****

**University of Wollongong**

School of Computing and Information Technology

Faculty of Engineering & Information Sciences

Bachelor of Computer Science (Digital System Security)

SIM Session 2, 2019

**CSCI321 – Project**

Technical Document

|  |  |
| --- | --- |
| **Project Topic:** | Android Network Sniffer |
| **Submitted To:** | Mr. Premarajan P |
| **Submitted By:** | Kan Kar Jun Alvin (5709659)  Liao Weisheng Wilson (6212591)  Nyein Soe (5710959) |
| **Group:** | SSP19/2B |
| **Date:** | 28/8/2019 |

**Contents**

1. [Executive Summary](#ExSummary)  4
2. [Introduction](#Intro) 5

2.1. Project Background 5

2.2. Purpose 7

2.3. Roles and Responsibilities 9

1. [Market Research](#Market)  10

3.1. Common Existing Products 12

3.2. Comparison and Overview of Products 16

1. [Non-Functional Requirements](#Nonfunc) 17

4.1. Security Requirements 17

4.2. Software Quality Attributes 17

4.3. Performance Requirements 17

1. [System Functional Features](#SystemFunc) 18

5.1. Program overall flow diagram 19

5.2. Description of the functions 20

5.3. Refined Program overall flow diagram 21

5.4. Revised Description of the functions 22

1. [Requirement Analysis](#RequireAnaly) 23

6.1. Use Case Diagram 23

6.2. Use Case Identification 24

6.3. Refined Use Case Diagram 28

6.4. Refined Use Case Identification 29

1. [System Design](#SystemDesign) 35

7.1. Overview 35

1. [Implementation](#Implementation) 37

8.1. Development methodology 37

8.2. Development Tools 38

8.2.1. Programming Language 38

8.3. Development Platforms 38

8.3.1. Android Studio Developer Kit 38

8.3.2. Software as a Service/Product 38

1. [Risk Assessment](#RiskAssess) 39
2. [Limitation and Constraints 40](#LimitConstr)
3. [Prototype](#Proto)

11.1. Overview 41

11.2. Layout 41

1. [Final Product](#finalProd)  44

12.1 Overview 44

12.2. Functionalities 46

12.2.1. Device Scan 46

12.2.2. Packet Scan 47

12.2.3. Network Status Functions 47

12.2.4. Help Function 47

1. [Conclusion](#Conclusion)  48
2. [Annexes](#Annexes) 49

14.1. Meeting minutes 49

14.2. Project Diary 56

14.3. Project Planning Gantt Chart 57

14.4. Website and Blog 58

14.5. Test Cases 58

1. **Executive Summary**

The main objective of this project is to create a network sniffer in an Android OS, where the program is able to listen to the network communication and break down the packets to readable formats. Our main goal is to create an executable program or even as an android application which will then provide easy user-access for the targeted users, especially for those whose roles are related to the IT/Network field so that the program can be used to perform analysis on the networks, packets and etc.

For instance, a WEP program crack can be implemented to provide a more robust implementation.

1. **Introduction**

***2.1. Project Background***

In general, network sniffing involves using sniffer tools that enable real-time monitoring and analysis of data packets flowing over computer networks. It can be a hardware device or a separate software program or a combination of both. It is also called as packet sniffing, snoop, network probe, packet analyzer, network analyzer, or protocol analyzer. Basically, it examines traffic on the network and takes snapshot copies of the packet data.

**How does it work?**

Packet sniffers work by intercepting and logging network traffic that they can 'see' via the wired or wireless network interface that the packet sniffing software has access to on its host computer.

On a wired network, what can be captured depends on the structure of the network. A packet sniffer might be able to see traffic on an entire network or only a certain segment of it, depending on how the [network switches](https://www.lifewire.com/definition-of-network-switch-817588) are configured, placed, etc. On wireless networks, packet sniffers can usually only capture one channel at a time unless the host computer has multiple wireless interfaces that allow for multichannel capture.

Once the raw packet data is captured, the packet sniffing software must analyze it and present it in human-readable form so that the person using the packet sniffing software can make sense of it. The person analyzing the data can view details of the 'conversation' happening between two or more nodes on the network. Network technicians can use this information to determine where a fault lies, such as determining which device failed to respond to a network request.

Hackers can use sniffers to eavesdrop on unencrypted data in the packets to see what information is being exchanged between two parties. They can also capture information such as passwords and authentication tokens (if they are sent in the clear). Hackers can also capture packets for later playback in replay, man-in-the-middle, and packet injection attacks that some systems may be vulnerable to.

In addition, all networks use “packets” to send data. The size of a data file may be quite huge and sending it as a single packet would strain the network and cause congestion. The integrity of the data may also be affected. Hence, whenever a user sends a file or an email it is broken up into smaller parts or packets and then sent to the destination. Each data packet includes:

1. Destination address
2. Number of packets
3. Reassembly order
4. Source address

Once the data packet gets to its destination, all the footers and headers are removed, and the data is reconstituted. Every network and computer have a filter that discards any packets that aren’t addressed to it.

In Network Sniffing, the Packet sniffers intercept and log the network traffic through the packet sniffing software. Based on the capability of the software, a packet sniffer may be able to access traffic on the complete network or just a part of the network.

The packet sniffing software then analyses and converts it into a user-friendly form. This enables the hacker to function as a man in the middle and access the details of the communication between two parties/organizations. Unencrypted data is very vulnerable, and hackers can gain access to passwords, authentication tokens and other credentials. Hackers use the captured packets for man-in-the-middle and packet injection attacks.

Network sniffing, like SSL sniffing is widely used by hackers, but it can also be used for network troubleshooting. Our product includes this and other security options to ensure the safety of your information, so contact us today to learn more.

***2.2. Purpose***

Typical use of network sniffer is to analyze network traffic and bandwidth utilization, so that underlying troubles in the network can be identified. There are, however, two directional usages of sniffer which have coexisted since it was first produced:

Positive usage of a sniffer is also its regular usage, which has as its objective the desire to maintain the network and keep it working normally.

1. Capturing packets;
2. Recording and analyzing traffic;
3. Decrypting packets and displaying in clear text;
4. Converting data to readable format;
5. Showing relevant information like IP, protocol, host or server name and so on.

Not all packet sniffing software products have the same functions; some sniffers can analyze hundreds of protocols whereas others can only deal with one or two. The most common protocols analyzed by sniffer are TCP/IP, IPX, DECNet-Ordinarily, a sniffer is used as assistant tool of the network engineer for monitoring and analyzing a network, detecting intrusion, controlling traffic or supervising network activity. IT should be noted that such features may also be utilized by hackers as a snooping tool to break into other computers.

Negative usage of a sniffer is well known as its harms to network security:

1. Catching password, which is the main reason for most illegal uses of sniffing tool;
2. Capturing special and private information of transactions, like username, credit ID, account, and password;
3. Recording email or instant message and resuming its content;
4. Some Sniffers can even modify target the computer's information and damage the system;
5. Interrupting the security of a network or to gain higher level authority.

With more and more hackers using packet sniffers, it has become one of the most important tools in the defense of cyber-attacks and cyber-crime.

*Monitoring network usage* – Packet sniffers are great at monitoring the network usage at any given time, helping Network Managers identify whether a particular network is normal or congested. Also, making it possible to identify bottlenecks within the network and identify and improve the performance with infrastructure upgrades.

*Identifying problems* – As mentioned earlier, packet sniffers can identify network-related issues. This is possible because a packet sniffer can analyze the conversation between two or more nodes in a network. So, in the event of a network error, the information captured by the packet sniffer can be used to identify the erroneous packets and pinpoint the node that failed to answer the request(s). Making it easy to identify faulty devices within the network in an efficient manner and providing the ability to take swift corrective actions.

*Detecting security loopholes* – A disturbing fact about packet sniffers is their ability to work as spying tools. They also help the good guys, such as your Network Manager, by testing the vulnerabilities of a network. Once these vulnerabilities are detected, it is easier to remove the loopholes thus preventing the possibilities of hacking attempts.

***2.3. Roles and Responsibilities***

|  |  |
| --- | --- |
| **Name** | **Roles/Responsibilities** |
| NYEIN SOE | Team Leader, Researcher, Implementation Designer, Tester |
| LIAO WEISHENG WILSON | Lead Developer, Implementation Designer, Tester |
| KAN KAR JUN ALVIN | Backup Developer, Implementation Designer, Planner, Technical Writer |

|  |  |  |  |
| --- | --- | --- | --- |
| **Roles/Responsibilities** | Nyein Soe | Liao Weisheng Wilson | Kan Kar Jun Alvin |
| Team Leader | Y |  |  |
| Research | Y | Y | Y |
| Documentation | Y | Y | Y |
| System Designer |  | Y | Y |
| Lead Programmer |  | Y |  |
| System Tester | Y | Y | Y |

1. **Market Research**

Based on our research, we have found several packet/network sniffing tools which are currently being used and they are as mentioned below.

● SolarWinds Packet Analysis Bundle

● WireShark

● PRTG Network Monitor

● Steel Central Packet Analyzer

● Tcpdump

● Network Miner

● Kismet

● Fiddler

● EtherApe

● Packet Capture

Each type of tools has their own functionalities and to use some of them, the user will have to pay in order to gain access to the full feature of the respective tools.

In addition, there are several types of sniffers and they are as mentioned below.

● *ARP Sniffers:* In ARP sniffing, the network traffic is not sent to the hosts, but it is sent to the ARP cache of both network hosts, which is then forwarded to the network administrator. ARP Sniffers sniff the data when it is in the ARP cache. The ARP sniffer is popular among hackers, as the data captured in the cache allows them to create a map of the IP addresses and their associated MAC addresses. This map allows the hackers to conduct packet-spoofing attacks, search for router-based vulnerabilities and ARP poisoning attacks.

● *IP Sniffers:* IP Sniffers sniff all the data that corresponds with a specific IP address filter. This allows capturing of specific data packets for analysis and diagnosis. This method is also popular among hackers, who use it for stealing data and also for stealing the TCP session. They also use this to create fake TCP sessions, act as a man-in-the-middle and unleash MitM attacks.

● *MAC sniffers:* MAC sniffers sniff all the data that corresponds with a specific MAC address filter.

● *LAN sniffer*s: These are deployed on internal LANs and they have the capability to scan the complete IP range.

● *Protocol sniffers:* These are used to sniff data that are related to the network protocols that are used on the network.

● *Web password sniffers:* Hackers steal http sessions parse them to acquire login credentials, user IDs and passwords. While many websites protect their external facing web pages/websites with SSL, they do not use SSL or use lesser secure encryption for their internal web pages. Hackers can exploit this vulnerability.

Some of the key aspects being used in network sniffing include, PCAP, WinPCAP. These programs are used as an application programming interface (API) for packet-capturing. Sniffers also need to place the computer's network card in *promiscuous mode* to receive all the data that passes by, not just packets addressed to it.

***3.1. Common Existing Products***

**WireShark**

WireShark is a windows-based network sniffing tool that shows all the network information of the network the device is currently connected to. WireShark is able to provide details like the resolved addresses, protocol hierarchy status, connections, endpoints, packet lengths and the IO graphs.

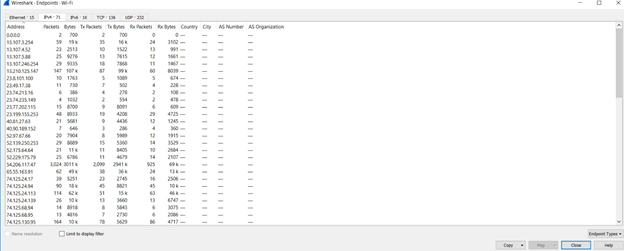


Figure 3.1.1. Features of WireShark

**PRTG**

PRTG is an android based app that provides information on the devices and services that are connected to the network. These include information on servers or FTP services that are on the network. Example of these information are the ping, model number, serial number of the devices, and the number of files on the FTP server.

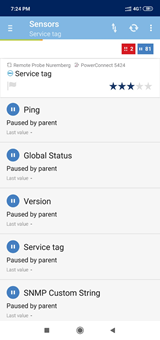
****

Figure 3.1.2. Features of PRTG

**Packet Capture**

Packet capture is an android based app that provides information on the usage of the network by the various apps and processes on the device. This information includes the security standards, MAC address, as well as the server details.

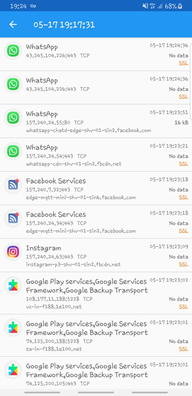
****

Figure 3.1.3. Features of Packet Capture

**Network Miner**

Network Miner is also a windows-based network sniffing tool that shows the packet information of the network that the device is connected to. It provides information like the packets sent; these packets show associated packets and if there are ongoing sessions as well as the packet size.

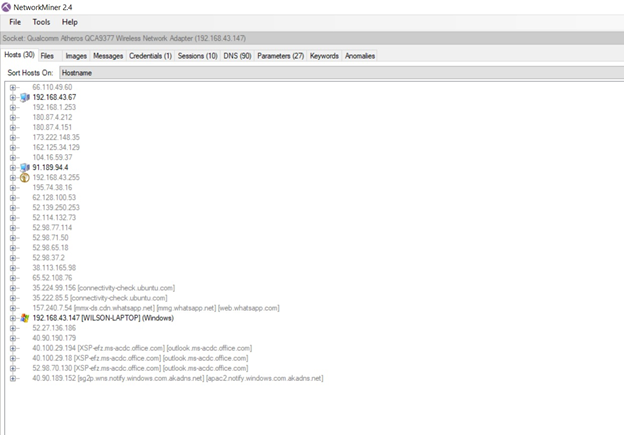
****

Figure 3.1.4. Features of Network Miner

***3.2. Comparison and Overview of Products***

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Products\Functions** | ***Show details?*** | ***Platform*** | ***Capture from what source?*** | ***Separate packets by protocols?*** | ***Select adapter?*** | ***IO Graph*** | ***Purpose*** | ***Paid Features?*** |
| ***Wireshark*** | Y | Windows | Wifi | Y | N | Y | IT Professionals to conduct network monitoring | Y |
| ***PRTG*** | Y | Android | Wifi | N | N | N | To casually monitor the websites that has been connected the most | N |
| ***Packet Capture*** | Y | Android | VPN | N | N | N | To casually monitor the websites that has been connected the most | N |
| ***Network Minor*** | Y | Windows | Wifi | Y | Y | N | IT Professionals to conduct network monitoring | Y |

Table 3.2.1. Function comparison of different Products

1. **Non-Functional Requirements**

***4.1. Security Requirements***

For our program, the user will require a rooted android device so that the application can be installed. In addition, the application may be free, but we have implemented in a way that it is only compatible with some of the android devices (e.g. Samsung Galaxy Note 2). This purpose is implemented in such a way that for instance, in an organization, only the IT department is in charge of all the rooted android devices with the installed application. Thus, only these employees or users in this IT department will be able to use. If a third-party user wants to utilize the application, he will have to go through the authorities in the department itself and will have to fill out some registration form so that the authorities can pass the device to these users and only then, they will be able to use our application through these devices.

In addition, one of the features our application provides is to check the port validity. This feature works in a way that once the IP packets has been sniffed, the user will be able to check by clicking onto the sniffed data itself and this will then bring the user to the respective website where they can check for any anomalies in the port, address, IP, etc., and the same time checking if the port is valid and secure.

***4.2. Software Quality Attributes***

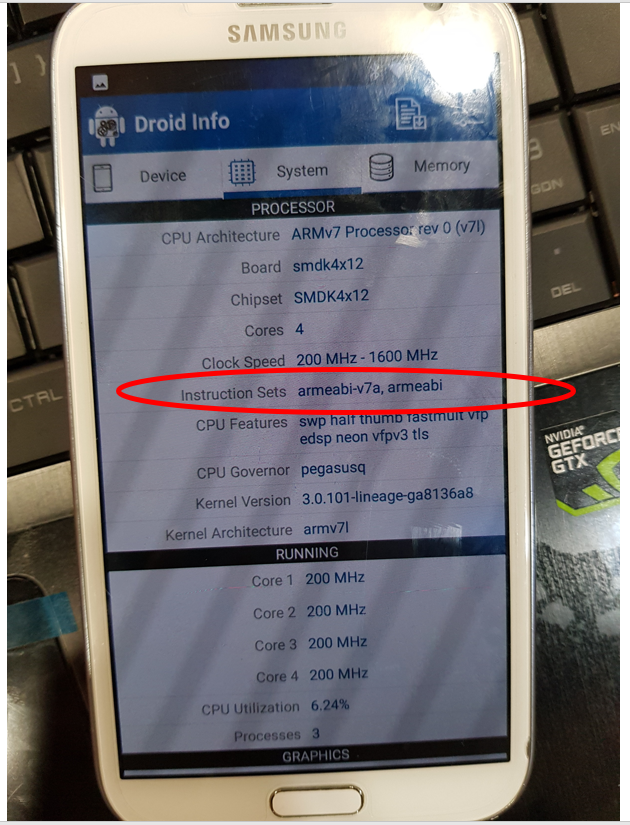
Despite the fact that the users having to go through account registration, the program will be made available for anyone at any point of time. And also, we are aiming to make the program for the IT department so that any IT professionals or even interns in such field will be able to use this program. In addition, we are also aiming to implement in such a way that the program can provide ease of access for all the users so that they will not have any problems in using it.

***4.3. Performance Requirements***

For our program, only certain devices are compatible to run. A recommended device is using Samsung Galaxy Note 2 and this device needs to be rooted in order for the application be installed and run normally [See Appendix A on the Website]. Although the rooting procedure may be a bit tedious, but the network sniffing application is implemented as a user-friendly program. A basic instruction on how to use the application will be provided on our website as well. For the rooting, the user is to make sure that he/she is installing a third party software Armeabi, where the device should support Armeabi Architectures.

* + - Armeabi Architecture which is used so that the device can support such architecture of the CPU.

Website:  [**android-network-sniffer.webnode.com**](http://android-network-sniffer.webnode.com/)



1. **System Functional Features**

For the system or program that we are about to implement, we have decided to come up with a series of functions followed by one another. For instance, once the user opens the application, the background functions will download necessary resources. Then the user will have to connect to a certain. network, fill in the IP address which he/she wishes to scan/sniff followed by displaying the data, filtering and sorting the data so on and so forth.

In addition, once the results have come out, they will have the options such as sorting the data, displaying them in certain views, saving the statistics and even exporting these data. Therefore, to provide further aid on the description of the system functionalities, we have come up with an overall program flow diagram and it is as shown below.

***5.1. Program overall flow diagram***

Attached below is the overall flow diagram of the basic program that we will be implementing.

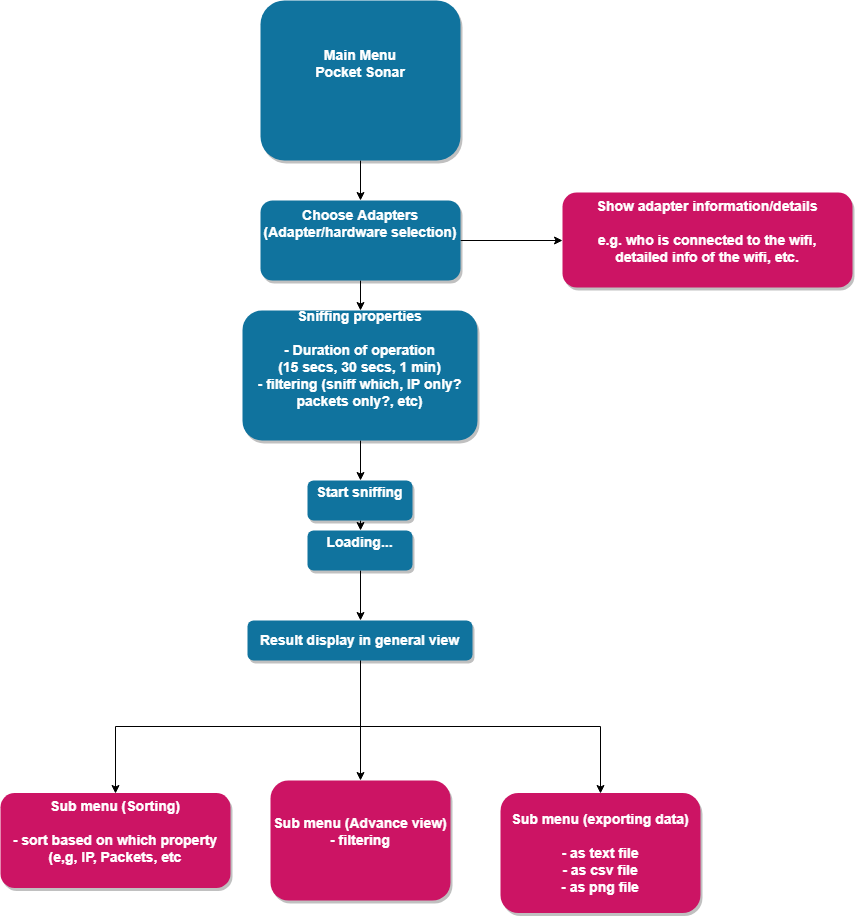


Figure 5.1.1. Program flow diagram

***5.2. Description of the functions***

* **Choose Adapters:** For this function, the users are able to select which network adaptor they will be using for the overall sniffing process. But if the mobile device is already connected to a network such as Wi-Fi or 4G, then the application will just automatically select the adaptor selection internally and move on to the sniffing process.
* **Show adaptor information:** There will be something like a sub-function which comes after choosing the adaptors. This sub-function will be able to show some basic information of the connected network
* **Scanning/Sniffing:** This is the very core function of our main program.In this part, there are several sub features as well. But the user will have to key in the IP address in which he/she wishes to scan or sniff. For the scanning part, this basically scan all the devices that are connected to the same IP address keyed in previously and display out some basic information of these respective devices. For sniffing, the procedure is the same as scanning where as the user will have to key in the required IP address and can simply press the start button to initiate the sniffing process. Afterwards, this will then display out all the IP packets and other related information.
* **Display:** This simple function is just to display out all the sniffed data on the UI so that the IT personnel can analyze and see if there is any anomalies. This will include some sub features such as display in a plain text form or in graphical form.
* **Exporting data:** This function is to export out all the sniffed data and save it either locally or in the cloud as text, csv or graphical files. The purpose of this function is for the users to refer back and compare with the new sniffed/scanned data.

***5.3. Refined Program overall flow diagram***

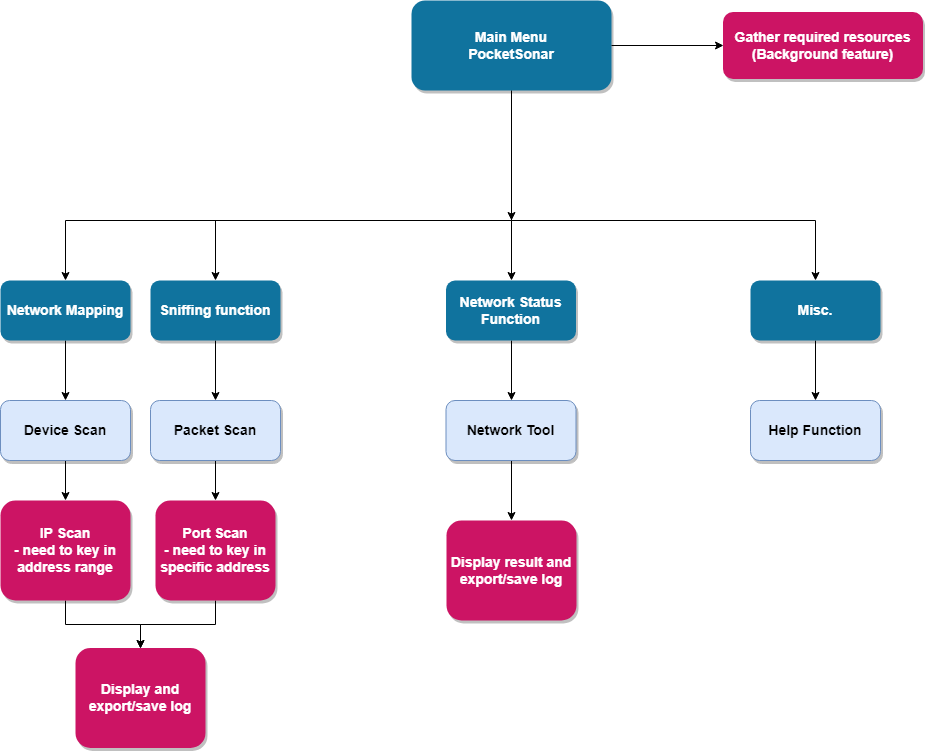


Figure 5.3.1. Refined Program flow diagram

***5.4. Revised Description of the functions***

* **Main Menu (Gather required resources):** At the beginning of the program, there will be a splash screen where the application will be gathering all the required resources for the functions to work. This will be working as a background feature because it is not necessary for the user to know what resources are being gathered. For our application, the user will only have to focus on analyzing the network.
* **Adapter/Network connection:** Before starting the network analysis, the user is to make sure that the mobile device is connected to the network which he/she wishes to scan. If the network is not connected, the relative scanning features will not function as well. For the network, it can be Wi-Fi, 4G, etc.
* **Function menu:** After the main menu, the user will be brought to the function menu where all the different scan features will be provided. He/she will then choose accordingly to start the network analysis.
* **Sniffing function:** Under this function, there are two main sub function, *Device Scan*and *Port Scan***.** Both features will need the user to key in the IP Address of the network connected. In Device Scan, the application will be focusing on scanning the IPs of the connected network while for Port Scan, it will be focused on checking the validity of the port. After all the scanning is done, the application will then display out the log and these logs will be automatically be saved internally in the mobile device storage.
* **Network Status Function:** This part of the function will provide some general network analysis features such as Network Tools, followed by displaying the respective results for the user to analyze.
* **Miscellaneous:** In addition to all the network analysis related functions, we have provided a miscellaneous menu where the user can find the *Help* function. Once the user goes into this function, the application will provide some brief descriptions on how each of the function is supposed to work, providing support contact details and application version.

1. **Requirement Analysis**

***6.1. Use Case Diagram***

In addition, we have come up with a use case diagram based on the overall flow diagram as mentioned above.

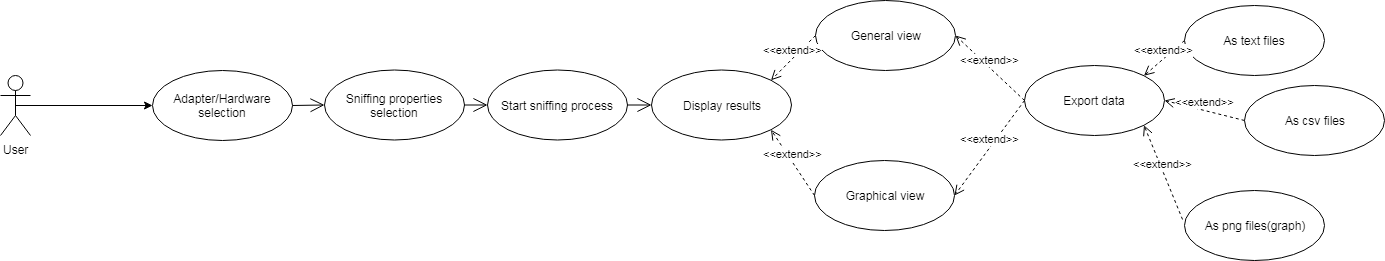


Figure 6.2.1. Use Case diagram

***6.2. Use Case Identification***

|  |  |
| --- | --- |
| **Use Case Textual Description** | |
| **UC - ID** | 1.1 |
| **Name** | Adapter/Hardware Selection |
| **Description** | This feature will allow the logged in user to select the adapters such as Wi-Fi, router, etc. |
| **Actor (s)** | User |
| **Main Scenario** | * User chooses a network before starting the sniffing process |

|  |  |
| --- | --- |
| **Use Case Textual Description** | |
| **UC - ID** | 1.2 |
| **Name** | Sniffing properties section |
| **Description** | This feature will allow the user to sniff packets or scan networks depending on the IP address |
| **Actor (s)** | User |
| **Main Scenario** | * User keyed in the desired IP address that he/she wishes to scan and analyse |

|  |  |
| --- | --- |
| **Use Case Textual Description** | |
| **UC - ID** | 1.3 |
| **Name** | Start sniffing process |
| **Description** | Once the adapter/hardware has been chosen, this function will start the process of collecting IP address and packets. |
| **Actor (s)** | User |
| **Main Scenario** | * Sniff the packets based on the connected network |

|  |  |
| --- | --- |
| **Use Case Textual Description** | |
| **UC - ID** | 1.4 |
| **Name** | Display results |
| **Description** | Once the sniffing process is done, this part will display out all the data such as IP address, source, packets, etc. |
| **Actor (s)** | User |
| **Main Scenario** | * User will be able to analyze the output data of packets after sniffing. |

|  |  |
| --- | --- |
| **Use Case Textual Description** | |
| **UC - ID** | 1.4.1 |
| **Name** | General view |
| **Description** | The Display results will be viewed in plain text form. |
| **Actor (s)** | User |
| **Main Scenario** | * User will be able to view the output in a plain text form |

|  |  |
| --- | --- |
| **Use Case Textual Description** | |
| **UC - ID** | 1.4.2 |
| **Name** | Graphical view |
| **Description** | The Display results will be viewed in graphical form. |
| **Actor (s)** | User |
| **Main Scenario** | * User will be able to view the output in a graphical form |

|  |  |
| --- | --- |
| **Use Case Textual Description** | |
| **UC - ID** | 1.5 |
| **Name** | Export data |
| **Description** | After all the above functions have been used, this part will provide the function to export out all the data and save it as a separate file. |
| **Actor (s)** | User |
| **Main Scenario** | * User will be able to use the saved files to compare with the newly sniffed data. |

|  |  |
| --- | --- |
| **Use Case Textual Description** | |
| **UC - ID** | 1.5.1 |
| **Name** | As text files |
| **Description** | The exported file will be saved as a .txt file. |
| **Actor (s)** | User |
| **Main Scenario** | * User will be able to save the output data as .txt file. |

|  |  |
| --- | --- |
| **Use Case Textual Description** | |
| **UC - ID** | 1.5.2 |
| **Name** | As csv files |
| **Description** | The exported file will be saved as a .csv file. |
| **Actor (s)** | User |
| **Main Scenario** | * User will be able to save the output data as .csv file. |

|  |  |
| --- | --- |
| **Use Case Textual Description** | |
| **UC - ID** | 1.5.3 |
| **Name** | As png files (graphs) |
| **Description** | The exported file will be saved as a .png file. |
| **Actor (s)** | User |
| **Main Scenario** | * User will be able to save the output data as .png file. |

***6.3. Refined Use Case Diagram***

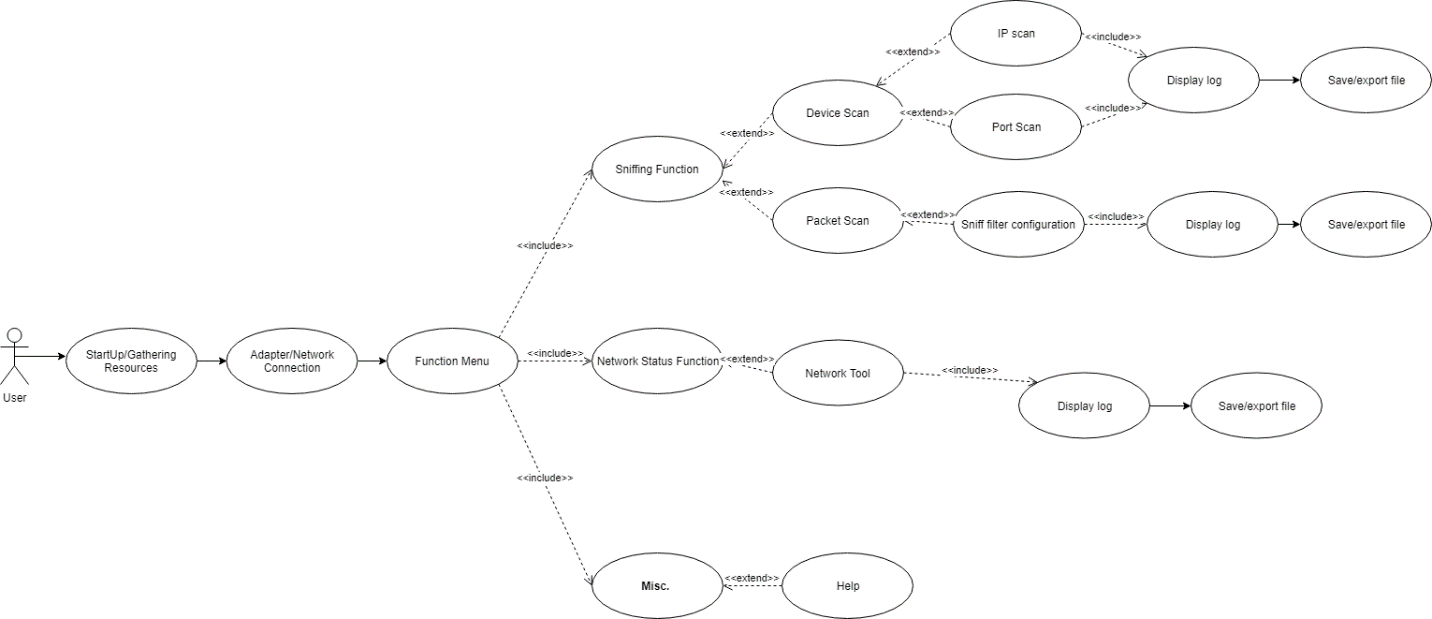
****

Figure 6.3.1. Refined Use Case diagram

***6.4. Refined Use Case Identification***

|  |  |
| --- | --- |
| **Use Case Textual Description** | |
| **UC - ID** | 1.1 |
| **Name** | Startup/Gathering resources |
| **Description** | During the start up of the program, it will automatically gather all the required resources for the application to function properly. |
| **Actor (s)** | User |
| **Main Scenario** | * This background feature will provide gathering resources but for this part, it is not necessary for the user to know what is happening unless the user is a developer, then he/she can look further into the code implementation. |

|  |  |
| --- | --- |
| **Use Case Textual Description** | |
| **UC - ID** | 1.2 |
| **Name** | Adapter/Network connection |
| **Description** | The mobile device needs to be connected to the network which needs to be analyzed. |
| **Actor (s)** | User |
| **Main Scenario** | * User is to ensure that the mobile device is connected to the network |

|  |  |
| --- | --- |
| **Use Case Textual Description** | |
| **UC - ID** | 1.3 |
| **Name** | Function Menu |
| **Description** | This part will provide additional sub menus for different type of network analysis functions. |
| **Actor (s)** | User |
| **Main Scenario** | * User can choose which function he/she wishes to use. |

|  |  |
| --- | --- |
| **Use Case Textual Description** | |
| **UC - ID** | 1.3.1 |
| **Name** | Sniffing function |
| **Description** | This is the core of the application where all the different types of network scanning will be provided. |
| **Actor (s)** | User |
| **Main Scenario** | * User can choose which network scanning method he/she wishes to use. |

|  |  |
| --- | --- |
| **Use Case Textual Description** | |
| **UC - ID** | 1.3.1.1 |
| **Name** | Device Scan |
| **Description** | Under this feature, there are two types of scan, IP and Port Scans. |
| **Actor (s)** | User |
| **Main Scenario** | * User needs to fill in the required IP address and start analyzing using the drop-down menu. |
| **Use Case Textual Description** | |
| **UC - ID** | 1.3.1.1.1 |
| **Name** | IP Scan |
| **Description** | This function focuses mainly on scanning of the IPs of subnet together with the class types. |
| **Actor (s)** | User |
| **Main Scenario** | * User need to key in the range of the IP address before starting the scan. |

|  |  |
| --- | --- |
| **Use Case Textual Description** | |
| **UC - ID** | 1.3.1.1.2 |
| **Name** | Port Scan |
| **Description** | This will focus more on checking the validity of the ports connected. |
| **Actor (s)** | User |
| **Main Scenario** | * User needs to key in a specific IP address before starting the scan. |

|  |  |
| --- | --- |
| **Use Case Textual Description** | |
| **UC - ID** | 1.3.1.1.3 |
| **Name** | Display logs and Save/Export file |
| **Description** | After scanning, the application will display out the respective log data and these logs will be saved or exported internally to a device storage. |
| **Actor (s)** | User |
| **Main Scenario** | * User can use these save logs for comparison or further analysis. |

|  |  |
| --- | --- |
| **Use Case Textual Description** | |
| **UC - ID** | 1.3.1.2 |
| **Name** | Packet Scan |
| **Description** | This part of the function will be focusing on the packet sniffing of the respected network connected. |
| **Actor (s)** | User |
| **Main Scenario** | * User can just simply start/stop the scanning provided the device is connected to the network. |

|  |  |
| --- | --- |
| **Use Case Textual Description** | |
| **UC - ID** | 1.3.1.2.1 |
| **Name** | Sniff filter configuration |
| **Description** | The application will allow a filtering function which can be set so that only specific packets will be scanned and displayed. |
| **Actor (s)** | User |
| **Main Scenario** | * User needs to set filter by keying in an IP or keyword |

|  |  |
| --- | --- |
| **Use Case Textual Description** | |
| **UC - ID** | 1.3.1.2.2 |
| **Name** | Display log and Save/Export log |
| **Description** | After scanning, the application will display out the respective log data and these logs will be saved or exported internally to a device storage. |
| **Actor (s)** | User |
| **Main Scenario** | * User can use these save logs for comparison or further analysis. |
| **Use Case Textual Description** | |
| **UC - ID** | 1.3.2 |
| **Name** | Network Status Function |
| **Description** | This sub function will provide some additional general network analysis features |
| **Actor (s)** | User |
| **Main Scenario** | * User can perform other types of network analysis functions |

|  |  |
| --- | --- |
| **Use Case Textual Description** | |
| **UC - ID** | 1.3.2.1 |
| **Name** | Network Tool |
| **Description** | The main feature of the Network Tool function is to detect network devices leading to the connection of the application and its targeted address so that the user will be able to analyze and take immediate action on the anomalies and preventing further damage. |
| **Actor (s)** | User |
| **Main Scenario** | * User can review and analyze data based on log display |

|  |  |
| --- | --- |
| **Use Case Textual Description** | |
| **UC - ID** | 1.3.2.4 |
| **Name** | Display and Save/Export file |
| **Description** | After scanning, the application will display out the respective log data and these logs will be saved or exported internally to a device storage. |
| **Actor (s)** | User |
| **Main Scenario** | * User can use these save logs for comparison or further analysis. |

|  |  |
| --- | --- |
| **Use Case Textual Description** | |
| **UC - ID** | 1.3.3 |
| **Name** | Miscellaneous |
| **Description** | Providing additional features. |
| **Actor (s)** | User |
| **Main Scenario** | * To provide general information for the application to the user. |

|  |  |
| --- | --- |
| **Use Case Textual Description** | |
| **UC - ID** | 1.3.3.1 |
| **Name** | Help Function |
| **Description** | General guide on how to use the application. |
| **Actor (s)** | User |
| **Main Scenario** | * To provide general information for the application to the user. |

1. **System Design**

***7.1. Overview***

Since our program will be in a form of an android application on a mobile device, the for the interface will be as follow.

Title

Menu Logo

function

function

function

Figure 7.1.1. Example of main page interface

Title

New page name

function

function

function

Figure 7.1.2. Example of sub page interface

1. **Implementation**

***8.1. Development methodology***

Some of the methods that we have used in tackling this project are as mentioned below.

* **Initial Planning:**
  + Understand and grasp the knowledge on what is related to the topic and start the necessary researches.
* **Planning and Requirements:**
  + Research on the multiple comparable products available on the market and gathering list of useful data/information which we can use in implementing our program.
  + Assigning of roles and responsibilities
  + Further research on the requirement of the implementation
  + Work based on the timeline and schedule as provided.
* **Analysis and Design:**
  + Identifying the necessary functionalities which will be related and useful for our overall program.
  + Coming up with flow diagrams, use case diagrams, etc. to aid the implementation process.
  + Further refining of the functionalities
* **Implementation:**
  + Starting of the code implementation based on the factors mentioned above
* **Testing:**
  + The whole team will conduct testing and analysis of the implemented program
  + Coming up with possible potential bugs and resolving such issues.
* **Evaluation:**
  + Examine the overall development of the project.

***8.2. Development Tools***

***8.2.1. Programming Language***

JAVA/JAVA IDE (Java JDK 8)

***8.3. Development Platforms***

***8.3.1. Android Studio***

Since our program needs to be implemented in an android platform, we have decided to use Android Studio as a development platform as it fits our requirements and at the same time, the program is a free open source and thus, leading to convenience.

As mentioned above, we are using JAVA as a base programming language. In addition, the development kit includes UI services as well and this is another requirement, we need in order to build the application.

***8.3.2. Software as a Product***

We have decided to develop this Android Network Sniffer as a Software as a Product instead of software as a service.

We will be developing an application for this program where all the users will be able to download from a webpage and directly install it on their mobile devices. Once installed, the user will just have to go through registration and once this has been done, they can start to analyze the network using the application. Some of the reason why we have decided to come up with SaaP is due to mobility, compatibility, convenience and user-friendliness.

1. **Risk Assessment**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Impact Type** | **Risk Seriousness (%)** | **Likelihood of Occurrence** | **WBS (affected work/task)** | **Risk Description** |
| **1** | Performance of program is not efficient and hangs | 65 | 0-4 | Affect the flow of the program | There will always be a possibility that the program might end up freezing or not responding, while the user will have to wait, and the program may or may not continue depending on the situation. |
| **2** | Core functions rely on java libraries | 40 | 0-2 | Unless libraries are implemented in Java, we are unable to implement the instructions | Some specific instructions or applications uses C++ or compiled binaries to do task that the library cant support, therefore expanding the potential of the application |
| **3** | Rooting | 50 | 0-2 | Unable to run main application to the mobile device | As mentioned, our program will need rooted android devices and only then the application can be installed and able to function. |

Table 12.1. Possible Risk Assessment

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Impact type/Risk type** | **Proposed management plan** | **(Possible) Percentage of Reduction in Risk** |
| **1** | Performance of program is not efficient and hangs | Programming multi-threaded implementation so that freezing will not occur on the main thread. Have a good programming practice by closing objects and locks that tend to eat up resources | -45 |
| **2** | Core functions rely on java libraries | When creating our Android application, we need to open up to use of compiled binaries so we can fully utilize the linux environment of Android | -70 |
| **3** | Rooting | For this part, if the user wants to root their own android devices, they need to find a rooted phone or the steps required to achieve rooting. On our end we have provided a guide to root Samsung Note 2 4G LTE N7105 | -100 |

Table 12.2. Risk prevention plan

1. **Limitation and Constraints**

As mentioned previously, our program can only work on certain android devices and this was implemented for security purpose as well. For the program to be installed and function properly, the android devices (preferably Samsung Galaxy Note 2) needs to be rooted and at the same time some third-party software needs to be installed. Only if these are done successfully, the program will be able to work properly.

Thus, the device must support Armeabi Architecture CPU so that the system can function more efficiently.

1. **Prototype**

***11.1. Overview***

Our application will be a Software as a Product as mentioned previously. Some of the interfaces for the prototype that we have come up with so far will be mentioned below.

***11.2. Layout***

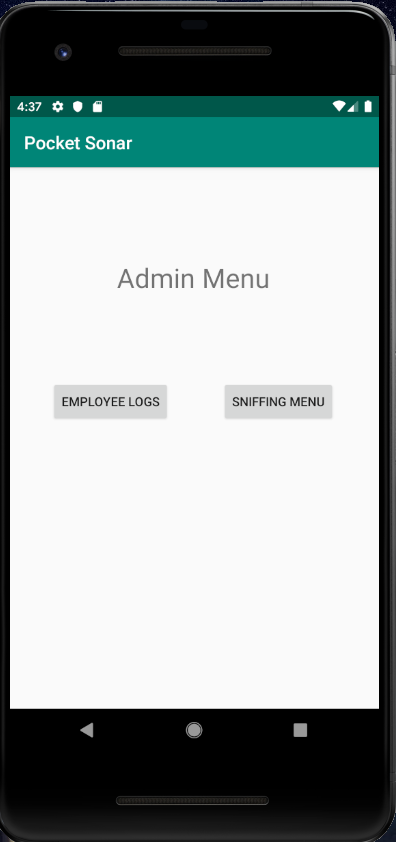


Figure 11.2.1. Main page

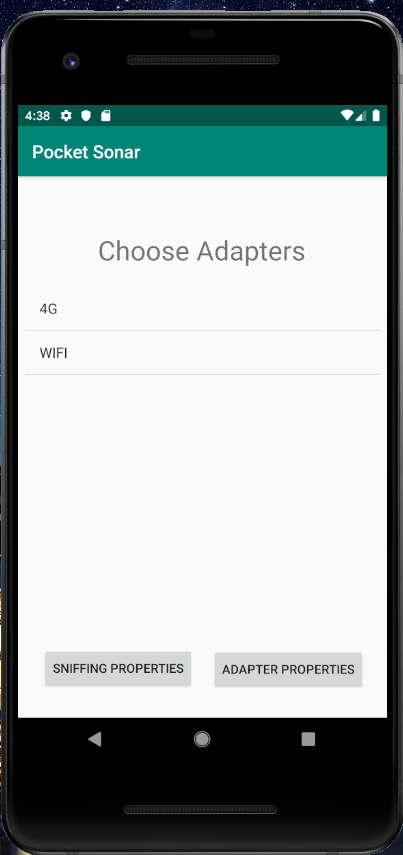


Figure 11.2.2. Adapter selection



Figure 11.2.3. Adapter information

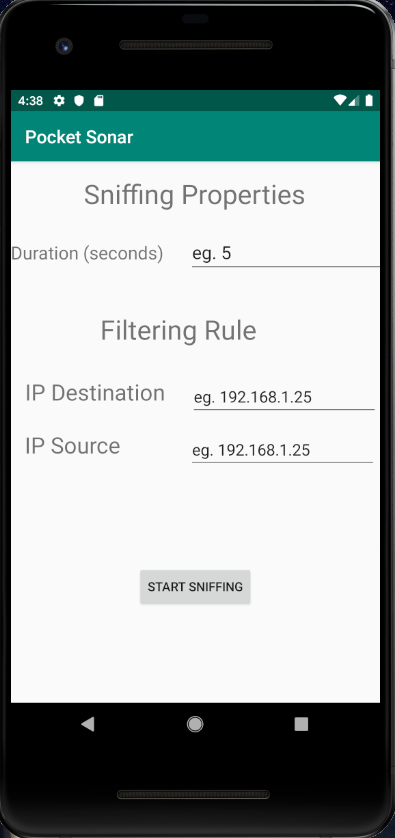


Figure 11.2.4. Sniffing properties selection

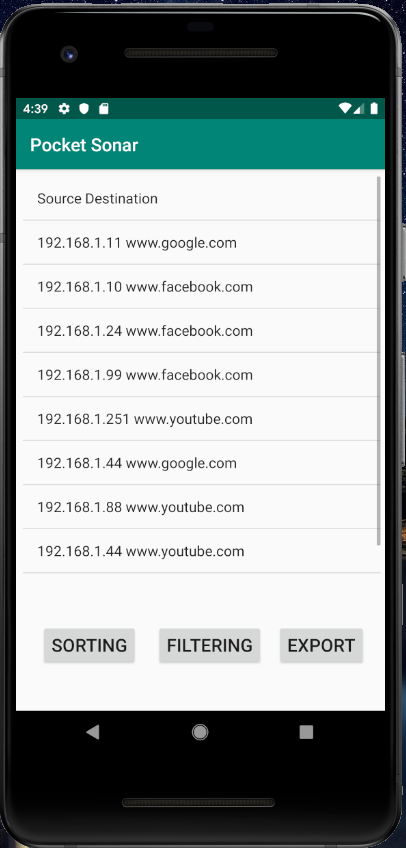


Figure 11.2.5. Sniffing results

1. **Final Product**

***12.1 Overview***

Some of the figures shown below are the general overview of the user interface and functions. The main functions and how they work will be provided in the user manual.



Figure 12.1.1. ProductOverview\_1

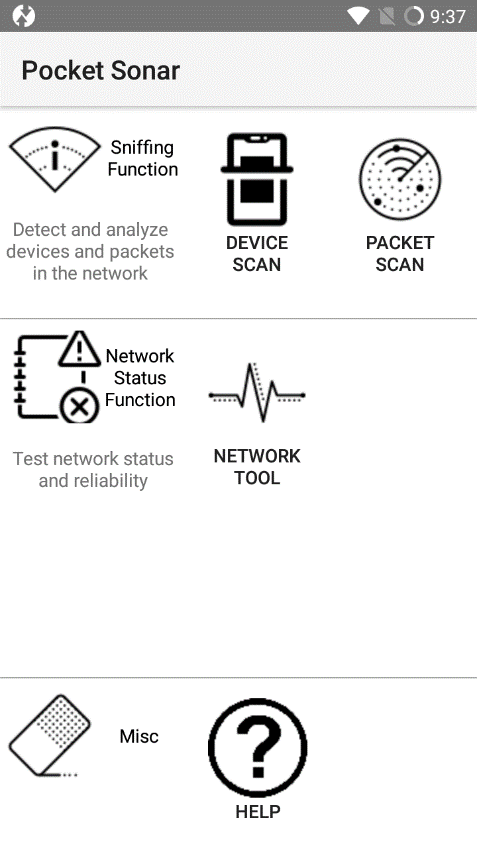


Figure 12.1.2. ProductOverview\_2



Figure 12.1.2. ProductOverview\_3

***12.2. Functionalities***

As mentioned, our program will be best compatible with a rooted android device (Samsung Galaxy Note 24G, N7100) and this is part of the security feature as mentioned in Section 4.1.

***12.2.1. Device Scan***

This function is the core function of our program. We are currently using NMAP 7.1.3 compiled binary for ARMEABI device. Our application will communicate with a shell that runs the binary and format the output in the application when required. To elaborate more, there are two modes under this function, one to scan all the IPs in the subnet or one single IP and the other one is to scan the port for the connected device.

For the first mode the user is required to key in the IP of the subnet and by this, it also involves the Class types (e.g. Class A, B or C). Our program is implemented in a way that it will be able to scan the ranges for all the classes. Once scanned, the program will display out the log and at the same time these logs will be automatically saved under the device storage which can be accessed at a later time or in the future for comparison purposes. The format of the file saved will be the date followed by prefix ZX and IP XZ and also with the file type mapf. Eg. 2018-07-15 23:12:23 ZX192.168.2.3XZ.mapf.

For the second mode, which is the Port Scan, the user will have the key in the specific IP address and only then should continue to scan. Once the scanning is done and logs are displayed, the user will be able to click on these logs and this will open up a webpage which shows all relevant TCP services associated with the port. The user can uses these to analyze the validity of the port which is also implemented as part of a security feature. The format of the file saved will be the date followed by prefix ZX and IP XZ and also with the file type portf. Eg. 2018-07-15 23:12:23 ZX192.168.2.3XZ.portf

For both of the scanning processes, we have used NMAP which is why rooting is required. In addition, we are implementing using binary, in which the execution of the binary file requires elevated functions. These functions will only work if the device is rooted as we need elevated permissions to directly access the wireless interface. Also, these binary files are used to perform the scanning where the user can just open the command prompt or shell inside the android devices. It will allow us to key in the binary as well as some commands for troubleshooting purposes. The reason why we used binary method is because in our opinion, it can make the application smaller in size and thus, will not require much storage space when being installed.

Reference: <https://nmap.org/>

Reference: <https://github.com/kost/nmap-android/releases>.

***12.2.2. Packet Scan***

In this part of the program, we will be providing some filtering feature to provide ease-of-access for the user. We will be providing filtering configuration as well whereas the user will have to type a certain keyword and during the filtering process, it will only display out the results containing the input keyword. The keyword doesn’t have to be just simple words such as Google, Facebook, Instagram, etc., but also can be IP addresses, and the program will only sniff out the packets that are related to what was filtered. If no filter was set, the program will basically sniff out all available packets.

For this part, the user will have to set a default DNS of the mobile device and only then will be able to can. In addition, since we are using TCP dump to perform the sniffing process, it actually requires a certain number of packets to come through before the scanning starts. Once it has been scanned, it will forward the requests back to the default DNS.

We are implementing part of TCP dump in our program. The application supports the Promiscuous mode, which only requires the connection to the AP to perform scanning.

Reference: <https://www.tcpdump.org/>

Reference: <https://www.androidtcpdump.com/>

***12.2.3. Network Status Functions***

This part of the program will provide some general network analysis functions such as Ping test, and Trace route.

*PING* – To test the ping of the website, IP, etc. If the ping is reachable, then the user will be able to review and analyze the network. Ping uses the inbuilt binary of Android to do so

*Trace Route* – To provide information and display the entire route between the user and the targeted IP address. Trace Route uses NMAP to perform the trace and output it back to the user

***12.2.4. Help function***

This function will provide information and guidance on how to use the application and at the same time providing other information such as what implementations are being used, version of the program, so on and so forth.

1. **Conclusion**

After going through all the research, we have found out that there is a variety of network sniffers. Some of them have similar functions but some software provides better function and a more detailed information regarding the packets. In addition, different software is based on different OS/platform. Some of them are based on windows, some in IOS and last but not least, in androids. We have also learned that network sniffers are tools which intercept logs and analyses the traffic. For instance, as data streams flow across the network, the sniffer captures each packet and, if needed, decodes the packet's raw data, showing the values of various fields in the packet, and analyses its content according to the appropriate RFC or other specifications.

From a search through the different types of network sniffers available in the market currently, the network sniffers used in android platform and Windows platform are mainly for high level usage, where the purpose of the application is to provide IT professionals real-time monitoring of the network. However, these applications usually provide detailed information only, it does not provide a more interactive view like diagrams and chart presentations.

Through our research, our goal is to create an application, aimed at developing a powerful android network sniffer that allows IT professionals to empower their android devices. However, we will add in more features like including interactive presentations in a text format, as well as making the application more user friendly. The application should provide detailed information on the network, which IT professionals can make use of the information acquired to assist them in their daily work, or to help them work more efficiently.

1. **Annexes**

***14.1. Meeting minutes***

* **Meeting minutes 1**

Venue: SIM

Date: 13/4/2019

Duration: 2 hours

*Members present:*

1. Nyein Soe

2. Wilson

3. Alvin

*Topics Discussed:*

1. Nomination of team leader

2. Splitting up the tasks for research

* **Meeting minutes 2**

Venue: SIM

Date: 27/4/2019

Duration: 3 hours

*Members present:*

1. Nyein Soe

2. Wilson

3. Alvin

*Topics Discussed:*

1. Download & test apps

2. Excel updates

3. Template plan - Pending

4. Update plan

5. Tidy docs

6. Update blog

* **Meeting minutes 3**

Venue: Starbucks, Cuppage Road

Date: 17/05/2019

Duration: 5 hours

*Members present:*

1. Nyein Soe

2. Wilson

3. Alvin

*Topics Discussed:*

1. Compiled research & findings

2. Discussed on problems encountered

3. Updated project plan

4. Update blog

5. Discussed on software implementation plan

* **Meeting minutes 4**

Venue: SIM

Date: 22/5/2019

Duration: 3 hours

*Members present:*

1. Nyein Soe

2. Wilson

3. Alvin

*Topics Discussed:*

1. Reviewed objective and purpose of program

2. Drafted necessary diagrams

3. Drafted user rights and access control list

* **Meeting minutes 5**

Venue: SIM

Date: 19/6/2019

Duration: 2 hours

*Members present:*

1. Nyein Soe

2. Wilson

3. Alvin

*Topics Discussed:*

1. Finalized diagrams

2. Finalized user rights and access control list

3. Implementation of prototype

* **Meeting minutes 6**

Venue: SIM

Date: 27/6/2019

Duration: 2 hours

Members present:

1. Nyein Soe

2. Wilson

3. Alvin

Topics discussed:

1. Finalized documents (Preliminary Technical Documents, Preliminary User Manual)

2. PowerPoint slides for End Term Review

3. Testing of prototype

* **Meeting minutes 7**

Venue: Starbucks, Suntec City

Date: 8/7/2019

Duration: 2.5 hours

Members present:

1. Nyein Soe

2. Wilson

3. Alvin

Topics discussed:

1. Researched on network sniffing codes and functions

2. Testing and implementation of researched codes and functions

* **Meeting minutes 8**

Venue: SIM

Date: 17/7/2019

Duration: 3 hours

Members present:

1. Nyein Soe

2. Wilson

3. Alvin

Topics discussed:

1. Testing and implementation of researched codes and functions

2. Troubleshooting and reviewing of codes and functions

* **Meeting minutes 9**

Venue: SIM

Date: 1/8/2019

Duration: 2 hours

Members present:

1. Nyein Soe

2. Wilson

3. Alvin

Topics discussed:

1. Testing and implementation of researched codes and functions

2. Troubleshooting and reviewing of codes and functions

3. Rooting of Android based smartphones

* **Meeting minutes 10**

Venue: National Library

Date: 9/8/2019

Duration: 4 hours

Members present:

1. Nyein Soe

2. Wilson

3. Alvin

Topics discussed:

1. Review and testing of program functions

2. Update existing documents, finalize test cases

* **Meeting minutes 11**

Venue: SIM

Date: 20/8/2019

Duration: 2.5 hours

Members present:

1. Nyein Soe

2. Wilson

3. Alvin

Topics discussed:

1. Review and update of all documents

2. Preparation of hard copy of documents

3. Preparation of all essential items

* **Meeting minutes 12**

Venue: SIM

Date: 24/8/2019

Duration: 4 hours

Members present:

1. Nyein Soe

2. Wilson

3. Alvin

Topics discussed:

1. Review and update of all documents

2. Printing of hard copy of documents

3. Review website and database features

* **Meeting minutes 13**

Venue: SIM

Date: 28/8/2019

Duration: 3 hours

Members present:

1. Nyein Soe

2. Wilson

3. Alvin

Topics discussed:

1. Improve performance and UI of program, and test with test cases again

2. Printing of hard copy of documents

3. Preparation of mpeg video and burning of files to disc

***14.2. Project Diary***

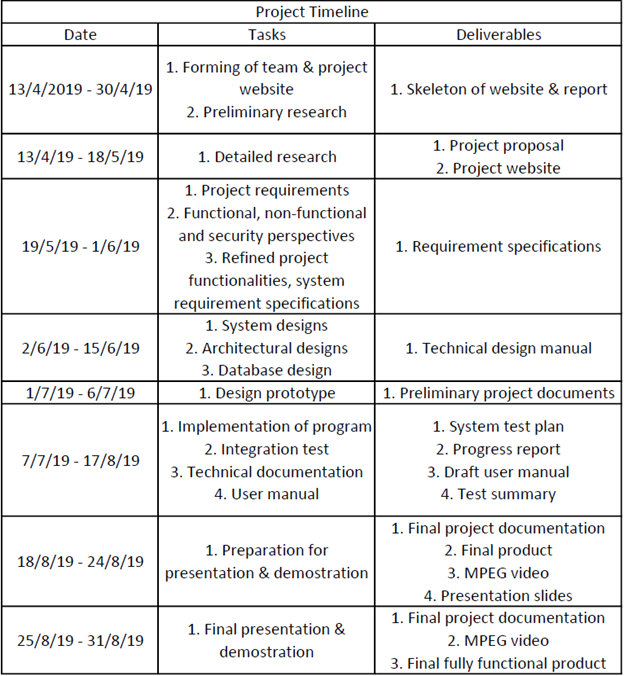
****

Table 14.2.1. Project Diary/Timeline

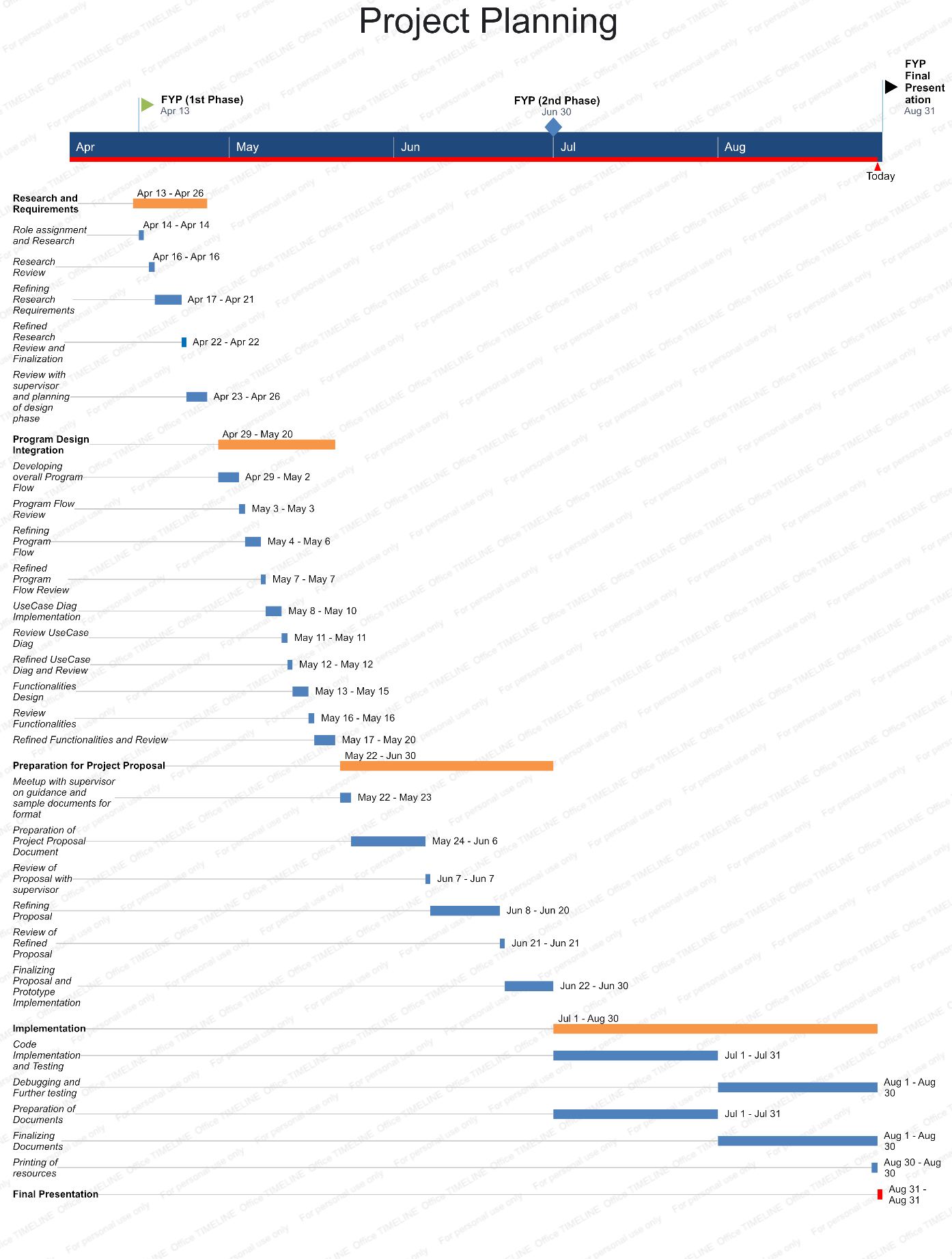
***14.3. Project Planning Gantt Chart*** 

Figure 14.3.1. Project Planning Gantt Chart

***14.4. Website and Blog***

Our website link is <https://android-network-sniffer.webnode.com/>

***14.5. Test Cases***

Link to Test Cases: <https://docs.google.com/spreadsheets/d/1USfWywW8qju7HZoPm8yHKX2S4_zoFR-hohe7s4rGdcI/edit?usp=sharing>