execution Partition. In: 20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013. The Internet Society; 2013. Available from: https://www.ndss-symposium.org/ndss2013/high-accuracy-attack-provenance-binary-based-execution-partition. pages 10

[30] Lee KH, Zhang X, Xu D. LogGC: Garbage Collecting Audit Log. CCS '13. New York, NY, USA: Association for Computing Machinery; 2013. Available from: https://doi.org/10.1145/2508859.2516731. pages 10

[31] Linda Rosencrance. What is a Security Information and Event Management System?; 2020. Available at https://searchsecurity.techtarget.com/definition/security-information-and-event-management-SIEM. pages 11

[32] Daniel Nieuwenhuizen. A behavioural-based approach to ransomware detection; 2017. Available at https://labs.f-secure.com/assets/resourceFiles/mwri-behavioural-ransomware-detection-2017-04-5.pdf. pages 11, 12

[33] Brian Turner, Mike Williams . Best ransomware protection of 2021: free and paid decryption tools; 2021. Available at https://www.techradar.com/best/best-ransomware-protection. pages 11

[34] Verma M, Kumarguru D, Deb S, Gupta A. Analysing Indicator of Compromises for Ransomware: Leveraging IOCs with Machine Learning Techniques; 2018. p. 154–159. pages 12

[35] Asmitha KA, Vinod P. A machine learning approach for linux malware detection. In: 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT); 2014. p. 825–830. pages 12

[36] Hou S, Saas A, Chen L, Ye Y. Deep4MalDroid: A Deep Learning Framework for Android Malware Detection Based on Linux Kernel System Call Graphs. In: 2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW); 2016. p. 104–111. pages 12

[37] Chen YC, Li YJ, Tseng A, Lin T. Deep Learning for Malicious Flow Detection; 2018. pages 13

[38] Vinayakumar R, Soman KP, Senthil Velan KK, Ganorkar S. Evaluating shallow and deep networks for ransomware detection and classification. In: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI); 2017. p. 259–265. pages 13

[39] Sgandurra D, Muñoz-González L, Mohsen R, Lupu EC. Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection; 2016. pages 13

[40] Gomez-Hernandez, Jose and Álvarez-González, L and García-Teodoro, Pedro. R-Locker: Thwarting ransomware action through a honeyfile-based approach. Computers & Security. 2017 12;73. pages 13

[41] MITRE Organisation. Common Attack Pattern Enumeration and Classification; 2007. Available at `https://capec.mitre.org/`. pages 14

[42] VirusShare Team. VirusShare; 2021. Available at `https://virusshare.com/`. pages 14, 19

[43] Malshare Team. Malshare; 2021. Available at `https://malshare.com/`. pages 14

[44] The Zoo Team. The Zoo; 2021. Available at `https://github.com/ytisf/theZoo/tree/master/malware/Binaries`. pages 14

[45] VirusBay Team. VirusBay; 2021. Available at `https://beta.virusbay.io/sample/browse`. pages 14

[46] VirusSign Team. VirusSign; 2021. Available at `https://virussign.com/downloads.html`. pages 14

[47] VirusSamples Team. VirusSamples; 2021. Available at `https://www.virussamples.com/`. pages 14

[48] VirusTotal Team. VirusTotal; 2021. Available at `https://www.virustotal.com/gui/home/upload`. pages 14

[49] Google Cloud Team. CISO's Guide to Cloud Security Transformation; 2021. Available at https://services.google.com/fh/files/misc/ciso-guide-to-security-transformation.pdf. pages 15

[50] Software Informer Team. Software Informer; 2021. Available at https://software.informer.com/software/. pages 19

[51] Open source contributors. Bash Ransomware; 2021. Available at https://github.com/SubtleScope/bash-ransomware. pages 19

[52] Marco Santos. Precision or Recall: Which Should You Use?; 2020. Available at https://towardsdatascience.com/explaining-precision-vs-recall-to-everyone-295d4848edaf. pages 33

[53] David Chismon. HUNTING WITH HONEYPOTS; 2016. Available at https://www.f-secure.com/en/consulting/our-thinking/hunting-with-honeypots. pages 36

[54] Brendan Gregg. Choosing a Linux Tracer; 2015. Available at https://www.brendangregg.com/blog/2015-07-08/choosing-a-linux-tracer.html. pages 37

[55] Linux Ransomware. Training and Testing Dataset for Linux Ransomware Detection; 2021. Available at https://drive.google.com/file/d/184xIktvwkpDE4lrbzuvyxLrvtMY4dsjN/view?usp=sharing. pages 39