**Software Requirements**

**Specification**

**for**

**<Android Network Sniffer>**

**Version 1.0**

**Prepared by**

**< Soh Yu Xuan, Timothy Chin, Kenneth Huang, Kendrick Tan >**

**<22 June 2018>**

# Table of Contents

[Table of Contents 2](#_Toc522286831)

[Revision History 2](#_Toc522286832)

[1. Introduction 1](#_Toc522286833)

[1.1 Purpose 1](#_Toc522286834)

[1.2 Document Conventions 1](#_Toc522286835)

[1.3 Intended Audience and Reading Suggestions 1](#_Toc522286836)

[1.4 Project Scope 1](#_Toc522286837)

[1.5 References 1](#_Toc522286838)

[2. Overall Description 2](#_Toc522286839)

[2.1 Product Perspective 2](#_Toc522286840)

[2.2 Product Features 2](#_Toc522286841)

[2.3 User Classes and Characteristics 2](#_Toc522286842)

[2.4 Operating Environment 2](#_Toc522286843)

[2.5 Design and Implementation Constraints 3](#_Toc522286844)

[2.6 User Documentation 3](#_Toc522286845)

[2.7 Assumptions and Dependencies 3](#_Toc522286846)

[3. System Features 4](#_Toc522286847)

[3.1 Network Sniffing Feature 4](#_Toc522286848)

[3.2 System Feature 2 (and so on) 4](#_Toc522286849)

[4. External Interface Requirements 5](#_Toc522286850)

[4.1 User Interfaces 5](#_Toc522286851)

[4.1.1 User start sniffing PCAP 7](#_Toc522286852)

[4.1.2Saving the Sniffed data into a file 8](#_Toc522286853)

[4.2 Hardware Interfaces 11](#_Toc522286854)

[4.3 Software Interfaces 11](#_Toc522286855)

[5. Other Nonfunctional Requirements 11](#_Toc522286856)

[5.1 Performance Requirements 11](#_Toc522286857)

[5.2 Safety Requirements 11](#_Toc522286858)

[5.3 Security Requirements 12](#_Toc522286859)

[5.4 Software Quality Attributes 12](#_Toc522286860)

[6. Other Requirements 12](#_Toc522286861)

[Appendix A: Glossary 12](#_Toc522286862)

[Appendix B: Analysis Models 12](#_Toc522286863)

[Appendix C: Issues List 12](#_Toc522286864)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
| Kendrick Tan | 02/07 | Correction of certain description and explanation | 1.1 |
|  |  |  |  |

# 1. Introduction

## 1.1 Purpose

This document describes the requirement specification for the Android Network Sniffer application “NetSniffer” that captures packets through the use of existing packet capturing technologies. It presents a means of viewing packets that are currently in a network and possibly provide some information on the packets. It will explain the Features of the system, Interface, what the system can do, what are the possible constraints and limitation when using the Network Sniffer, possible additional features of the Network Sniffer compared to currently known Android Network Sniffer. The document is intended for developers and users of the system.

## 1.2 Document Conventions

This document follows the basic SRS convention methodology and also based on templates that are online in which to draft this document

## 1.3 Intended Audience and Reading Suggestions

This document is intended for developers, project managers, users, testers and documentation writers. The SRS below contains information regarding the project, scope of the project, references used in writing the SRS, testers who if possible solve any issues that the current developers may have faced, and also for users who wish to understand what the project created was about. It is suggested when reading to have knowledge with regards to Networking, Android, Android Programming, Java, C Programming, as they will provide an easier understanding.

## 1.4 Project Scope

The android network sniffer application where the application should be able to capture packets and view them. However, there may be certain limitations and constraints that may be imposed on due to the requirements which will be further explained in the later sections

Objectives are to have the ability to perform real time capturing and viewing of packet data and save the data into a file which can be viewed later, additional features such as filtering of traffic, viewing

## 1.5 References

<https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database>

<https://aakashtechsupportdocs.readthedocs.io/en/latest/prodpersp.html>

# 2. Overall Description

## 2.1 Product Perspective

The application is solitary consisting single mobile application. The mobile application will be used to capture and view packets, also to save packet information in text files on the mobile device. The mobile application will need to be rooted for this application to work. By default network card for mobile phone manufacturers disable the ability to change the mode of the network card. Rooting the phone allows the ability to change the mode of the network card that allows the capturing of packets. This is a network/data centric application that requires some storage space. It also utilizes text files and storage locations on the mobile device. Storage locations such as "/sdcard/Download/".

The mobile application also has some restrictions such that it has to be connected to a wifi access point to capture packets.

## 2.2 Product Features

The main features of our product includes:

* Network Sniffing: To be able to start & stop network sniffing
* To be able to provide real time viewing of sniffed data
* To be able to save the sniffed data into a file
* To be able to read from the saved file
* To be able to filter packet type captured

## 2.3 User Classes and Characteristics

The type of User classes for our product would be mostly IT users, developers.

IT Users: users that work in the field of IT such as System Administrators, who constantly have work involving Network such as Monitoring of Network Traffic for anomaly, or even regular data collection of network traffic to improve the system such as avoiding overhead. IT users would be the type of users that most frequently use Android Network Sniffers

Developers: Users that wish to develop a similar type of Application may want to user our product as reference or possibly create a more improved version of ours as the technology improves

## 2.4 Operating Environment

The device of our product choice must support Android Studio, programming language will mostly be done in Java, with the exceptions of binaries done in C using the Pcap library.

Hardware of the device must have a chipset that allows NIC promiscuous mode. Device also needs to be rooted.

## 2.5 Design and Implementation Constraints

Because of the platform in which our product is on which is Android there will be limitation and constraints compared to the Personal Computer type of Network Sniffer.

When designing our product, the constraints are as follows:

* Network Cards/Wifi Chips for mobile devices are manufactured in such a way that they are not allowed to change their mode.(Why root is needed)
* Network Interface Cards not being able to change their mode means that the alteration of the underlying firmware for the cards is needed. (Firmware allows to change card mode)
* Not all wifi chips support promiscuous/monitor mode, resulting in some phones unable to capture packets. (Only certain wifi chips support packet capture)
* The device must be rooted which is to give Super User Access to the phone. However, it is to be noted that rooting a phone would be similar to what would be known as Jail Breaking an iPhone.
* Phone being rooted and requiring the network card means only 1 person test app at a time. (Inconvenience, cannot use emulator to fully test app)
* Device being rooted results in super user privileges given to applications that should not have the permissions. This requires something to manage the SU privileges.
* The inability to give the app SU results in the need to spawn processes that have SU permission.
* The inability to give the sniff app SU also means that code cannot be used in to sniff. (must use process -> binary)
* Limited resources on mobile devices.
* Have to be coded in Android environment, Java or C.

## 2.6 User Documentation

Other than the product of an Android Network Sniffer there will also be a Technical Documentation that elaborate clearly the entire technical aspect of the product

Other than the Technical Document there will also be a User Documentation to allow ease of use for the users who will be using our application

## 2.7 Assumptions and Dependencies

Assumptions -> users are using devices that are rooted, Android Marshmellow 5.0 and above, wi-fi card must be able to support promiscuous mode.

Dependencies -> C compile Binaries

# 3. System Features

Below list our system features of the product and all possible features available

## 3.1 Network Sniffing Feature

Users should be able to start and stop the network sniffing of the application. Be able to see, capture and save packet data.

3.1.1 Description and Priority

This feature is considered the core, High Priority, of our entire product where by the user is able to start and stop sniffing of network. The user should be able to capture packet data and this data should be able to be saved onto storage in the device.

3.1.2 Stimulus/Response Sequences

User starts the network sniffing and the device will display packet data onto the screen. The network sniffing can be stopped at any time.

Upon starting and subsequently pressing the stop button, stopping the network sniffing, the device will save packet data to a file in the ‘/sdcard/downloads’ folder. Once the packet data is saved, it can be viewed and further analysed on the computer using other 3rd party programs like Wireshark.

3.1.3 Functional Requirements

Start and Stop sniffing of packets. Able to save the sniffing data to a file which can be retrieved later for viewing/analysis.

## 3.2 Network Sniffing Filter Feature

Filter data captured from sniffing a network.

3.2.1 Description and Priority

Be able to filter the data from the sniffing feature, Medium Priority. Can be filtered by anything. Can be used in conjunction with the network map feature of obtaining Mac Address of a device. Filter packet data captured by the Mac Address.

3.2.2 Stimulus/Response Sequences

Filter can be set at any time. User enters the filter. The data displayed on screen is related to the filter.

3.2.3 Functional Requirements

Able to filter data captured from sniffing.

## 3.3 Network Sniffing Analysis Graph Feature

Creates and displays a graph dependent on the data captured/saved based on a .pcap file.

3.3.1 Description and Priority

Reads a .pcap file and creates a graph based on the file, Medium Priority. The file can be from the app itself or from elsewhere. Being able to create a graph showing the number of packets related to a device. The Y-axis being the number of packets and the X-axis being the device. The graph is created and displayed to the screen.

3.3.2 Stimulus/Response Sequences

The user first chooses the .pcap file to read. Based on the .pcap file, the graph will be generated onto the screen. The graph shows the number of packets in relation to a specific device.

3.3.3 Functional Requirements

Create a simple analysis related to the network sniffing data captured.

## 3.4 Network Mapping Feature

Users should be able to start the mapping of the network. Be able to see devices connected to a specific network.

3.4.1 Description and Priority

Being able to map the network, Medium Priority, showing every device that is connected to a specific network/sub-network.

3.4.2 Stimulus/Response Sequences

User enters a specific network address to map to. User then press the start network map button. The output pertains to devices connected to the network. The device data should be displayed to the screen.

3.4.3 Functional Requirements

Be able to see devices connected to a specific network. Also to check data collected against a list of authorized mac addresses.

## 3.5 Network Mapping Check Mac Feature

Allows checking of mac addresses found against a list of authorized mac addresses.

3.5.1 Description and Priority

After mapping the network, the user can check against a list of authorized mac addresses to see if there are any devices missing or unauthorized devices connected, Medium Priority.

3.5.2 Stimulus/Response Sequences

After mapping the network, all the details are displayed to the screen. Scrolling all the way to the bottom of the screen, there is a ‘Nmap Done’. Upon pressing of ‘Done’, the user will be prompted to choose the file to check against mac addresses found. A short report will be generated.

3.5.3 Functional Requirements

To check the validity of mac addresses found

## 3.6 Network Mapping OS Feature

Mapping the OS of a device in the network and display possible vulnerabilities.

3.6.1 Description and Priority

User should be able to enter an address related to a device in the network and find out what OS the device is running, Medium Priority. Also able to see vulnerabilities against that OS.

3.6.2 Stimulus/Response Sequences

User should be able to enter an address related to a device in the network. The type of OS the device is running should be displayed onto the screen. Upon pressing on the output displayed, a web browser with a list of possible vulnerabilities against the OS will be displayed.

3.6.3 Functional Requirements

Find out what OS the device is running. Also be able to see OS vulnerabilities for the device.

# 4. External Interface Requirements

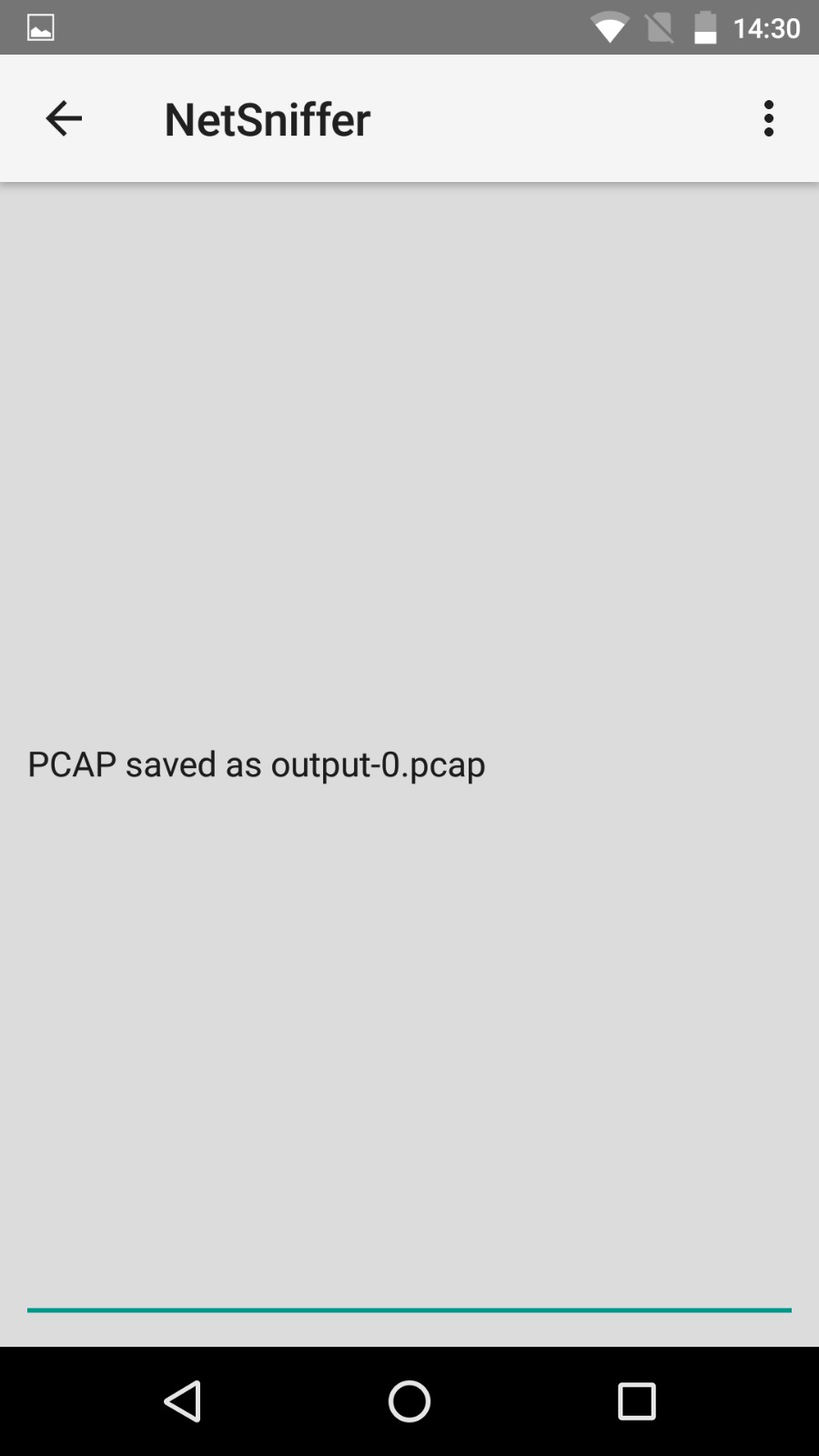
## 4.1 User Interfaces



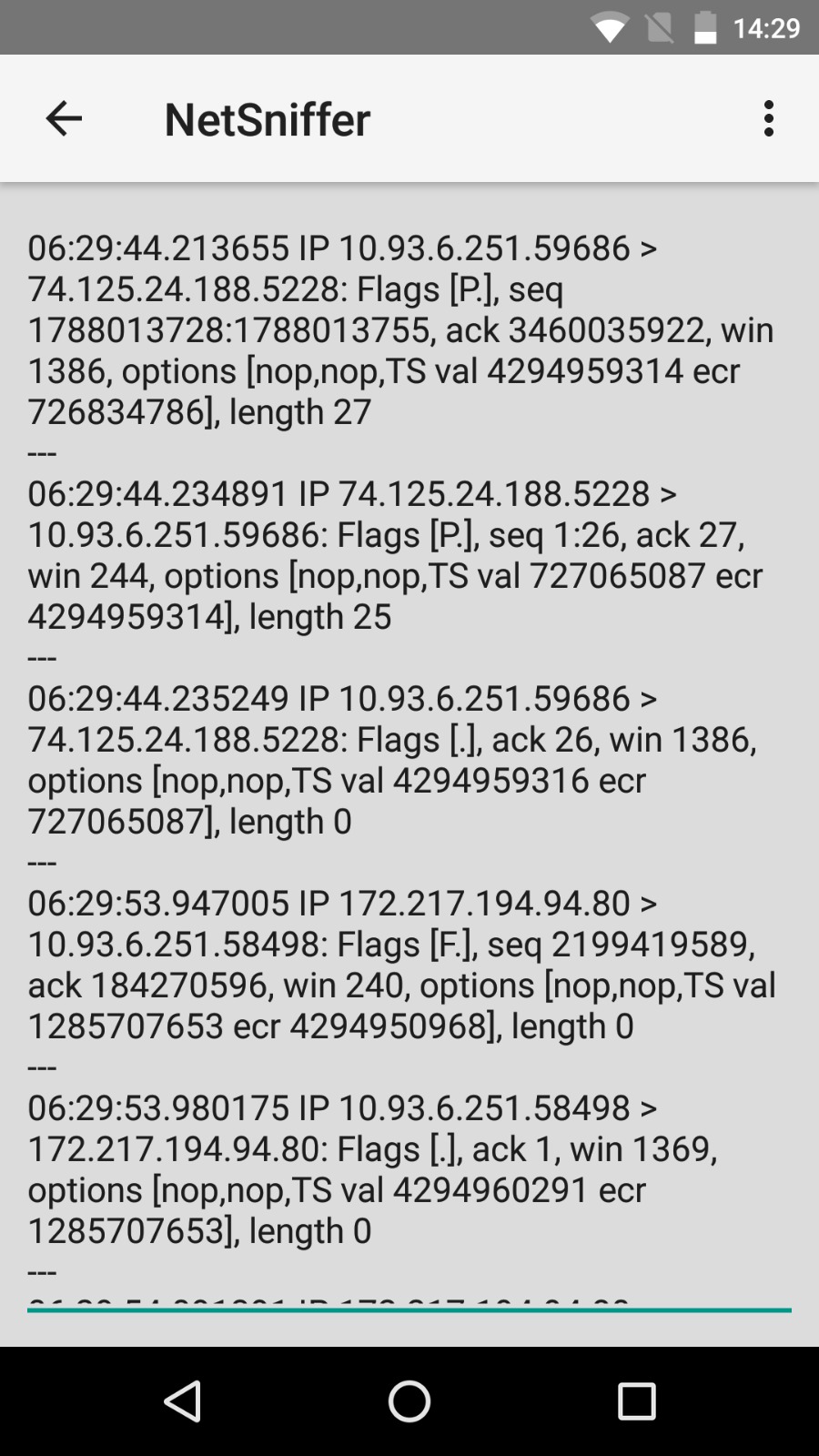
Main Screen of the Application



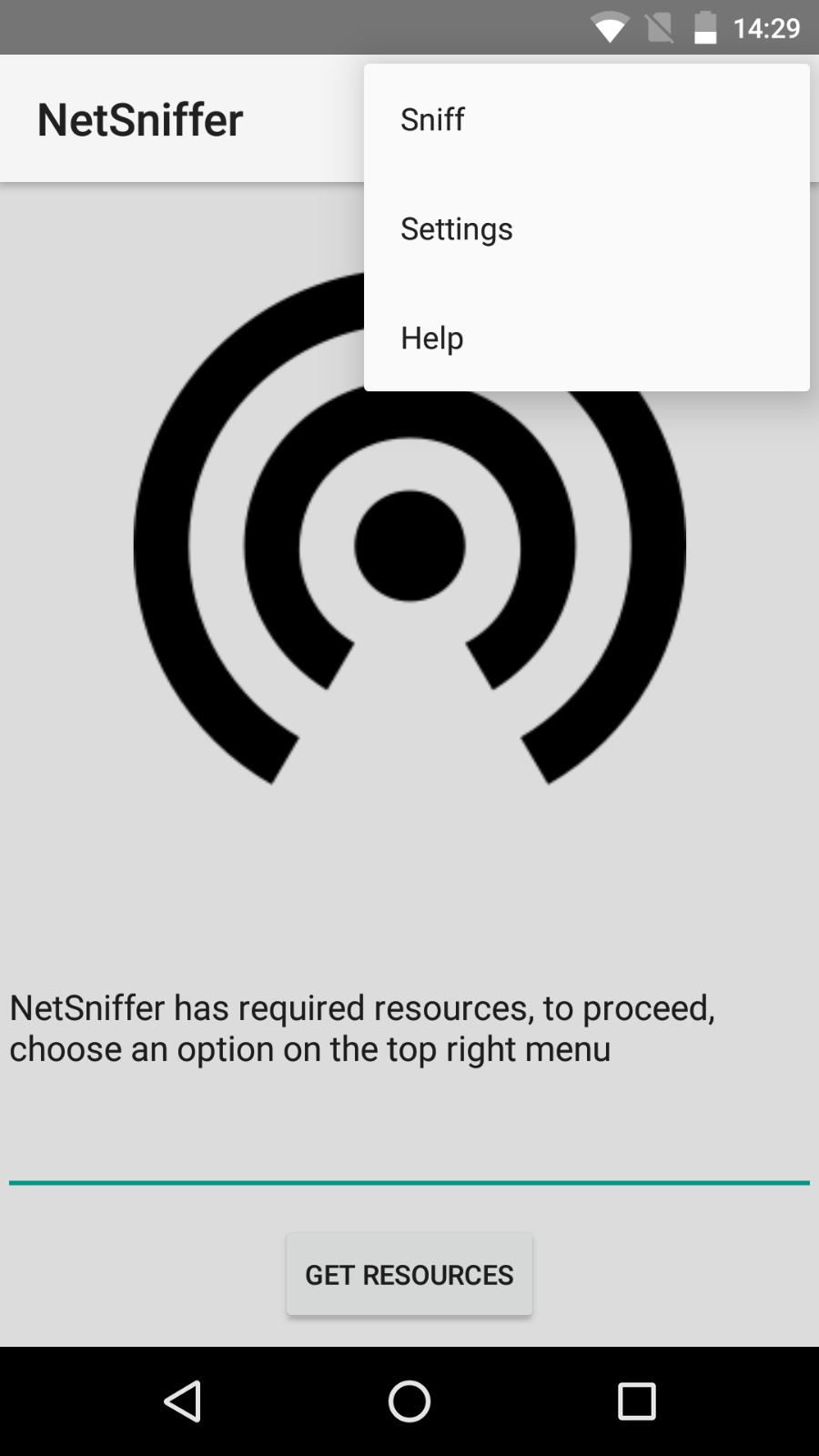
## 4.1.1 User start sniffing PCAP



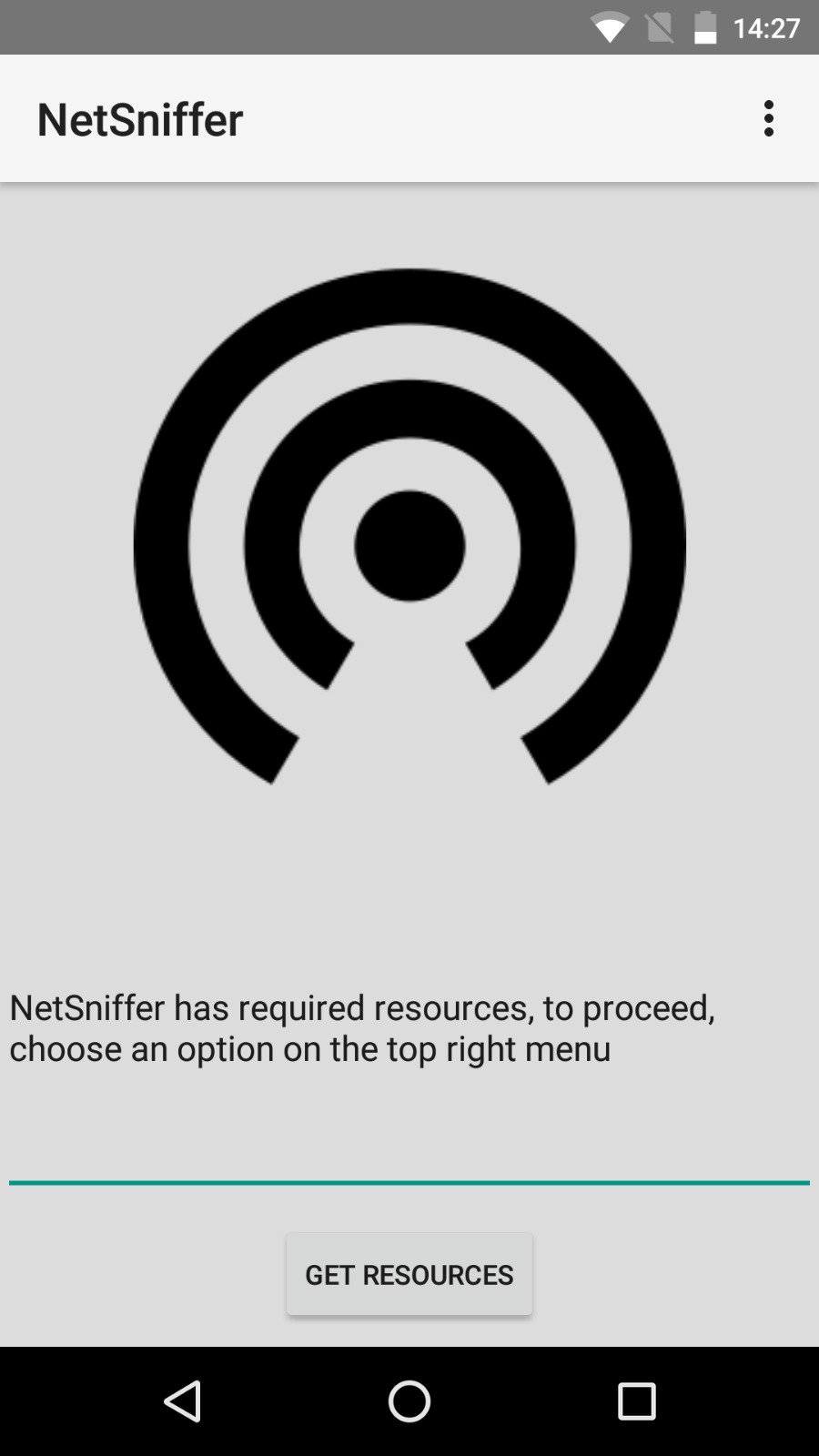
## 4.1.2Saving the Sniffed data into a file



Prototype version Real time display of sniffing, with delay of 3s and updates after every 1s



Menu of our Application where user can Select to sniff



Application Main Page which is displayed to the user when they launch the application

## 4.2 Hardware Interfaces

Devices used must have chipset that allows Monitor mode, device also needs to have root access. After Sniffing of the network the file will be saved in a Pcap format that can only be run by applications that can read this particular format.

Example of chipset BCM 4325,4329,4330, 4335,4339.

Nexus 5 device for our project uses BCM 4339.

## 4.3 Software Interfaces

Prototype version -> TCP Binary for ARM devices

* Stored in assets
* Extracted to Internal Storage (app)
  + /data/data/com.example.yuxuan.netsniffer/tcpdump
* Main Activity checks if exists
* Sniff Activity
  + Creates processes (threads)
  + Queries Binary and Output to a text file
  + Updates UI
* Nmap Activity
  + Creates processes
  + Binary and Output to text file
  + Update UI

# 5. Other Nonfunctional Requirements

## 5.1 Performance Requirements

Wi-Fi card Range that is device specific as different devices uses different Wi-Fi cards and even if the Wi-Fi cards are compatible they may have different ranges to do the sniffing

## 5.2 Safety Requirements

Rooting the phone is one of the requirements of the device however it is to be noted that when rooting the phone there will be some effects that the user must be aware of

Disadvantages when Rooting phone

* Rooting immediately voids your phone’s warranty
* Rooting have a risk of “bricking” the phone: Device might become dead and unusable
* Poor performance: when additional features that need to be added might cause the device to lose performance speed and features
* Viruses: custom programs used might make changes to software codes which might have a chance of introducing virus

## 5.3 Security Requirements

There is a security issue when using the phone in which if the device is rooted. There will be an increase in risk of viruses because rooted device allows for special permissions that allow additional customization of the device. For example, places that were not allowed to be modified can now be accessed. This causes vulnerabilities as things that shouldn’t be changed are now open.

There is now a need to manage the Super User permission on the device. “SuperSu” android application in this case was used to manage permissions.

## 5.4 Software Quality Attributes

# 6. Other Requirements

# Appendix A: Glossary

IT: Information Technology

Pcap: a type of file extension similar to .txt .html

# Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

# Appendix C: Issues List

1. Output from runOnUiThread should be formatted in some way for user friendliness
   1. Working on adapter with ListView(dynamic) in help activity