Preventing sensitive data being sent without encryption

Student: *Jake Williamson – C1433282*
Supervisor: *Eirini Anthi*
Moderator: *George Theodorakopoulos*
Module Code: *CM3203*
Module Title: *One Semester Individual Project*
Credits: *40*

Abstract

To make users more protective over their information, this project aims to make a proof of concept Android application that can inform users if their data is being sent unencrypted, in a way that other people can see. In addition to notifying the users, the project aims to give users control over what data their phone is sending. This application was developed in Android Studio using Java so that the application could follow guidelines set out by Android. The application works by detecting if information is being sent, and then presenting the user with the choice of allowing this information to be sent or preventing it; this is accomplished by implementing Android's VpnService, analysing each packet of network information that the phone sends, then allowing that packet to be sent based on the user's response.

The result of this project is an application that runs in the background and can detect if other applications are sending data without encrypting it first. By running this application, the user can see if their data is being shared, and who it is being shared with. This project also outlines future work that can be done to turn this proof of concept into a fully functional application.

Keywords: Android security, live network analysis, VpnService

Acknowledgements

I would firstly like to thank my supervisors Irene and Liam. Irene for giving me the opportunity to work on this project and giving me the guidance I needed to complete this work. Liam for sparing his time to help me with my, probably basic, development issues and pointing me in the right direction.

Secondly, I would like to thank my peers for making sure my application actually does what it is supposed to do and making sure that I am aware of any problems it had.

Finally, I would like to thank my parents for supporting me through this project.

Table of Contents

1. INTRODUCTION **1**

2. RELATED WORK **2**

3. DESIGN **4**

4. SPECIFICATION **6**

FUNCTIONAL REQUIREMENTS 6
NON-FUNCTIONAL REQUIREMENTS 8
RISK ANALYSIS 9
WIREFRAME DESIGNS 10

5. IMPLEMENTATION **21**

DEVELOPMENT ENVIRONMENTS 21
INTERESTING FEATURES 22
DIFFICULTIES FACED 24

6. TESTING **25**

FUNCTIONAL TESTING 25
NON-FUNCTIONAL TESTING 26

7. FUTURE WORK **29**

8. CONCLUSIONS **30**

9. REFLECTION ON LEARNING **31**

10. APPENDIX A – DATA CHECKER FLOWCHART **32**

11. APPENDIX B – AUTO FLOWCHART **33**

12. APPENDIX C – FUNCTIONAL TEST CASES **34**

13. APPENDIX D – ERROR TESTING **38**

14. REFERENCES **39**

Table of Figures

Figure 1 - Flow of Data in Application....4
Figure 2 - Application Activity Diagram....5

Wireframe 1 - Splash Screen....10
Wireframe 2 - Home Screen....10
Wireframe 3 - No Data Notification....11
Wireframe 4 - VPN Started....**Error! Bookmark not defined.**
Wireframe 5 - Edit Data Add Row....**Error! Bookmark not defined.**
Wireframe 6 - Edit Data Page....**Error! Bookmark not defined.**
Wireframe 7 - Data Saved....13
Wireframe 8 - Edit Data Delete Rows....13
Wireframe 9 - History Page....14
Wireframe 10 - Clear History Option....14
Wireframe 11 - Empty History Page....15

Table 1 - Risk Analysis Table....9
Table 2 - Application Features....21
Table 3 - Functional Requirements Implemented....25
Table 4 - Functional Test Results....25
Table 5 - Samsung Galaxy S8+ Test Times....27
Table 6 - Emulated Nexus 5 Test Times....27
Table 7 - Error Test Results....28

1. Introduction

Since 2009, the number of Android users has increased significantly [1], and the number of applications in the Google Play Store has increased by more than two orders of magnitude [2]. Most of these applications require that the user shares personal information such as address, age, Date of Birth (D.O.B), email, gender, name, phone number, and other information. However, the user has no guarantee that their information is being kept secure and that the application is trying to prevent unauthorised access to that information.

According to Kaspersky, a leading antivirus provider, data leakage is the number one threat to mobile devices, where applications send personal information to a remote server [3]. Gibler et al. [4] showed that 9.6% of Android apps, from a selection of 24,350 apps, forward sensitive data to third-party servers that can personally identify a user, such as the user’s phone information, Global Position System (GPS) location, Wi-Fi data, and audio recorded with the microphone. Even though each of these apps explicitly states what data is going to be collected, it is not clear to the user how the data will be used, or who it will be sent to. Due to this high percentage of Android apps sharing identifiable information, and with identity theft increasing in the UK [5], an application is in need that will inform users if their data is being shared without encryption and who it is being shared with.

Furthermore, Boyels et al. [6] showed that 54% of app users decided not to install an app when they discovered how much personal information they had to share to use the app, also, 30% of users uninstalled an app that was already on their phone when they learned it was collecting personal information that they did not want to share.

This paper details how a proof of concept application was created to inform users when their personal data is being shared; this information includes the user’s name, email address, D.O.B, phone number, and will also give them the choice of adding more fields if they desire. Additionally, this application will give the users the option to prevent this data from being shared and recording what application is sharing their data, giving them an informed decision on whether they want to keep the application or remove it. The paper is organised in the following way: Design covers a basic overview of how the application should work; Specification covers the application’s functional and non-functional requirements, and the wireframe designs of the application; Implementation covers the tools used by the developer and any interesting issues that arose; Testing describes the evaluation done on the application to ensure it meets the requirements; finally, the report shall end with the conclusion and future work.

2. Related Work

Previous attempts of trying to inform users which mobile applications are sending unencrypted data have required a lot of setting up, and usually require more than just the user's phone; whether this is additional hardware or using another website. While this approach works, it does require a lot more time than having an application installed on their phone.

The first method of determining if an application is failing to encrypt personal information is to set a computer or laptop up as a wireless hotspot, connecting a mobile phone to the hotspot, and running Wireshark [7] to capture the data sent by the phone. The user can then search for personal information using Wireshark's filters and determine which data is being shared and where it is going. The problems with this approach are that it requires some technical background, and it does not prevent the data from being shared. However, it is a trusted method if the user does have the technical knowledge as "Wireshark is the world's foremost network protocol analyser" [8].

The second method is a program called Datapp [9], an application created by University of New Haven's Cyber Forensics Research and Education Group (UNHcFREG). This application works by using its own version of Wireshark and automating the above process. The benefit of this is that it does not require any technical knowledge, as it can set everything up for the user and will show helpful messages if it requires any user interaction. However, this method also requires additional hardware and that the user's laptop is connected via a wired connection, meaning the user cannot monitor their data whilst they are on the move. Also, it does not stop the data being shared.

The third method was a service called Mobilescope [10] that could let a user examine all the data that their Android apps transferred. This service was a website where a user could see logs of the data transferred by the apps on their device. It worked by routing all the device's internet traffic through their servers so that it can analyse the data and notify a user if their data is being shared. The main issue with this method is that it sends the user's data to Mobilescope's servers, which means that they would have access to that data. Also, while it does alert the user if their data is being shared, it does not give them the option to block the data and, finally, their website has now been deemed "Not secure" by Google.

Another application that logs user's internet traffic is Packet Capture [11], an Android application that captures the phone's network traffic and logs the result. The issue with this application is that it does not alert the user if their data is being shared and the user interface is not very user-friendly, especially to those who do not have a technical background.

Finally, Hexene’s LocalVPN [12], an Android application that implements a basic packet interceptor built upon Android’s VpnService [13]. This application is what the project’s application will be built upon as it provides just a basic implementation, without adding unnecessary features.

3. Design

The proposed design for this project's application will work by using Android's VpnService [13] to capture the device's network data; this is being used so that the user's device does not require rooting [14] for the application to work; the application will be able to reach a much broader audience. To implement this, the VpnService will act as an intermediary between all applications on the phone, and the internet.

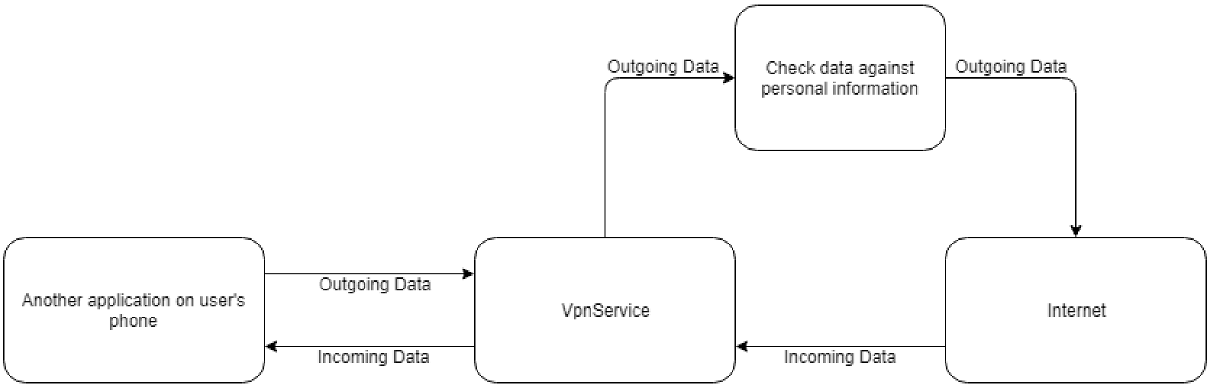


Figure 1 - Flow of Data in Application

Figure 1 shows how the user's data will flow between an application on the user's phone and the internet. Data coming from the internet will be sent to the VpnService and forwarded on to the application it was intended for. Data coming from an application and heading to the internet will be sent into the VpnService, which will then call a function that will determine if that data contains unencrypted personal information.

The activity diagram of an application is a flowchart that is used to represent the flow from one activity to another activity. The activity can be described as an operation of the system [15].

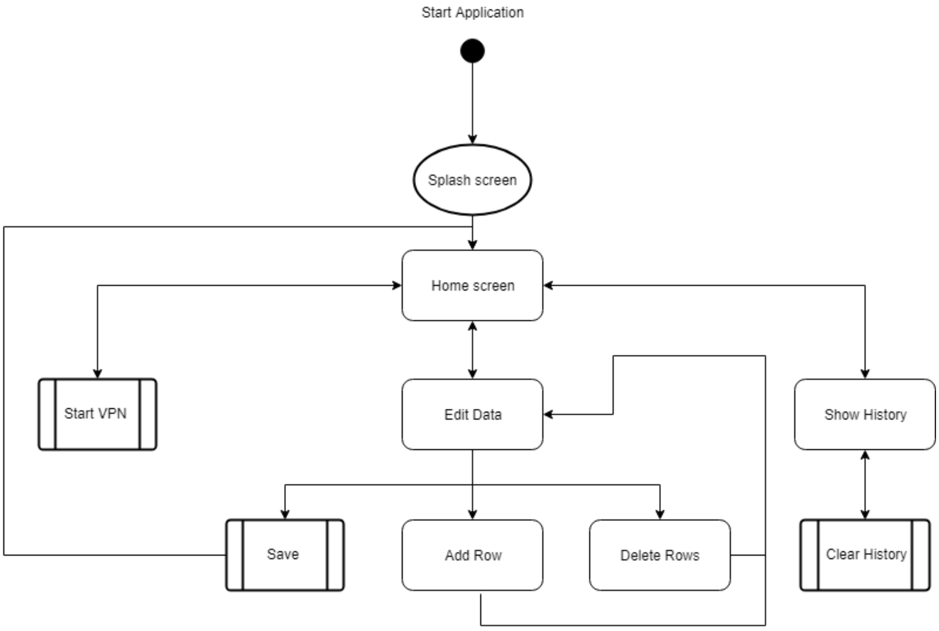


Figure 2 - Application Activity Diagram

Figure 2 shows the application’s proposed flow diagram. When the user starts the application the splash screen will be shown, which is collecting necessary data about the device, such as installed applications, Wi-Fi information, and network information. When the phone has the required information, it will load the application’s home page; from here, the user can start the VPN, edit their personal data stored in the application, or show a history of what the application has detected. From the “Edit Data” page, the user can choose to save any changes made, add a new row, or delete a row. From the “Show History” page, the user has the option to clear the history.

4. Specification

This section of the report will provide a detailed description of the functionalities of the Android application. This section will cover each of the application's functional and non-functional requirements.

Functional Requirements

Functional requirements are "the internal workings of the software: that is, the calculations, technical details, data manipulation and processing, and other specific functionality that shows how the use cases are to be satisfied" [16]

1. The application must use Android's VpnService to analyse the user's outgoing network traffic

Acceptance Criteria

- While enabled, the application will have access to each outgoing network packet [17] on the user's mobile phone.

Justification

- This requirement is necessary as the app will not be able to function if it cannot analyse the outgoing data.

2. The user should be able to enter what sensitive data they want the application to search for

Acceptance Criteria

- There should be a set of default fields that the user can fill in, such as name, email, Date of Birth (D.o.B), and phone number.
- The user should be able to add as many fields as they want.
- Other than the default fields, the user should be able to delete whichever field they want.
- The application should check for all data that the user has entered.

Justification

- This requirement is necessary as the app should be able to check for any personal data that the user requires.

3. If the application detects unencrypted data leaving the phone, the user will be notified and given the option to allow or block the data

Acceptance Criteria

- When the application detects unencrypted data, an urgent notification [18] will appear on the user's screen
- The notification will give the user the option to stop the phone sending the unencrypted data, or to allow it.
- Until the user has selected an option, the application will not send the data anywhere.
- If the user selects "allow", then the application will send the data to its intended recipient.
- If the user selects "block", then the application will remove the user's data and delete the packet.

Justification

- This requirement will allow the user to have full control over what happens to their data. Giving them the option to block the connection if they are not comfortable with it.

4. There should be a record showing the history of the user's actions

Acceptance Criteria

- Upon request, the application should present the user with a list of all data that the application has detected.
- The list should contain date and time of the action; the data that was allowed or blocked; the IP address of the server the data was going to; if it was blocked or allowed; and who blocked or allowed it.
- The user should have the ability to clear this list.

Justification

- This requirement means that the user will be able to keep a record of what unencrypted data the application has found, and when it found that data.

Non-Functional Requirements

Non-functional requirements “impose constraints on the design or implementation (such as performance requirements, quality standards, or design constraints).” [16].

1. Usability

Acceptance Criteria

- The application should be intuitive; a user should be able to learn how to use the application without any additional help
- The user interface should follow Google’s quality guidelines [19] and Nielson’s Heuristics [20] [21], as well as follow a consistent theme throughout the application.

Justification

- The application’s interface must be easy to use for any user to improve user experience and help to protect a greater range of people.

2. Performance

Acceptance Criteria

- The application should load within 10 seconds; this will give the application sufficient time to collect the phone’s list of installed applications, Wi-Fi information, and network information.

Justification

- By having an application that works seamlessly, the user spends less time waiting, and less time getting frustrated.

3. Reliability

Acceptance Criteria

- The application should be able to handle unexpected input without crashing or impacting the user’s performance.

Justification

- As the application is meant to run in the background, it needs to be able to handle exceptions elegantly without impacting the user’s overall experience.

Risk Analysis

Risk analysis is the process of defining and analysing the dangers posed by potential natural and human-caused adverse events [22].

Removal of required APIs – The application will require Android’s VpnService to run. If Google were to remove this service from Android, then the application would need to find a new way of running on non-rooted devices; building a custom VPN Service that takes network data directly from the network card would accomplish this.

This has been ranked as high impact because without the service the application would not run on non-rooted devices. However, it has been ranked as low risk because it is unlikely that Android will remove this feature.

Skillset required – Because the application will be running on Android, knowledge of Java is going to be required to be able to develop this app. The developer of this application has developed Android applications, and used Java, before, however not to the complexity or professionalism required for this project. To guarantee that the developer can tackle the complexity of this problem sufficient research will go into all aspects of the project. To ensure the professionalism of this project the developer will investigate standards set by Android.

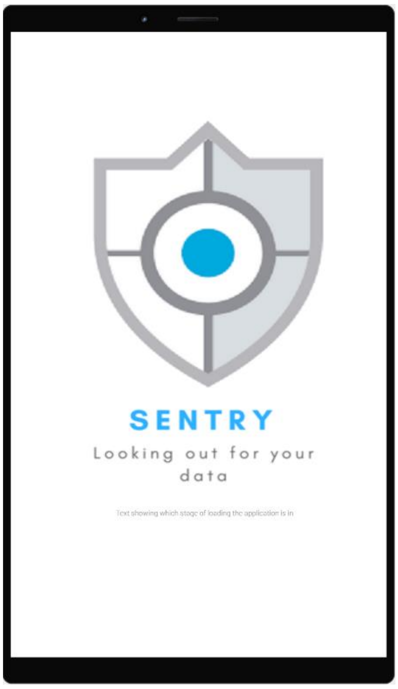
	Low Impact	**Medium Impact**	**High Impact**
Low Risk		Skillset required	Removal of required APIs.
Medium Risk			
High Risk			

Table 1 - Risk Analysis Table

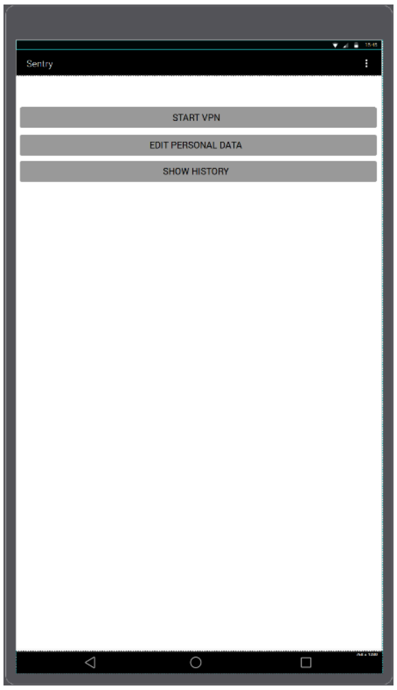
Table 1 shows an overview of the risks detected; they have been categorised by the chance of that risk happening and the impact it would have on the project.

Wireframe Designs

To develop the Android wireframe designs, the free tool Justinmind Prototyper [23] version 8.3.1 was used to keep the project's development costs to a minimum. Because of the time constraints on the project, and the developer's limited experience with the tool, it was decided that the wireframes would show only a basic layout, that the developer would use as a guide.


SENTRY
Looking out for your data
Text showing which stage of loading the application is in

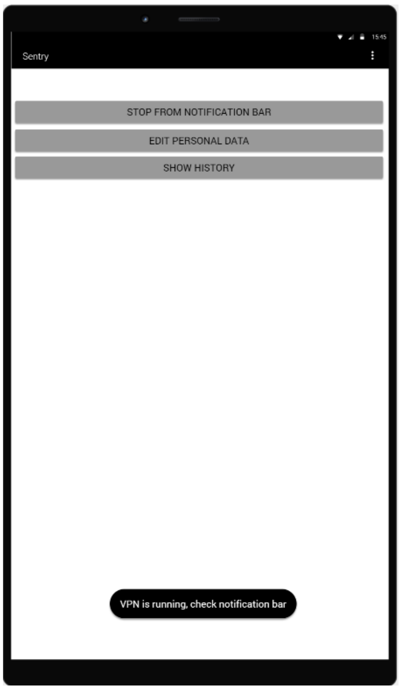
Wireframe 1 - Splash Screen



Wireframe 2 - Home Screen

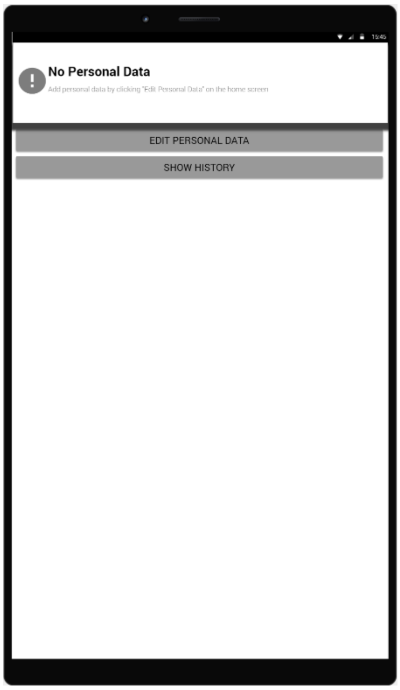
Wireframe 1 shows the splash screen for the application. While this screen is displayed, the application is gathering the list of installed application, as well as the Wi-Fi state of the phone, and the network state. The list of installed applications will be used in the Future Work section of this report. The Wi-Fi state and the network state are so the application knows when it cannot run the VPN.

Wireframe 2 shows the home screen for the application. This screen allows the user to access various aspects of the application. From here the user can start the VPN, edit their details, and view a list of data that the application has detected leaving the phone.



VPN is running, check notification bar

Wireframe 3 - VPN Started



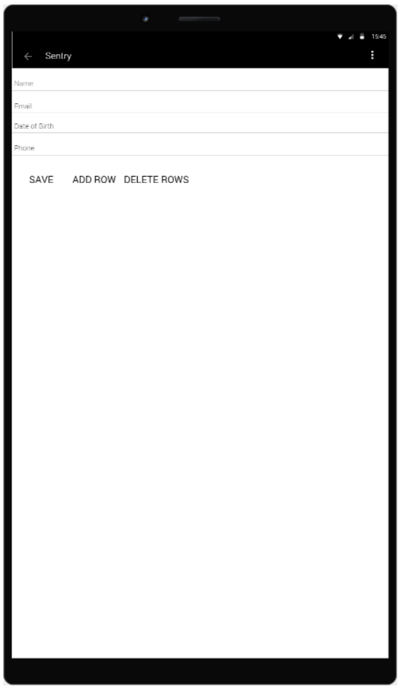
No Personal Data

Add personal data by clicking "Edit Personal Data" on the home screen

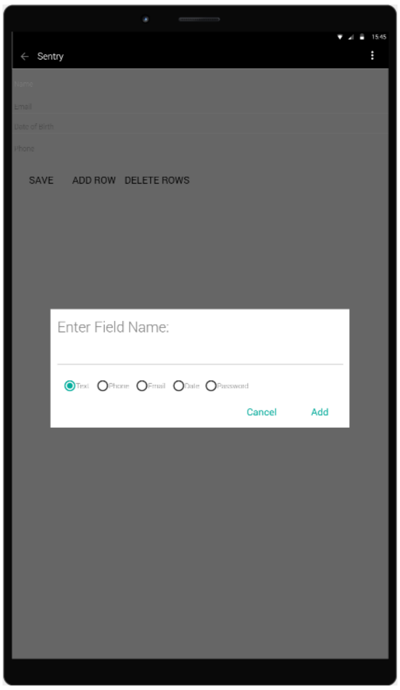
Wireframe 4 - No Data Notification

Wireframe 3 shows a notification staying that the VPN service has started successfully. This lets the user know that the application is running, and where they can find information about the VPN.

Wireframe 4 shows an error message if the user tries to start the VPN service without entering any personal data. This error message is in plain English, with steps on how to resolve the error; this stops users from getting confused or frustrated by complex error codes.



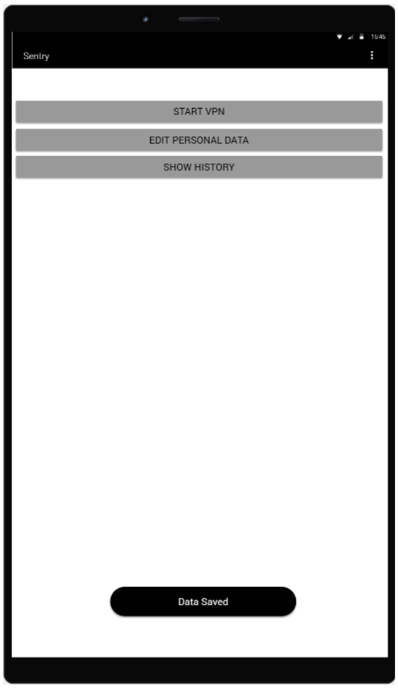
Wireframe 5 - Edit Data Page



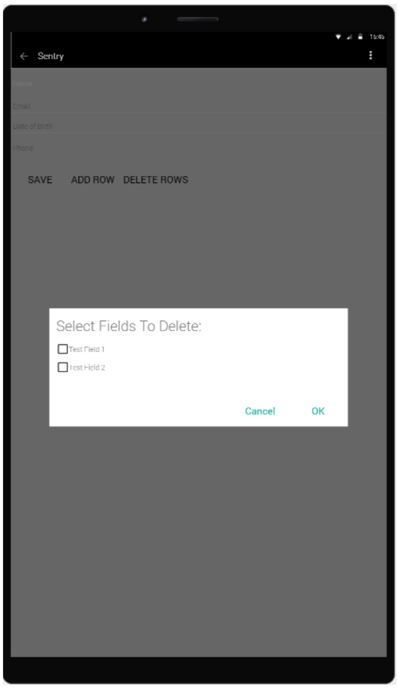
Wireframe 6 - Edit Data Add Row

Wireframe 5 shows the “Edit Data” page, from this screen the user will be able to add all the data that they want the application to check for. They can add as many rows as they require and delete rows that they do not need.

Wireframe 6 shows the “Add Row” dialogue; this dialogue allows the user to set the name of the information they would like to add. Also, they can set the data type of the new row; which allows for more accessible input.



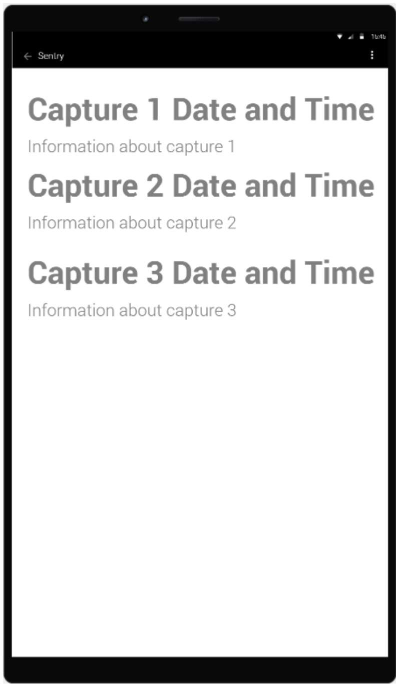
Wireframe 7 - Data Saved



Wireframe 8 - Edit Data Delete Rows

Wireframe 7 show the user a notification informing them that their data has been saved; this lets them know as there is no other feedback to pressing the “Save” button.

Wireframe 8 shows the “Delete Rows” dialogue; this dialogue allows the user to delete multiple rows if they no longer wish to use them.



Capture 1 Date and Time

Information about capture 1

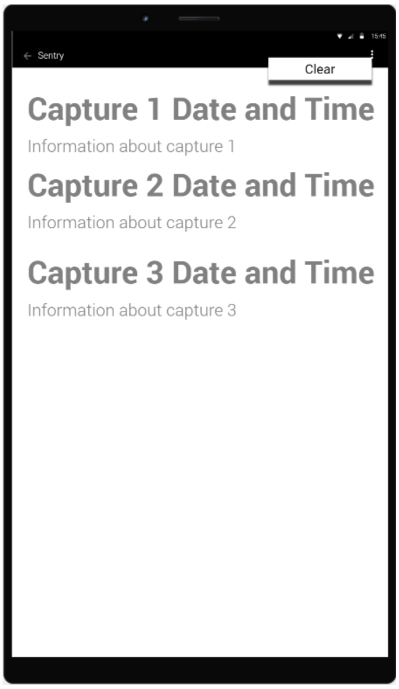
Capture 2 Date and Time

Information about capture 2

Capture 3 Date and Time

Information about capture 3

Wireframe 9 - History Page



Capture 1 Date and Time

Information about capture 1

Capture 2 Date and Time

Information about capture 2

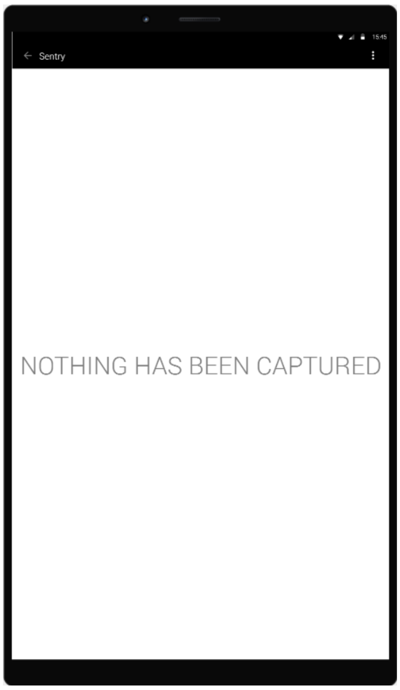
Capture 3 Date and Time

Information about capture 3

Wireframe 10 - Clear History Option

Wireframe 9 shows the “History” page; this page contains a list of every time the application has detected data leaving the device. It contains the date and time of the incident, as well as what data was captured and where it was heading.

Wireframe 10 shows the “Clear” option on the History page; this option will clear the history, removing all rows.



Wireframe 11 - Empty History Page

Wireframe 11 shows the History page without any information recorded. The text in the centre of the screen is to let the user know that nothing has been captured and that the application has not crashed.

Evaluation of Wireframe Designs

Comparison of designs against Google's quality guidelines

Because Android applications are designed to work on many types of devices, Google has defined a list of guidelines [19] that will allow for a seamless user experience no matter what device the application is on.

1. Standard Design

Requirements:

- The app does not redefine the expected function of a system icon (such as the Back button).
- The app does not replace a system icon with a completely different icon if it triggers the standard UI behaviour.
- If the app provides a customised version of a standard system icon, the icon strongly resembles the system icon and triggers the standard system behaviour
- The app does not redefine or misuse Android UI patterns, such that icons or behaviours could be misleading or confusing to users.

Evaluation:

The wireframe designs show that the application's UI obeys this guideline since the application does not redefine the use of a system icon (the Back button). Also, it does not replace any standard icons. Furthermore, the icons used are not misleading; the icons do what a user would expect them to do.

2. Navigation

Requirements:

- The app supports standard system Back button navigation and does not make use of any custom, on-screen "Back button" prompts.
- All dialogues (pop-up boxes) are dismissible using the Back button
- Pressing the Home button at any point navigates to the home screen of the device.

Evaluation:

Throughout the application, the standard Back button is used to allow the user to go to the previous page. The application uses no custom Back buttons as this can confuse the user, causing them to get frustrated. Also, the user can close any dialogue without continuing with the action, meaning that users can cancel out of something if they change their mind. Finally, the application does not override the Home button in any way; this means that the user can always get to their home screen by tapping on the Home button.

Comparison of designs against Nielsen’s Usability Heuristics

Nielsen’s heuristics are probably the most-used for user interface (UI) design and will be used to evaluate the UI of this application. Nielsen listed the principles used for this evaluation on his website [20], which he based on an article he wrote in a journal [21].

1. Visibility of system status

Definition:

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

Evaluation:

The wireframe screenshots show that this application does comply with this principle as the application gives the user feedback after every action. Wireframes 1, 3, and 7 shows that the application will give appropriate feedback to the user after completing an action. Wireframe 1 shows that the application will tell the user what it is doing, while the loading screen is displayed; showing that the application has not frozen. Wireframe 3 shows that the application notifies the user when they have started the VPN service. Wireframe 7 shows that the application will notify the user that the application has saved their data.

2. Match between system and real world

Definition:

The system should speak the users’ language, with words, phrases, and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

Evaluation:

The application uses icons that should be familiar to all who use Android applications, and the controls are intuitively placed to help users navigate through the application with ease.

The button to start the VPN service, the core aspect of the application, is placed at the top as this is the button the user will use the most. The button to edit the personal data is placed second, as this button will be used every time a personal detail changes. Finally, the button to show the application’s history is placed last, as this is likely to be the last action the user takes before finishing the application.

3. User control and freedom

Definition:

Users often choose system functions by mistake and will need a clearly marked “emergency exit” to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

Evaluation:

The screenshots show that on every screen other than the home page and the loading page, there is a Back button that allows the user to get out of any unwanted state. Furthermore, in every dialogue, there is a cancel button that can be seen and understood so that the user can get out of any dialogue without having to complete the action.

The application, however, does not include the “undo” and “redo” functionality; this functionality would be useful when deleting a row on the Edit Data page, although the designer decided not to implement it, as it is not easy to accidentally delete a row. Instead, the user is asked to confirm if they are sure they want to delete.

4. Consistency and standards

Definition:

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

Evaluation:

The screenshots show a consistent colour scheme throughout the application to provide the user with a familiarity no matter which section they are using. The design also follows Google’s conventions and guidelines [19], which means that users’ will be familiar with aspects of the application, as they follow other apps’ designs.

5. Error prevention

Definition:

Even better than a good error message is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit the action.

Evaluation:

When the user tries to start the VPN service without entering any personal data, the application will notify the user that they have not entered their data, the VPN service will also not start; this is to prevent any errors with the application checking for data that is not there.

6. Recognition rather than recall

Definition:

Minimise the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for the use of the system should be visible or easily retrievable whenever appropriate.

Evaluation:

Because the application is a service that runs in the background, there are not a lot of items or interactions for the user to remember. All the interactions in the application are labelled and are in intuitive locations, meaning that the user can easily recognise actions that they are looking for, rather than having to try and remember where the actions are.

7. Flexibility and efficiency of use

Definition:

Accelerators – unseen by the novice user – may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

Evaluation;

Due to the applications small number of features, which consist of 1 or 2 screens, there is not a need to cater for inexperienced and experienced users. Shortcuts and searches would not improve the overall ease of use, or the efficiency of the application because of the small number of screens used within the application.

8. Aesthetic and minimalist design

Definition:

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

Evaluation:

The screenshots above show the application's minimalistic design as only the information needed by the user is on the screen; this is to reduce mess on the screen that may be confusing to users. Wireframes 4, 5, and 6 shows that all data entry fields are labelled concisely to inform users of their purpose and the information that they require.

9. Help users recognise, diagnose, and recover from errors

Definition:

Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

Evaluation:

Wireframe 4 shows that this application adheres to this principle, the error message is clear and explained in plain language without using any error codes or complicated jargon. Wireframe 4 also shows the steps that the user needs to take to resolve the error and allow them to continue with the application.

10. Help and documentation

Definition:

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Evaluation:

Due to the application's small size and simple functionality, there will be no built-in support or documentation. Producing a small, first-time tutorial for new users would be beneficial to the user experience as it would show them exactly how to use the application, but as the application is well labelled and intuitive, this has not been deemed necessary.

Suggested Improvements to User Interface

1. Remove unnecessary menu items

Wireframe 10 shows the menu used to clear the history. To do so, the user needs to tap the menu button in the top right corner, then tap clear. To improve this, replace the menu item with a symbol that deletes the log. The symbol used for this would be the Delete icon found on Material Design's icon page [24]; this icon is known as the delete icon and is easily recognisable. With the menu icon replaced with the delete icon, the user would merely need to tap the delete icon, and the application will clear the log.

2. Add visualisation to the History page

Wireframe 9 shows what the History page looks like when it contains data; however, the list is full of information and it is not easy for the user to quickly see if that data was allowed or blocked; to improve this, an icon will sit to the left of each row; the Done icon from Material Design will indicate that the data was allowed, and the Block icon will indicate that the application blocked the data.

5. Implementation

The Implementation section of this report details the software tools that the developer used in the development of this Android application. It will also detail any interesting issues that arose, and how the developer implements some core features.

Feature	Description
Block sensitive data being shared unencrypted	If the application detects that the user's sensitive data is being shared without encryption, then the user has the choice to stop this data being sent
Adding Personally Identifiable Information	The user can make the application search for any data that they want. The application comes with four default rows: name, email, Date of Birth, and phone number. But the user can add any other row that they require
View a log, and clear it	The application keeps a record of each time it detects the user's data being sent. It will also record where it was being sent to, when it was being sent, and if it was blocked by the user. The user can also clear this record if they require
Lockdown	Will prevent the phone from receiving and sending any data. This feature is to give the user a straightforward way of keeping their device offline, while still being able to make phone calls
Auto Block	If this feature is enabled, it will automatically prevent any sensitive data from being shared without the user's input.

Table 2 - Application Features

Table 2 shows a list of the application's features and a brief description of what that feature does.

Development Environments

There are two main integrated development environments (IDEs) that allow for Android development; these are Android Studio [25] and Eclipse [26]. For the project Android Studio v3.1.1 will be used as it "is the official Integrated Development Environment (IDE) for Android app development" [27]. Android Studio was chosen over Eclipse due to it using IntelliJ [28], which provides many useful features such as "Smart Code Completion" and a

“Flexible build system.” Furthermore, Android Studio is free to use and completely open source. Therefore it is cost effective and can be used for the project without spending any money. Both Java and C/C++ can be used to develop Android applications, and while the developer is proficient in both languages, Java was chosen as the primary language due to it being able to make use of all the available APIs.

Interesting Features

VpnService

To make the application available to more users, the application was built upon Hexene’s LocalVPN project [12], a packet interceptor for Android built on top of VpnService [13]. This decision was made as Android’s VpnService is required so that the application can monitor the phone’s network traffic without rooting [14] the device. Hexene’s LocalVPN was used as it was well documented, had a license permitting use for this project, and meant that developer did not have to spend time developing something that was already available; giving time to work on other core features such as checking for personal information.

Data Checking

To allow for a more extendable data checking function the VpnService extracts the data from the packet and sends it to a separate function to be checked. This function works by following these steps:

1. Receive data from VpnService
2. Convert data to lower case
3. Check if the data contains any PII
    a. If it does not move on to the next packet
4. If it does, notify the user that their data has been found
    a. If the user selects “Allow” let the data carry on
    b. If the user selects “Block” stop the data being sent
5. Record the event in the history
6. Move on to the next packet.

These steps can be visualised as a flowchart in Appendix A. In step 3 of the function, an ArrayList holding the user’s information is used to check if the data contains any sensitive information; this is done by iterating through the list, and using Java’s *contains* function [29], checking if each iteration is contained in the data. The checking function has been designed so that it is completely decoupled; the Future Works section details why this has been done and the benefits that it provides.

Auto Allow/Block

In case the user is not at their phone when data has been detected, the user can set the application to either automatically allow or block the connection after a set time. This means that if data is detected, and the user is not at their phone, the internet connection will not be blocked until they are back; to implement this feature, the application follows these steps:

1. When the user has been notified that their data has been found, get the current time, and store it
2. Check the timeout variable (set in the Settings page)
3. If the timeout variable equals -1
    a. Wait for the user to respond
4. Else, get the current time
5. While the current time minus the time the notification was found is less than the timeout variable
    a. Get the current time
6. Check what the “Auto” setting is
    a. If it is “Allow” let the packet be sent
    b. If it is “Block” destroy the packet
7. Record the event in history, and add a label informing that the decision was automatic

These steps can also be viewed as a flowchart in Appendix B.

Lockdown

Lockdown mode is a way of keeping all the user’s data from leaving the phone. While this mode is enabled, all data to and from the phone will be blocked; this prevents any of the user’s sensitive information being leaked from the phone and prevents any incoming connections monitoring the phone. To make the user aware that this mode is enabled, an ongoing notification is displayed on the phone that cannot be removed until the user deactivates this setting.

Difficulties Faced

The developer’s original plan for this project was to extend the open source application Netguard [30] as it contains a lot of useful features that could be used in this project. Because this application already logs traffic, the developer would need to add a function that would be able to match this log to the user’s personal information. Due to the complexity of the code, however, this task was incredibly difficult. Furthermore, the application contained a lot of unnecessary features that would not be beneficial to the project and going through the code and removing these features would have taken much time that could have been better used on other tasks. Because of these two issues, it was decided that the application would be built using a basic implementation of Android’s VpnService, such as Hexene’s LocalVPN.

The next difficulty the developer faced was decrypting SSL/TLS [31] information in the outgoing packets. To decrypt this information the developer attempted to perform a Man-In-The-Middle (MITM) attack [32] within the application; this would work by generating, signing, and installing a certificate on the device from within the application. However, doing this in Java proved complicated, so the developer designed the application in a way that could be easily extended and will be explained in greater detail in the Future Works section.

The final difficulty the developer faced was dealing with a MITM attack; if an attacker has managed to install a certificate on the user’s device, then the application should be able to detect this and warn the user when their data is being shared. To test this, the developer used MITMProxy [33] to simulate an attack on their phone. This was done by following the steps below:

1. Start MITMProxy and set the device to use it by setting the correct details in the Wi-Fi settings
2. Open a browser on the device and go to www.mitm.it
3. Select the Android icon and follow the instructions on how to install MITMProxy’s certificate.

After following these steps, the developer could see that MITMProxy was capturing data from the device. However, the application was unable to detect MITMProxy doing this and would not notify the user that their data is being shared. It is not currently known why this is the case but is a key feature that the developer will address in the Future Works section.

6. Testing

This section of the report contains the testing done on the project, and the results, to ensure that the application meets the requirements laid out in the Specification section of this report.

Functional Testing

The functional tests were used to ensure that the application functions correctly and according to the functional requirements described in the Specification section of this report.

Functional Requirement	Implemented
The application must use Android's VpnService to analyse the user's outgoing network traffic	Yes
The user should be able to enter what sensitive data they want the application to search for	Yes
If the application detects unencrypted data leaving the phone, the user will be notified and given the option to allow or block the data	Yes
There should be a record showing the history of the user's actions	Yes

Table 3 - Functional Requirements Implemented

Table 3 shows that all functional requirements have been implemented in the application.

Test Case ID	PASS/FAIL	Reason for Fail
Func_1	Pass	N/A
Func_2	Pass	N/A
Func_3	Pass	N/A
Func_4	Pass	N/A
Func_5	Fail	Lockdown mode works, returning to normal mode does not

Table 4 - Functional Test Results

Table 4 shows the test cases performed, whether that test case passed or failed, and the reason for failing. The tests cases can be found in Appendix C

Evaluation of Functional Tests

As can be seen from Table 3, all functional requirements have been implemented, and the application does what is expected of it. However, as Table 4 shows, test case 5 failed. One of the additional features implemented, the Lockdown feature, does not function as intended and therefore requires additional work to fix the bug. The current issue with the Lockdown feature is that turning it off does not course the application to function normally. When the application's VPN Service is enabled, the user can navigate to websites as normal; when Lockdown mode is enabled, the user will no longer be able to send or receive data, as intended. The issue occurs when the user then disables Lockdown mode, the application should return to normal, with the VPN running and the user able to access to websites again. However, when the user disables Lockdown mode, the user is still unable to access websites; to resolve this, the developer will need to look into how the application checks to see if Lockdown mode is enabled, and make sure that the setting is applying correctly.

Non-Functional Testing

The non-functional tests were used to ensure that the application follows the non-functional requirements described in the Specification section of this report.

Non-functional Requirement: Usability

Acceptance Criteria

- The application should be intuitive; a user should be able to learn how to use the application without any additional help
- The user interface should follow Google's quality guidelines [19] and Nielson's Heuristics [20] [21], as well as follow a consistent theme throughout the application.

Test Method

To test this requirement, the developer explained the project requirements to his peers and let them use the application. The peers were able to navigate the application, knowing what to expect from each button, and how to use the application's core features to its full extent, with full assurance that none of their data would be leaked.

Non-functional Requirement: Performance

Acceptance Criteria

- The application should load within 10 seconds; this will give the application sufficient time to collect the phone's list of installed applications, Wi-Fi information, and network information.

Test Method

To ensure that the application met this requirement, the application was timed from application launch to the home page being shown. To make sure that the test results were reliable, the tests were repeated 10 times, and an average was taken. These tests were taken on a Samsung Galaxy S8+ and on an emulated Nexus 5.

Test Number	Time (seconds)
1	6.36
2	6.13
3	7.72
4	6.13
5	7.76
6	6.80
7	6.52
8	7.81
9	6.65
10	6.47
Average Time: 6.84 seconds	

Table 5 - Samsung Galaxy S8+ Test Times

Test Number	Time (seconds)
1	1.77
2	1.56
3	1.49
4	1.45
5	1.62
6	1.45
7	1.53
8	1.50
9	1.65
10	1.57
Average Time: 1.56 seconds	

Table 6 - Emulated Nexus 5 Test Times

Evaluation of Results

Table 5 shows that the average loading time of the application on a Samsung Galaxy S8+ was 6.84 seconds; this is within the acceptance criteria of 10 seconds and therefore passes this test. Table 6 shows that the average loading time of the application on an emulated Nexus 5 is 1.56 seconds; which is within the acceptance criteria.

The huge variation in results is brought on by two major aspects. The first aspect is the fact that the emulated Nexus 5 is running on the developer’s laptop which is using a much more powerful processor and therefore can run tasks at a higher speed. The second aspect, and the one that makes the most impact is that the Samsung device is the developer’s daily phone; which contains a lot of applications. During the loading screen, the application is getting a list of every application on the phone; the Nexus 5, being an emulated device, does not have many apps and therefore loads quicker. To get around this issue, the application should get a list of apps on the first run, then just get the number of apps on each subsequent run. If the number of apps is different to the number of apps acquired on the first run, then get all the apps again. This way the loading times should be quicker, and only take an impact when the user has installed a new application.

Non-functional Requirement: Reliability

Acceptance Criteria

- The application should be able to handle unexpected input without crashing or impacting the user's performance.

Test Method

Error Test ID	PASS/FAIL	Reason for Fail
Err_1	Pass	N/A
Err_2	Pass	N/A

Table 7 - Error Test Results

Table 7 shows the test cases performed, whether that test case passed or failed, and the reason for failing. The tests cases can be found in Appendix D

Evaluation of Results

As can be seen in Table 7, both error cases have been passed, and therefore, this requirement is satisfied.

7. Future Work

Throughout using the application, the developer has identified the following improvements that can take place:

1. Implement a method of checking for Man-In-The-Middle (MITM) attacks
2. Fix the bug in Lockdown mode, as mentioned in the Testing section.
3. Implement MITM to decrypt SSL/TLS, as mentioned in the Implementation section.
4. Move the checking function, mentioned in the Implementation section, and its requirements into its own class.

Currently, the application cannot detect if data is being sent unencrypted due to a MITM attack. This is something that would make the application much more useful and allow for greater protection of user's data.

The current solution to the bug with Lockdown mode is to turn the application's VPN off, then turn it on again. While this easy for the user to do, it is not correct, and the user should not be expected to perform extra steps that are not necessary. Fixing this bug will allow the user to turn Lockdown mode on and off as they please, without having to interrupt the application's VPN, which could cause it to miss data.

As mentioned in the Implementation section, the application cannot detect if any data is being sent via SSL/TLS. This means that the application cannot detect if other applications are sharing data that they are not supposed to. To implement this, the application would install its own certificate on the device; then, before the packet's data gets sent to the checking function, the data would be sent to a function that decrypts the SSL/TLS and converts it into plaintext.

The next stage for the checking function is to move it into its own class, completely decoupling it from the application itself. Once it is in its own class other functions such as the SSL/TLS decryption function mentioned above, can be used in this class; only being called when necessary. The final stage of this work would be to move the VpnService and the checker class into its own library, which the project application then calls. This way, the library could be uploaded to a public repository where others can make use of it or improve it.

8. Conclusions

The main aim of this project has been met, as a proof of concept Android application has been developed that can detect if an application transmits unencrypted user data such as their name, email, Date of Birth, or phone number. Also, the application has been developed further, and useful features have been implemented that can help the application appeal to a wider audience.

The main reason for me choosing this project was because of my interest in cybersecurity, and the insight that this project would give me. Working on this project has furthered my interest in cybersecurity and has helped me to secure a graduate job in the field. Another reason for me choosing this project was to help increase my expertise in Android development, I believe that I have done this as when I started the project my only experience of Android development was small applications that I had made to see how it was done; now I have made excellent progress, making an application that uses complex features and services.

There is still an issue in the application with the Lockdown feature, as mentioned in the Testing section and the Future Work section, meaning that not everything is running perfectly. It has, however, been decided that it is not a core feature and can be worked on later. As all other requirements have been met, both functional and non-functional, I would deem this project a success.

9. Reflection on Learning

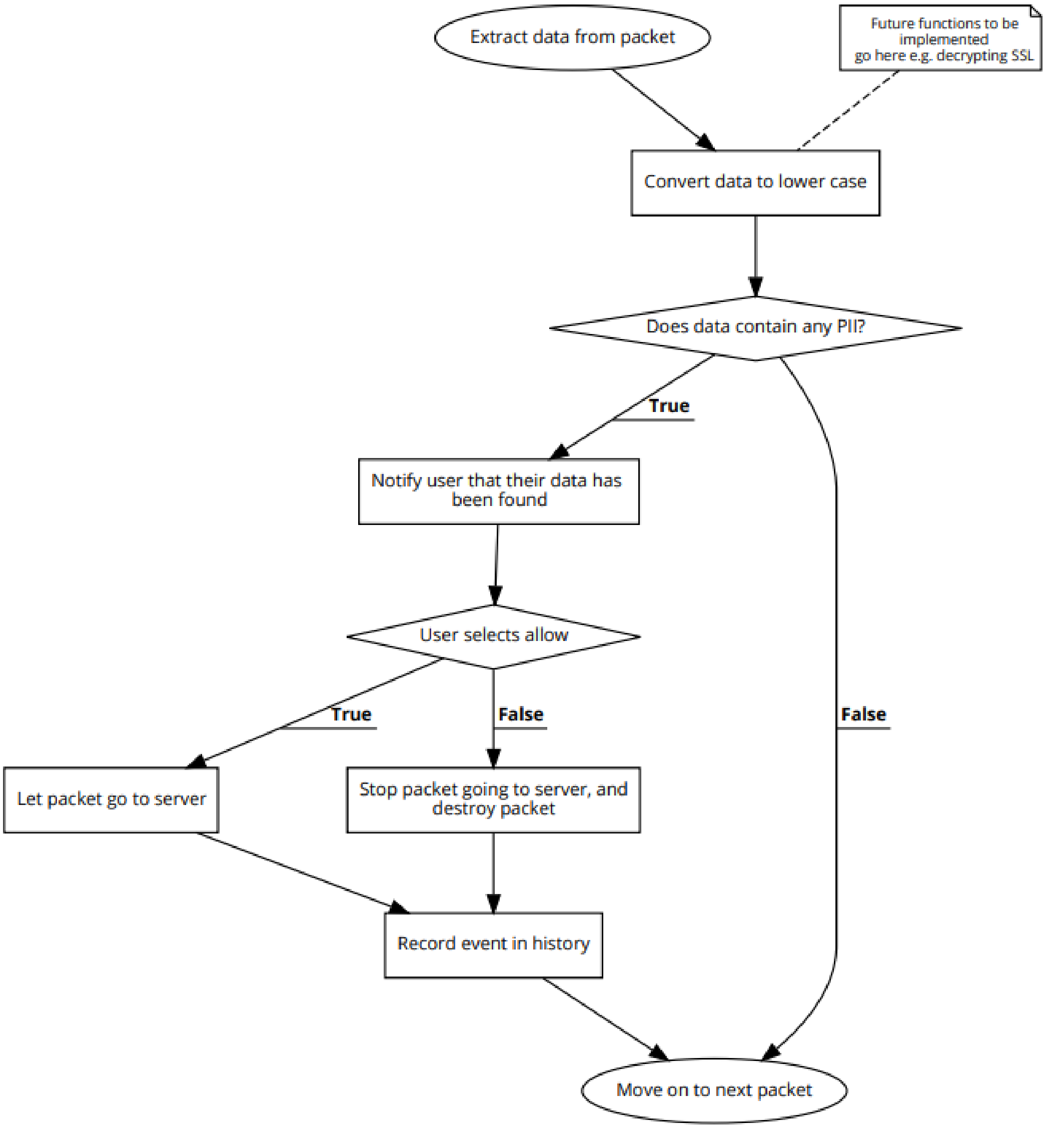
Developing this Android application has provided me with an excellent opportunity that improved my professional and technical skills, which will aid me in my future as a cybersecurity consultant. Since the application is of an original idea, and there are no others with the same functionality, I have shown an initiative in solving this problem that shows I am able to come up with intuitive ideas that can be applied directly to an issue.

In addition, I feel that my project management skills such as planning, and time management have improved since starting this project. During my placement year, I was taught that the best way to tackle a new problem was to get stuck in with it and see what you can get done immediately. This meant that I would try and implement features and functionality without giving it much thought, which meant that I would often get stuck when something was not working, and I could not remember what I had done previously. I learnt very quickly that this approach would not work for this sort of project, as such I learnt how to plan my approach; taking the problem and converting it into small, manageable tasks that I would be able to complete. An example of this planning and preparation would be the initial plan I had to do for this project, the initial plan had me clearly lay out my aims and objectives, and a Gantt Chart that enabled me to be able to work towards my goal at an appropriate pace.

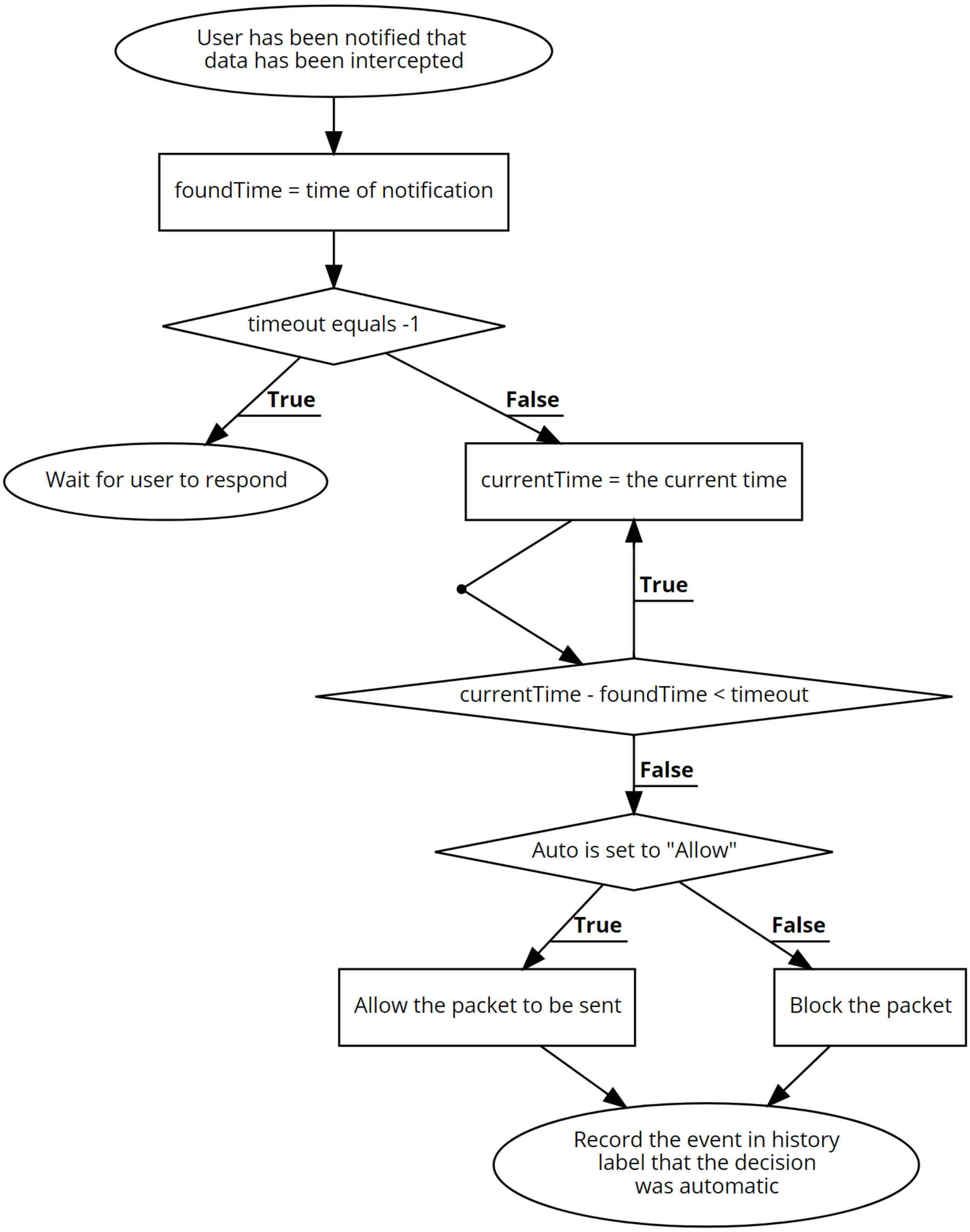
Finally, I have gained knowledge in communicating properly when working on a project. I now understand that communicating properly is paramount to a project’s success; when I now come across a problem that I cannot immediately solve, I must first remember to explore my own options on how I can solve this problem, before requesting help from those with expertise.

Due to the reasons mentioned in this section, I think that the experience I have gained from doing this project has equipped me with an outstanding set of skills that has enabled me to get my graduate job in a field that I enjoy and with a company that I am interested in.

10. Appendix A – Data Checker Flowchart


Extract data from packet
Future functions to be implemented go here e.g. decrypting SSL
Convert data to lower case
Does data contain any PII?
True
Notify user that their data has been found
User selects allow
True
False
False
Let packet go to server
Stop packet going to server, and destroy packet
Record event in history
Move on to next packet

11. Appendix B – Auto Flowchart


User has been notified that data has been intercepted
foundTime = time of notification
timeout equals -1
True
False
Wait for user to respond
currentTime = the current time
True
currentTime - foundTime < timeout
False
Auto is set to "Allow"
True
False
Allow the packet to be sent
Block the packet
Record the event in history label that the decision was automatic

12. Appendix C – Functional Test Cases

Test Case ID: Func_1		Purpose: To check that the application meets functional requirement 1	
Environment: Samsung Galaxy S8+ running Android 8.0.0 (API 26)			
Preconditions: Phone is connected to Android Studio, and the Logcat is open			
Step	Process	Result	Pass/Fail
1	Press button “START VPN”	VPN started; Icon appears in notification bar; Notification appears on screen	Pass
2	Check Logcat	Data will be displayed in the Logcat containing the packet information	Pass

Test Case ID: Func_2		Purpose: To check that the application meets functional requirement 2	
Environment: Samsung Galaxy S8+ running Android 8.0.0 (API 26)			
Preconditions: None			
Step	Process	Result	Pass/Fail
1	Press button “EDIT PERSONAL DATA”	Screen changes to “Edit Data” screen, can see default inputs.	Pass
2	Press “Add Row” button	Dialogue opens, user can enter field name, and select data type	Pass
3	Enter Field Name “Test” Leave “Text” as data type Press OK	Dialogue should disappear A new row should be added with the name “Test”	Pass
4	Press “Delete Rows” button	Dialogue opens “Test” appears with a checkbox	Pass

5	Select checkbox next to “Test” Press OK	Dialogue closes Row named “Test” disappears	Pass
6	Fill in all rows on screen Press “Save”	User will be taken back to Home screen Notification pops up notifying user of save	Pass
7	Press button “EDIT PERSONAL DATA”	Data entered in step 6 remains	Pass

Test Case ID: Func_3		Purpose: To check that the application meets functional requirement 3	
Environment: Samsung Galaxy S8+ running Android 8.0.0 (API 26)			
Preconditions: Test Case 2 has been completed, and there is information in “Edit Personal Data”			
Step	Process	Result	Pass/Fail
1	Press button “START VPN”	VPN started; Icon appears in notification bar; Notification appears on screen	Pass
2	Open an application that sends unencrypted data. Send a piece of information that is saved in Edit Data	A notification appears informing the user that their data has been detected	Pass
3	Tap Allow	The notification disappears, and the packet is sent on its way	Pass
4	Repeat Step 2	A notification appears informing the user that their data has been detected	Pass

5	Tap Block	The notification disappears, and the packet is destroyed	Pass

Test Case ID: Func_4		Purpose: To check that the application meets functional requirement 4	
Environment: Samsung Galaxy S8+ running Android 8.0.0 (API 26)			
Preconditions: Test Case 3 has been completed			
Step	Process	Result	Pass/Fail
1	Select “SHOW HISTORY”	The history page opens, there are at least 2 rows. 1 row for step 3 in test case 3. 1 row for step 5 in test case 3	Pass
2	Tap the bin in the top right corner	The history page should be cleared, and a message should be displayed to the user	Pass

Test Case ID: Func_5		Purpose: To check that “Lockdown” mode works	
Environment: Samsung Galaxy S8+ running Android 8.0.0 (API 26)			
Preconditions: None			
Step	Process	Result	Pass/Fail
1	Open the menu by tapping the icon in the top right-hand corner	A drop-down menu appears	Pass
2	Tap “Settings”	The Settings page will appear	Pass
3	Tap “Lockdown”	The Lockdown switch should be checked and change colour	Pass

4	Go back to the home page and tap “START VPN”	VPN started; Icon appears in notification bar; Notification appears on screen; Ongoing notification appears for Lockdown mode	Pass
5	Open web browser and try navigation to a website	Website will not load	Pass
6	Tap “Lockdown” notification	Settings page appears	Pass
7	Tap “Lockdown”	The Lockdown switch should be unchecked and change colour	Pass
8	Repeat step 5	User will be taken to website	Fail

13. Appendix D – Error Testing

Test Case ID: Err_1		Purpose: To check the application can handle unknown IP addresses	
Environment: Samsung Galaxy S8+ running Android 8.0.0 (API 26)			
Preconditions: Device is connected to a local Wi-Fi network			
Step	Process	Result	Pass/Fail
1	Press button "START VPN"	VPN started; Icon appears in notification bar; Notification appears on screen	Pass
2	Open web browser; Navigate to unused local IP address	User will not be able to see webpage, as there is nothing to see	Pass
3	Navigate to known webpage https://www.google.co.uk	User will be shown the Google search engine	Pass

Test Case ID: Err_2		Purpose: Make sure user is informed if they have not entered any personal information	
Environment: Samsung Galaxy S8+ running Android 8.0.0 (API 26)			
Preconditions: "Edit Data" page is empty			
Step	Process	Result	Pass/Fail
1	Press button "START VPN"	VPN started; Icon appears in notification bar; Notification appears on screen; Urgent notification appears informing user they have not entered any information	Pass

14. References

[1] Statista, “Smartphone OS market share worldwide 2009-2017,” 02 2018. [Online]. Available: https://www.statista.com/statistics/263453/global-market-share-held-by-smartphone-operating-systems/. [Accessed 09 05 2018].

[2] Statista, “Google Play Store: number of apps 2009-2017 | Statistic,” 12 2017. [Online]. Available: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/. [Accessed 09 05 2018].

[3] Kaspersky, “Top 7 Mobile Security Threats: Smart Phones, Tablets, & Mobile Internet Devices – What the Future has in Store,” Undated. [Online]. Available: https://www.kaspersky.co.uk/resource-center/threats/top-seven-mobile-security-threats-smart-phones-tablets-and-mobile-internet-devices-what-the-future-has-in-store. [Accessed 09 05 2018].

[4] C. Gibler, J. Crussell, J. Erickson and H. Chen, “AndroidLeaks: Automatically Detecting Potential Privacy Leaks in Android Applications on a Large Scale,” *Trust and Trustworthy Computing,* vol. 7344, pp. 291-307, 2012.

[5] R. Jones, “Identity fraud reaching epidemic levels, new figures show,” 23 August 2017. [Online]. Available: https://www.theguardian.com/money/2017/aug/23/identity-fraud-figures-cifas-theft. [Accessed 31 January 2018].

[6] J. Boyles, A. Smith and M. Madden, “Privacy and Data Management on Mobile Devices,” 5 September 2012. [Online]. Available: http://www.pewinternet.org/2012/09/05/privacy-and-data-management-on-mobile-devices/. [Accessed 31 January 2018].

[7] Wireshark, “Introduction,” Undated. [Online]. Available: https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html#ChIntroWhatIs. [Accessed 04 05 2018].

[8] Wireshark, “About Wireshark,” Undated. [Online]. Available: https://www.wireshark.org/about.html. [Accessed 09 05 2018].

[9] I. Baggili, “UNHcFREG Releases Datapp - An application for testing if data sent from your mobile apps is unencrypted,” 11 05 2015. [Online]. Available: https://www.unhcfreg.com/single-post/2015/05/11/UNHcFREG-Releases-Datapp-An-application-for-testing-if-data-sent-from-your-mobile-apps-is-unencrypted. [Accessed 09 05 2018].

[10] T. Simonite, “How to Detect Apps Leaking Your Data,” 10 08 2012. [Online]. Available: https://www.technologyreview.com/s/428772/how-to-detect-apps-leaking-your-data/. [Accessed 05 09 2018].

[11] Grey Shirts, “Google Play | Packet Capture,” 23 10 2017. [Online]. Available: https://play.google.com/store/apps/details?id=app.greyshirts.sslcapture&hl=en. [Accessed 09 05 2018].

[12] M. Naufal, “LocalVPN,” 29 06 2017. [Online]. Available: https://github.com/hexene/LocalVPN. [Accessed 03 05 2018].

[13] Android.com, “VpnService | Android Developers,” Undated. [Online]. Available: https://developer.android.com/reference/android/net/VpnService.html. [Accessed 17 April 2018].

[14] J. Hildenbrand, “Everything you need to know about rooting your Android,” 6 June 2016. [Online]. Available: https://www.androidcentral.com/root#A. [Accessed 21 April 2018].

[15] Tutorials Point, “UML - Activity Diagrams,” 08 01 2018. [Online]. Available: https://www.tutorialspoint.com/uml/uml_activity_diagram.htm. [Accessed 30 04 2018].

[16] A. Stellman and J. Greene, “Applied software project management,” Sebastopol, CA, O'Reilly, 2008, p. 110.

[17] Techopedia.com, “What is a Packet? - Definition from Techopedia,” [Online]. Available: https://www.techopedia.com/definition/5380/packet. [Accessed 17 April 2018].

[18] Google, “Notifications Overview | Android Developers,” 2018. [Online]. Available: https://developer.android.com/guide/topics/ui/notifiers/notifications.html#importance. [Accessed 19 April 2018].

[19] Google.com, “Core app quality | Android Developers,” 25 05 2018. [Online]. Available: https://developer.android.com/docs/quality-guidelines/core-app-quality. [Accessed 29 04 2018].

[20] J. Nielsen, “10 Heuristics for User Interface Design: Article by Jakob Nielsen,” 01 01 1995. [Online]. Available: https://www.nngroup.com/articles/ten-usability-heuristics/. [Accessed 29 04 2018].

[21] J. Nielsen, “Enhancing the explanatory power of usability heuristics,” *Proceedings of the SIGCHI conference on Human factors in computing systems celebrating interdependence - CHI '94*, pp. 152-158, 1994.

[22] M. Rouse, “What is risk analysis? - Definition from WhatIs.com,” 11 October 2010. [Online]. Available: https://searchmidmarketsecurity.techtarget.com/definition/risk-analysis. [Accessed 21 April 2018].

[23] Justinmind.com, “Prototyping tool for web and mobile apps - Justinmind,” Undated. [Online]. Available: https://www.justinmind.com/. [Accessed 29 04 2018].

[24] Google, “Material icons - Material Design,” Undated. [Online]. Available: https://material.io/icons/. [Accessed 30 04 2018].

[25] Google, “Download Android Studio and SDK Tools | Android Developers,” Undated. [Online]. Available: https://developer.android.com/studio/. [Accessed 04 30 2018].

[26] Eclipse Foundation, “Eclipse Oxygen | The Eclipse Foundation,” Undated. [Online]. Available: https://www.eclipse.org/oxygen/. [Accessed 30 04 2018].

[27] Google, “Meet Android Studio | Android Developers,” 20 04 2018. [Online]. Available: https://developer.android.com/studio/intro/. [Accessed 30 04 2018].

[28] JetBrains, “IntelliJ IDEA: The Java IDE for Professional Developers by JetBrains,” 27 03 2018. [Online]. Available: https://www.jetbrains.com/idea/. [Accessed 30 04 2018].

[29] Oracle, “Java Platform SE 7,” 27 02 2018. [Online]. Available: https://docs.oracle.com/javase/7/docs/api/. [Accessed 03 05 2018].

[30] M. Bokhorst, “Netguard,” 28 04 2018. [Online]. Available: https://github.com/M66B/NetGuard. [Accessed 04 05 2018].

[31] R. Barnes, M. Thomson, A. Pironti and L. A, “RFC 7568 - Deprecating Secure Sockets Layer Version 3.0,” 06 2015. [Online]. Available: https://tools.ietf.org/html/rfc7568. [Accessed 05 05 2018].

[32] Symantec, “What Is A Man In The Middle Attack?,” Undated. [Online]. Available: https://us.norton.com/internetsecurity-wifi-what-is-a-man-in-the-middle-attack.html. [Accessed 04 05 2018].

[33] mitmproxy, “mitmproxy,” Undated. [Online]. Available: https://mitmproxy.org/. [Accessed 10 05 2018].