Final Year Interim Project Report

**Full Unit – Interim Report**

DEVELOP A SECURITY SUITE FOR ANDROID-BASED SMARTPHONES.

Abd ElRahman Mohamed Hassan Abdou M Soliman

**Supervisor:** Christian Weinert



Department of Computer Science

Royal Holloway, University of London

November 22, 2022

Table of Contents

[Table of Contents 1](#_Toc119878966)

[Project Specification 2](#_Toc119878967)

[Chapter 1: Introduction 3](#_Toc119878968)

[1.1 Abstract 3](#_Toc119878969)

[1.2 Aims & Objectives 3](#_Toc119878970)

[1.3 Motivations 4](#_Toc119878971)

[1.4 Background Theory 5](#_Toc119878972)

[1.5 Literature Review 8](#_Toc119878973)

[1.6 Summary of Completed Work 8](#_Toc119878974)

[1.7 Plan for Term 2 8](#_Toc119878975)

[Chapter 2: Technical Achievements 9](#_Toc119878976)

[Chapter 3: 10](#_Toc119878977)

[Chapter 4: Software Engineering 11](#_Toc119878978)

[4.1 Code Breakdown 11](#_Toc119878979)

[4.2 Methodology 11](#_Toc119878980)

[4.3 Testing 11](#_Toc119878981)

[4.4 Figures 11](#_Toc119878982)

[Bibliography 12](#_Toc119878983)

Project Specification

As taken from InfoSec Final Year Project List:

**Aims:**Develop a security suite for Android-based smartphones.

**Background:**This project will develop a security suite for Android smartphones. There are several security apps on the Google Play Store which take care of certain security aspects but there isn’t any all-in-one solution which covers the security as a whole. This project will analyse the possibility of having one such security package by using secure sensor management and enforcing authentication and access control policies without modifying the ROM or Android OS.

**Prerequisites**

* Good programming skills, particularly Android app development, and knowledge of IDEs.
* Being able to autonomously download/compile/install tools from various repositories (e.g., GitHub).
* **Prerequisites**

Early Deliverables

1. A report describing different security apps for integrating into the security suite.
2. A proof-of-concept implementation of the security suite
3. Testing the system

#### Final Deliverables

1. Design and develop the final security suite
2. Complete report which must include a User Manual

# Introduction

## Abstract

## Aims & Objectives

With this project, my goal is to unite and package multiple existing android security modules into a do it all mobile security centre that lives on the user’s android device of choice and conveniently shows the user an overview of the level of security that their device currently possesses in a user-friendly understandable way. I aim to achieve this by using open-source implementations of various security features as the foundation that will allow me to develop my fit-for-purpose implementation that can be incorporated into my security suite without requiring a standalone app for each module or modifications to the OS. I aim to present the data being shown by my security suite in an understandable way that doesn’t panic the user if not necessary yet will also allow the user to use multiple of the modules listed below with the press of a button to keep their security up to standards. A secondary goal is to have this security suite run automatically in the background, ensuring that all security definitions are up to date for the malware detection algorithm and provide day-to-day security without user input.

This project will analyse the possibility of having an all-in-one security suite on android that:

* Requires no prior modification to the OS
* User-Friendly
* Up to date using the latest security definitions
* Is Compatible with older versions of Android
* Handles multiple modules of Security on an Android Device from this list:
  + **Anti-Virus/Malware Detection**
  + Firewall
  + **Overview/Management of app permissions**
  + **Overview/Management of active sensor usage (including camera, microphone, etc.)**
  + **File Encryption (.zip, .rar encryption?)**
  + Password Manager
  + VPN/ Tor
  + Secure Messaging
  + **App Access Control**

## Motivations

Since the first release of the Android mobile Operating System in 2008, Android Smartphones have developed to become an integral part of day-to-day life and as smartphones have improved and added features, the amount of personal data they managed also increased. Android Smartphones now can function as anything from a way to digitally sign contracts to complete bank branches in our pocket. They can replace anything from our Debit Cards to our laptops we become increasingly reliant on the palm-sized slabs of glass in our pockets, and the need for security becomes more and more crucial. Especially now more than ever as new malware is being developed and exploited quicker than exploits can be patched and as our devices become more and more connected to the outside world an up-to-date fully featured android security suite will help keep our phones secure.

To combat this, various Android smartphone manufacturers have been guaranteeing Android Security Updates on a monthly, bi-monthly, or quarterly basis for a limited number of years depending on the age of the device. This is beneficial in practice until the variation in the number of years being supported from manufacturer to manufacturer is realised. For example, Google, which has a 0.5% market share (appbrain.com, 2022), can promise at least 3 years of Security updates across all their devices with their latest smartphones offering at least 5 years (Google Support, 2022) whereas Oppo, with 10.1% market share (appbrain.com, 2022), uses a “the more you spend, the more you get policy”(Android Authority, 2022) with mid-range and budget models receiving less and fewer Security updates as you go down in price with their ultra-budget phones seeing no updates. Maintaining the security and increasing the longevity of these phones is one of my motivations for this project.

For most people, Android already comes with all the essential safety and security features such as Encryption and App Security and basic malware and anti-virus protection. However, the security level depends on whether the device is running the latest version of the OS. Unfortunately, the Android Version Market Share is very fragmented with only 23.5% of devices running the latest Android 12.0 with more than 27% using a version of Android that is over 3 years (Android Version Market Share Worldwide | Stat counter Global Stats, 2022). Naturally, owners of older android devices look to the Google Play store for a 3rd party Security solution such as those provided by McAfee or Malwarebytes and then soon realise that to cover every aspect of security on their device, they would need to download multiple apps because there is no full-featured Security suite that can act as an all-in-one solution for your security needs on your android device. This inconvenience may deter less tech-savvy users from maintaining the security of their older devices. Having an all-in-one Security Suite on Android will benefit and allow those users to protect their data more conveniently.

I have taken an interest in this project as I am hoping to pursue a career in the Cybersecurity Industry with a particular interest in Mobile Security. I am an avid Android User who has resisted the rise of IOS for many years and have taken an interest in Secure Messaging and Encryption having programmed a secure messaging app (reminiscent of something like WhatsApp or Messenger) using Java and Google Firebase. Although I’m new to on-device android security, it has always been a subject that I have wanted to pursue in the future as I do believe that the Play Store’s App Requirements are a bit relaxed in the security department and this is a major issue as Android apps keep popping up on the news because of malware detection.

## Background Theory

### What security apps can be implemented into the Security Suite?

The need for Mobile Security is becoming increasingly apparent as the number of mobile devices that are being used increases year after year. Over 70% of mobile devices in the world are powered by the Android Operating System (StatCounter Global Stats, 2022) with over 2.8 billion active users potentially being at risk of a security vulnerability. There has never been a larger number of potential vulnerabilities in Mobile Security than today with how connected personal devices are with the rest of the world. Mobile devices can be targeted on an OS, Application or Network Level. This report will identify and explain potential security apps that could be implemented in a security suite to mitigate the risk of a security breach.

#### Malware Detection

A traditional computer virus works by infecting local files on a particular device and then using that device’s resources to spread. However, Malware on an android device is not likely to work in this way and is more likely to come in the form of a malicious app with the most common attack vector being repackaging popular benign apps with malicious payloads that the user will install unknowingly. It is still possible for malware on android to work in the traditional sense through infected files that pose as benign such as images. This leads to there being 2 different types of Malware Detection that can be run on android:

* App-Based Malware Detection
* File-Based Malware Detection

App-Based Malware Detection

Android is a privilege-separated operating system which fundamentally means that user-installed android applications are not granted most permissions by default and must obtain more sensitive permissions to interact with OS services, hardware, and even other applications. An android app must define its requested permissions in its Androidmanifest.xml file. Every android app has this file which describes any essential information about the application such as:

* “The components of the app, which include all activities, services, broadcast receivers, and content providers. Each component must define basic properties such as the name of its Kotlin or Java class. It can also declare capabilities such as which device configurations it can handle, and intent filters that describe how the component can be started” (Android Developers Guide, 2022)
* “The permissions that the app needs to access protected parts of the system or other apps. It also declares any permissions that other apps must have if they want to access content from this app.” (Android Developers Guide, 2022)
* “The hardware and software feature the app requires, which affects which devices can install the app from Google Play.” (Android Developers Guide, 2022)

Most Android App-Based Malware Detectors focus on the permissions aspect of the Android Manifest file as each permission may equate to sensitive actions such as the sending of an SMS Message or access to the device’s contacts and files. Due to these permissions being so easily declared, they can be very helpful in recognising the true intentions of any given app. Permissions are known to be the “best single predictor of the app’s malignity reaching the accuracy of about 96%” (Sebastian Hahn et al., 2016) with its accuracy increasing as we add more metadata into the analysis. The main advantage of manifest analysis is that a practical implementation can be done on a device with automatic detection as apps are installed. Another advantage is that it can also scan APKs before the user installs the application. However, static analysis of apps can also lead to many false positives and even malicious apps being missed entirely as although the manifest file declares exactly all the sensitive actions the app is going to perform, the actual behaviour of the app/ what it does with those permissions can only be derived from its actual code.

For my security suite, I have gone for an implementation that will use the permissions and intent filters from a given app’s Android Manifest file to attempt to detect malware. More specifically, I have chosen to implement LibreAV which is an Open-Source Anti-Malware application for android that utilizes machine learning by retrieving datasets of malicious apps and benign apps along with the chosen app’s permissions and intent filters and running them through a Tensor Flow lite algorithm that will return a number from 0-1 with anything under 0.5 being safe and anything over 0.5 being classed as either risky or as malware. I am hoping that this prediction system along with showing the user the list of permissions an app is using will be enough to allow the user to make an informed decision as to whether they should keep or delete an app from their devices.

File-Based Malware Detection

Unfortunately, the Android platform is also vulnerable to more traditional malware from infected files such as documents or images. Android devices can get compromised just by downloading infected/malicious files such as PDFs. In this instance, it can be harder for an average user to be able to tell whether a document they downloaded is malware or not, especially due to Android’s default file manager not being as thorough or detailed compared to Windows or Linux. Thankfully, detecting File-Based Malware is a lot more straightforward than detecting malicious apps.

A File-Based malware detector works by keeping a database of virus definitions that is updated frequently as increased viruses get profiles. A virus definition is a binary pattern that identifies a specific virus (TechTerms.com, 2022). By checking files and programs against a database of virus definitions, the malware detector can determine whether the file/program is malicious or not. This database of virus definitions needs to be updated regularly due to the relentless pace that malware is being developed with at least once a week being the recommended update frequency of the database. Further development of the Malware Detector can lead to the ability to generate heuristics allowing the detector to detect unknown viruses just based on their similarity to an existing virus definition.

For this security app, I am using Hypatia which is an Open-Source implementation of a File-Based Malware Detector that works by hashing files on the device and comparing the hashes with a local virus definitions database that is being updated regularly. Hypatia is built on ClamAV databases and ESET databases and allows the user to choose which database to install as well as how detailed they would like the database to be. This provides the user with the choice of balancing the size of the database with their phone storage. Hypatia also allows for Realtime scanning meaning if a new file was to be transferred or downloaded on the device, it will be scanned instantly and alert the user of its status.

#### Permission Manager

It is common knowledge that when a user wants to modify permissions settings for any app on their device’s settings, they are unable to change all permissions and the settings app does not even show the user a comprehensive list of all the permission the app can use. It only shows basic permissions that the user can change such as files and contacts’ access. This is where a Permission Manager can be handy. Unfortunately, unless the device is rooted, the only service that can change permissions for apps is the OS and by extension the settings itself. This means that for the user to change all permissions through an external app, they would need to modify the OS of their device. This goes against one of my Aims and Objectives for this Security Suite.

Because of this, I have settled on an app that can show the user all the permissions a particular app is using by grabbing all the requested permissions from the AndroidManifest.xml file and viewing them as a list to the user. The user can also click on a permission string from the list to find out more information about those permissions. If the user wants to change any permissions, they can press the open in settings app which will directly open the settings page for that app and then the user can change the permissions android allows them to change.

Although not ideal, this watered-down version of a true permission manager still provides value to the user as it ensures that the user knows exactly how and what permissions each app is using. This could also help the user identify malicious apps on their own. For example, if there was to be a news app installed but it requests the ability to send SMS messages. This can be logically seen as malicious as one can assume that a news app does need to be able to send messages to function.

#### App Locker

All Android devices come with built-in locking mechanisms that allow the user to prevent malicious parties from easily gaining access to their device when said parties have physical access to the device. This always comes in the form of a lock screen. A lock screen is a view that appears upon start-up and every time the screen is woken up that prevents the user from accessing the rest of the device until the user enters their pin/password/pattern/biometrics. Once the user unlocks their device, they are greeted with their lock screen and apps and the rest of the device.

However, due to how often a user would unlock their phone each day, this is vulnerable to shoulder surfing and smudge attacks where a malicious party would look out for smudges on the screen to reverse engineer the password. This is also vulnerable to user error in the case where the user leaves their phone unattended but unlocked. App locker is the obvious solution to this problem where each app has its own lock screen that the user can set to either require a different or the same password as the main lock screen. This will allow individual apps (chosen by the user) to have an extra layer of security on top of the already built-in lock screen which will ensure that the user’s personal data is just that bit more secure.

#### File Encryption

File encryption is a way of encoding data to prevent tampering or unauthorized access. In simple terms, the device uses a complex algorithm to change each individual bit in a file with the intention of making it impossible to work out the original contents of the file without using a decryption key. This ensures that any personal data that was stored on the chosen file is kept secure if a malicious party were to snoop in your device files.

Thankfully, since the release of Android 5.0, the developers of android began taking encryption a lot more seriously and went as far as to implement support for full-disk encryption until Android 9.0 before switching to a more modern implementation through File-based encryption. Full-disk encryption on android requires the user to enable the feature in settings and once encrypted, the user would need to input a pin/password/pattern before the device even booted into windows. Once the user is authenticated, the device boots on to Android. From then on, any user-created files are automatically encrypted before being saved and any encrypted data is automatically decrypted when requested.(Android Source, *Full-Disk Encryption | Android Source*, 2022) File-based encryption works in the same way but allows different files to be encrypted with different keys that can be unlocked independently which eliminates the need to authenticate the user before boot. (Android Source, *File-Based Encryption | Android Source*, 2022).

Since File Encryption has existed in Android for many years now, I have chosen not to pursue implemented file encryption into the security suite.

#### Active Sensor Usage Detection

#TODO

## Literature Review

#TODO

## Summary of Completed Work

#TODO

## Plan for Term 2

#TODO

# Technical Achievements

## Technical Application Achievements

### Google Material Design 3

Google Material Design 3 is the latest version of Google’s open-source design system that gives developers an in-depth UX Guidance and UI Component implementations for Android with the intention of providing a consistent “personal, adaptive, and expressive experience”(Material Design 3, 2022).

Material Design Is split into 3 main parts: **foundations, styles, and components**. Foundations are a set of standards that define what google considers to a great user interface including standards regarding Accessibility and even how the interface reacts to inputs. It also heavily pushes Adaptive Design which allows the interface to adapt to the device it is currently running on. Whether that be specific screen sizes or adapting to tablets and even desktops.

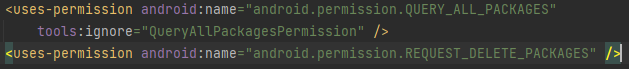
Styles dictates any visual aspects of a UI that allow the UI to have a distinct look and feel. This part of Material Design controls multiple attributes including:

* Colour – Mainly dynamic colour (keeping colours consistent with system colours)
* Elevation – Gives the UI a slight 3D looks like it’s popping out of the screen
* Icons – Standard buttons for actions such as Play or going back
* Motion
* Shape – Style of shapes such as roundedness in containers
* Typography – Make writing legible and appeasing to look that

Throughout my project, I am/will be looking to implement Material Design 3 into all aspects of my security suite to ensure that the User Interface is as consistent and friendly as android allows. In my experience, most anti-malware applications on android have non intuitive User-Interfaces which I found to be discouraging leading to me not opening them as much. I aim to eliminate this issue by simplifying the usage and look of the security suite using Google Material Design 3

## AndroidManifest.XML

Another Aspect of Android Development that I have been messing with and discovering new things about throughout my project is the AndroidManifest.XML. The AndroidManifest.xml is a manifest file in the root of the project that defines all the components, permissions, and intents that an android app may need. The aspect of this manifest file that I have been focusing on is the permissions. I’ve had to understand how to use more advanced permissions then what a conventional app may use. A couple of permissions that are commonly used across many of the apps in my security suite are:



The first permissions allow the app to query all the installed apps on the device and access a list of these apps whereas the 2nd allows the app to be able to delete selected apps at the discretion of the user without the need for the user to open the settings app and delete them manually.

I have also had to use different new components of the android manifest file such as the <service> component as well as the <receiver> component. On Android, a service is a like an activity component but lacks a UI that the user can interact with meaning services are usually background tasks or API’s that are used by other apps. A receiver component allows the app to receive intents that are broadcast by the OS. An Intent in its simplest form is a description of an operation that has been requested to be performed. I use a receiver in my Malware Detection app to check and thus scan when a new app has been installed.

Text

Description automatically generated

Text

Description automatically generated

Talk about a few More new technologies you have had to use

# User Interface Breakdown?

# Software Engineering

## Code Breakdown

## Methodology

## Testing

|  |  |  |  |
| --- | --- | --- | --- |
| Test Number | Expected Output | Actual Output | Screenshot/Figure |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Figures

Bibliography

3, MATERIAL DESIGN 2022. Get Started | Material Design 3.

GUIDE, ANDROID DEVELOPERS. 2022. *App Manifest Overview | Android Developers* [Online]. [Accessed 19/11/2022].

HAHN, SEBASTIAN, PROTSENKO, MYKOLA & MÜLLER, TILO. Comparative evaluation of machine learning-based malware detection on Android. Sicherheit, 2016.

SOURCE, ANDROID. 2022. *File-Based Encryption | Android Source* [Online]. Available: <https://source.android.com/docs/security/features/encryption/file-based> [Accessed 20/11/2022 2022].

SOURCE, ANDROID. 2022. *Full-Disk Encryption | Android Source* [Online]. Available: <https://source.android.com/docs/security/features/encryption/full-disk> [Accessed 20/11/2022 2022].

STATS, STATCOUNTER GLOBAL. 2022. *Mobile Operating System Market Share Worldwide | Statcounter Global Stats* [Online]. @statcountergs. Available: <https://gs.statcounter.com/os-market-share/mobile/worldwide> [Accessed 19/11/2022].

TECHTERMS.COM. 2022. *Virus Definition | TechTerms.com* [Online]. [Accessed 19/11/2022].

WU, DONG-JIE, MAO, CHING-HAO, WEI, TE-EN, LEE, HAHN-MING & WU, KUO-PING. DroidMat: Android Malware Detection through Manifest and API Calls Tracing. 2012. IEEE.