

**CSCI321 – Project (Android Packet Sniffer)**

Report

**Group SS18/2A Android Network Sniffer Report**

|  |  |  |  |
| --- | --- | --- | --- |
| **Team Number : <SS18/2A>** | | | |
|  | **Student Number** | **Name** | **Email Address** |
| 1 | 5498636 | Soh Yu Xuan(Team Leader) | yxsoh004@mymail.sim.edu.sg |
| 2 | 5498399 | Chin Tian Zhi Timothy | tztchin001@mymail.sim.edu.sg |
| 3 | 5498442 | Huang Shun Yang Kenneth | sykhuang001@mymail.sim.edu.sg |
| 4 | 5026556 | Tan Siyuan Kendrick | sktan024@mymail.sim.edu.sg |

Table of Contents

1. [1. Project Overview 1](#_Toc523136568)

[A) Vision](#_Toc523136569)

[B) Initial Use-Case Model](#_Toc523136570)

[C) Project Plan](#_Toc523136571)

1. [2. Roles and Responsibilities 2](#_Toc523136572)
2. [3. Risk Analysis and Counter Measures 3](#_Toc523136573)
3. [4. UML Diagrams 4](#_Toc523136574)

[Design Artefacts - Use Cases (Iteration 1)](#_Toc523136575)

[Design Artefacts – Activity Workflows (Iteration 1)](#_Toc523136576)

[Design Artefacts – Activity Workflows (Iteration 2)](#_Toc523136577)

[Design Artefacts – Data Flow Diagram (Iteration 1)](#_Toc523136578)

[Design Artefacts - Class Diagrams (Iteration 1)](#_Toc523136579)

[Design Artefacts - Class Diagrams (Iteration 2)](#_Toc523136580)

[Design Artefacts - Sequence Diagrams (Iteration 1)](#_Toc523136581)

[Design Artefacts - Sequence Diagrams (Iteration 2)](#_Toc523136582)

1. [5. Test Plan Design 31](#_Toc523136583)
2. [6. Evidence of using VCS (Version Control Software) 38](#_Toc523136584)

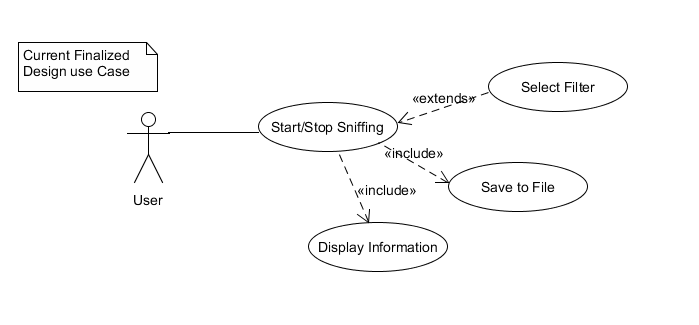
# 1. Project Overview

Designing of an Android Network Sniffer which in our team’s understanding is to achieve real-time scanning of network traffic and recording it to a file for later use. The application would be useful for people with Networking and Security background as the people who use this application would be System Administrators, Similar Application Developers and possibly general users. As such an application is already existing on computers our implementation of it onto a device like android has its limitation but it also thrills us to try and develop an actual working version of it.

## A) Vision

* Our key feature of the application would be real time capturing of packets and being able to see them displayed on the application.
* To be able to save packet data into a file for later view.
* After being able to achieve the first objective the second one would be to filter traffic based on certain criteria’s such as type of device or maybe specific type of traffic
* If possible be able to scan the access point for devices connected to it.

## B) Initial Use-Case Model



## C) Project Plan

* Due to the limitations of the Chipset our available device has been restricted and therefore plan to use Nexus 5 due to concerns such as device cost
* Due to the need to have root access, which may cause bricking to a phone we have opted to use older no longer in use devices
* The device needs to be in monitor/promiscuous mode our choice of device has been restricted.
* The rooting methods may or may not be suitable for our device so therefore we need to have alternate ways to get network traffic such as using VPN
* Using libraries from Pcap to assist with the project as well as tcp dump

# 2. Roles and Responsibilities

|  |  |  |  |
| --- | --- | --- | --- |
| Team Number : < **SS18/2A** > | | | |
|  | **Student Name** | **Role** | **Artefacts** |
| 1 | Soh Yu Xuan | Team Leader, Lead Programmer |  |
| 2 | Chin Tian Zhi Timothy | Lead Programmer |  |
| 3 | Huang Shun Yang Kenneth | Android UI Programmer |  |
| 4 | Tan Siyuan Kendrick | Team Manager, Lead Documenter | Technical Document, Initial Use case, research on products, SRS, Report |

# 3. Risk Analysis and Counter Measures

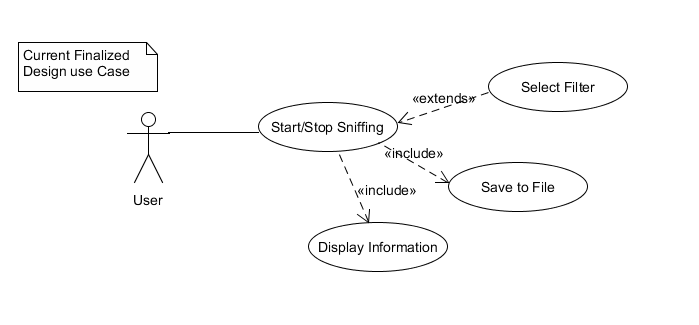
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Impact Type** | **Risk Seriousness (%)** | **Likelihood of Occurrence (%)** | **Risk Description** |
| 1 | Successful Task completion | 90% | 50% | Being able to complete rooting as it is the main criteria failing to do so will result in needing alternative methods which cause re-scope of project |
| 2 | Deadline | 90% | 10% | If deadline is not met all team members might fail the project |
| 3 | Unable to display on phone’s application | 50% | 20% | If application does not display correctly the bulk of the project might not work needing more time to debug |
| 4 | Unable to obtain device for project | 90% | 20% | If we are unable to have a device that meet our criteria |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Risk Description** | **Proposed Mgmt Plan** | **(Possible) Reduction in Risk Seriousness (%)** |
| 1 | Unable to root phone | Plan 1 : Search for alternative methods like VPN | -50% |
| Plan 2 : use alternate rooting method till successful | -25% |
| 2 | Unable to obtain device for project | Plan 1: Buy from people who are selling old phones | 75% |
| 3 |  |  |  |

# 4. UML Diagrams

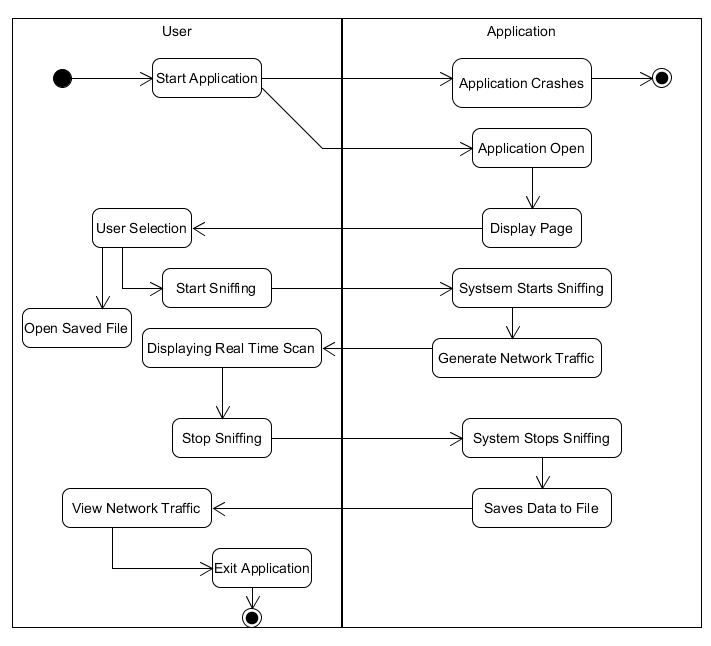
## Design Artefacts - Use Cases (Iteration 1)

Main use case diagram:

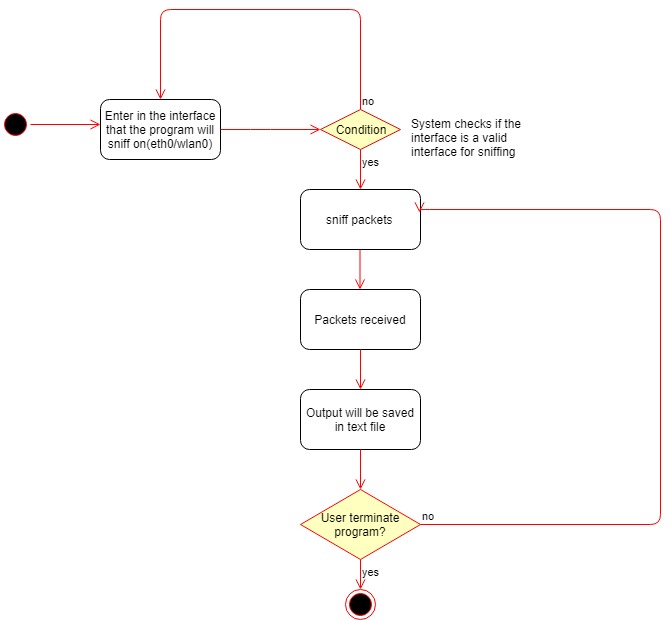


## Design Artefacts – Activity Workflows (Iteration 1)

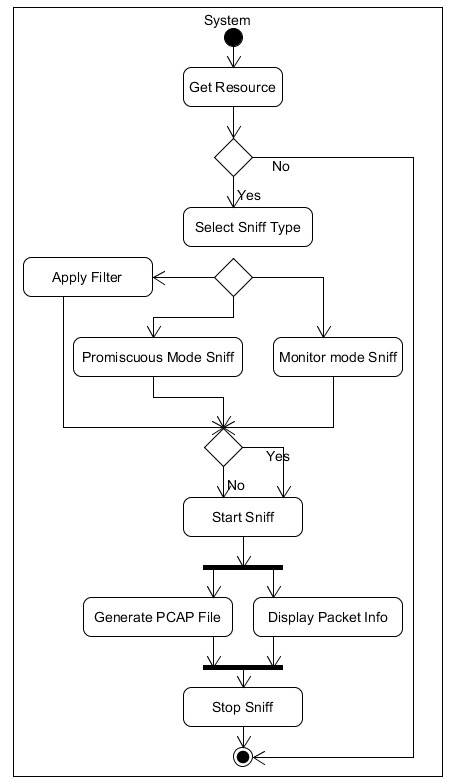
Android app activity diagram:

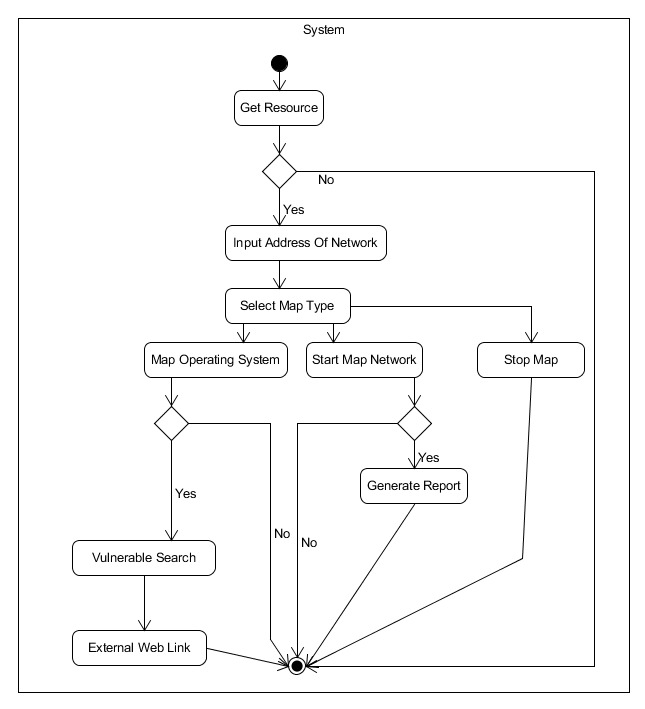
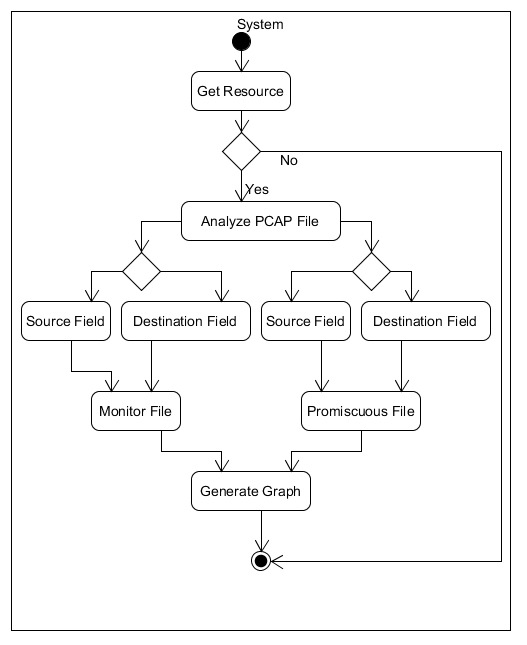


Pcbin activity diagram:



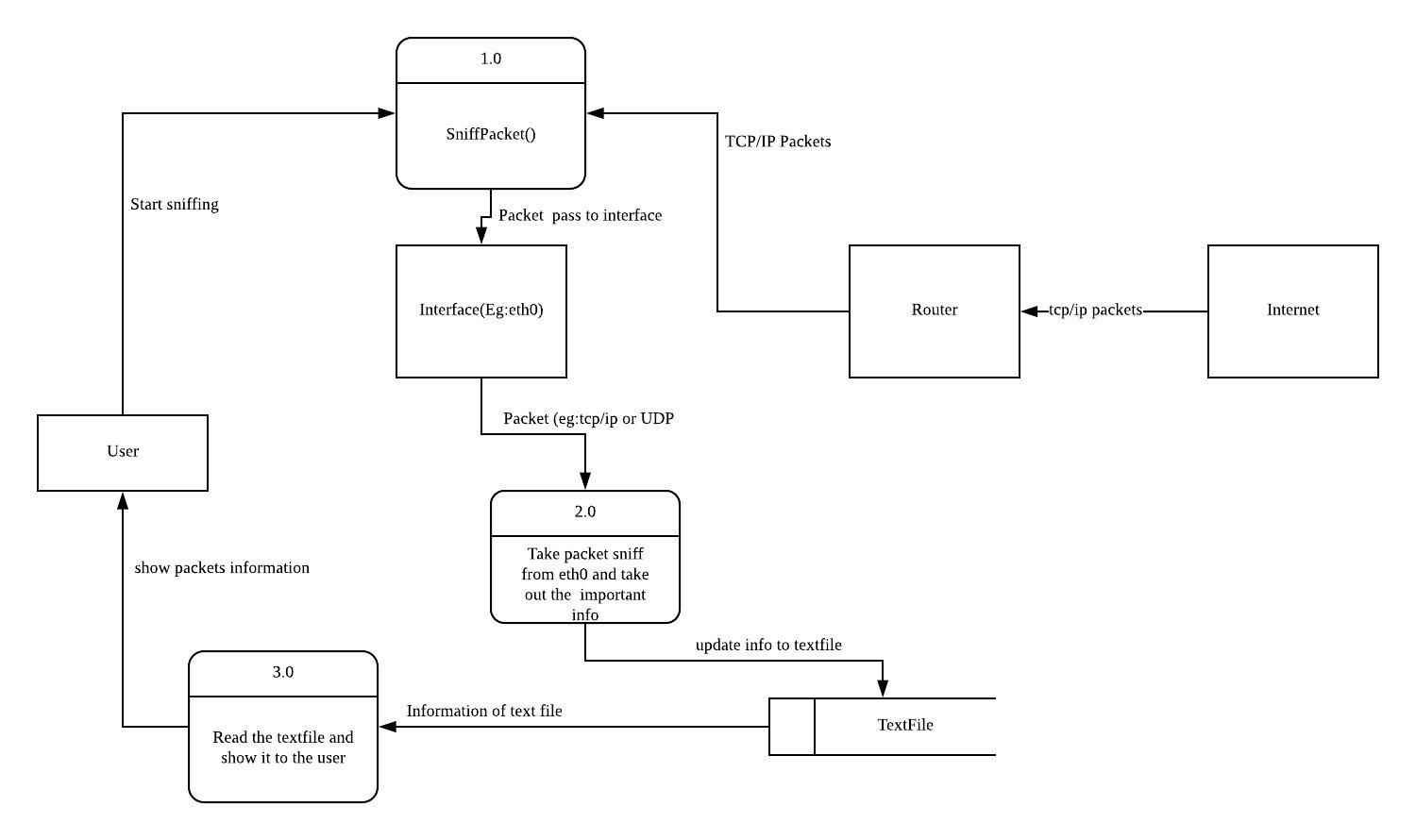
## Design Artefacts – Activity Workflows (Iteration 2)

****

****

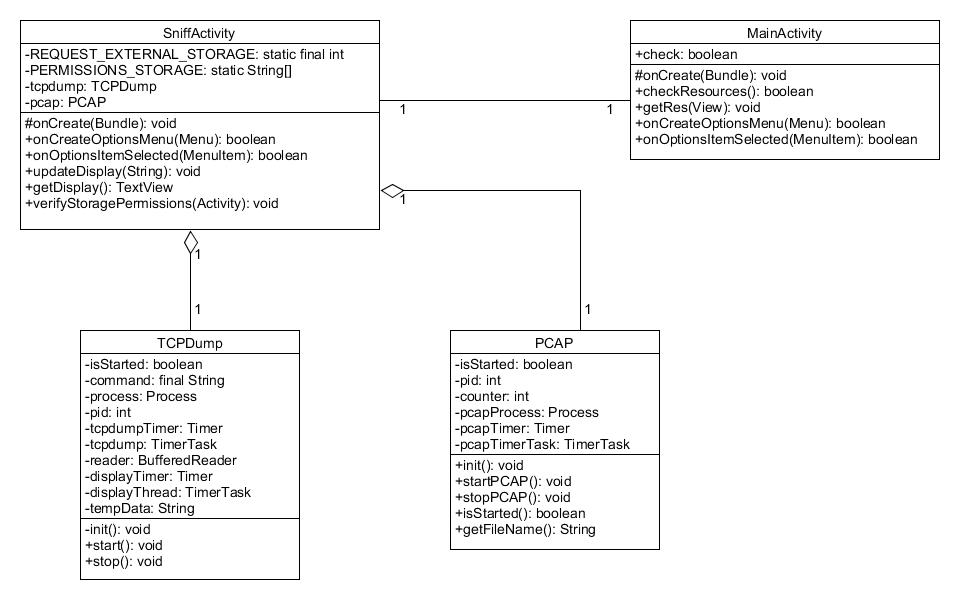
## Design Artefacts – Data Flow Diagram (Iteration 1)

Pcbin data flow diagram:

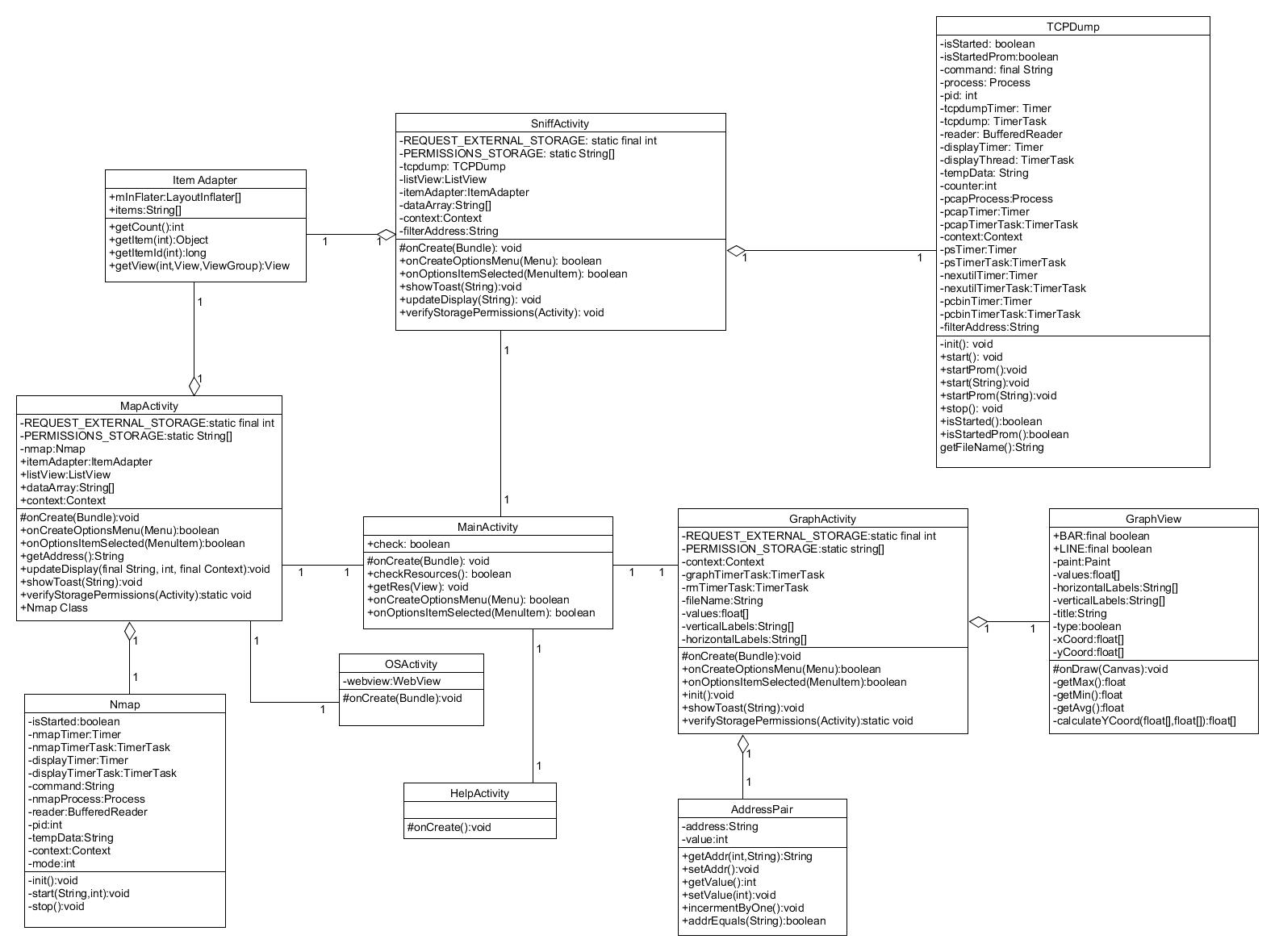


## Design Artefacts - Class Diagrams (Iteration 1)

Main class diagram:

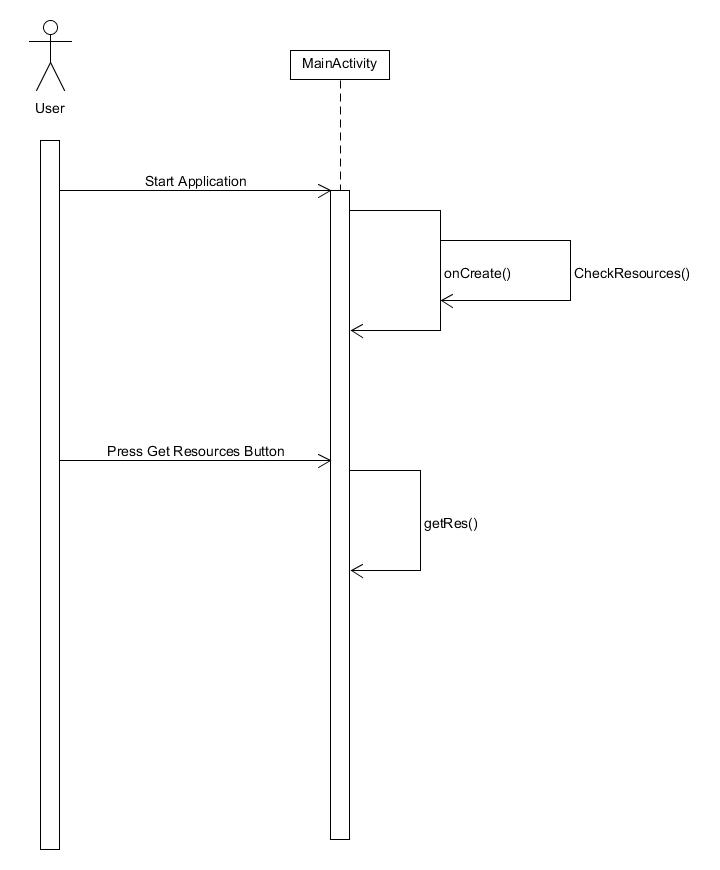


## Design Artefacts - Class Diagrams (Iteration 2)

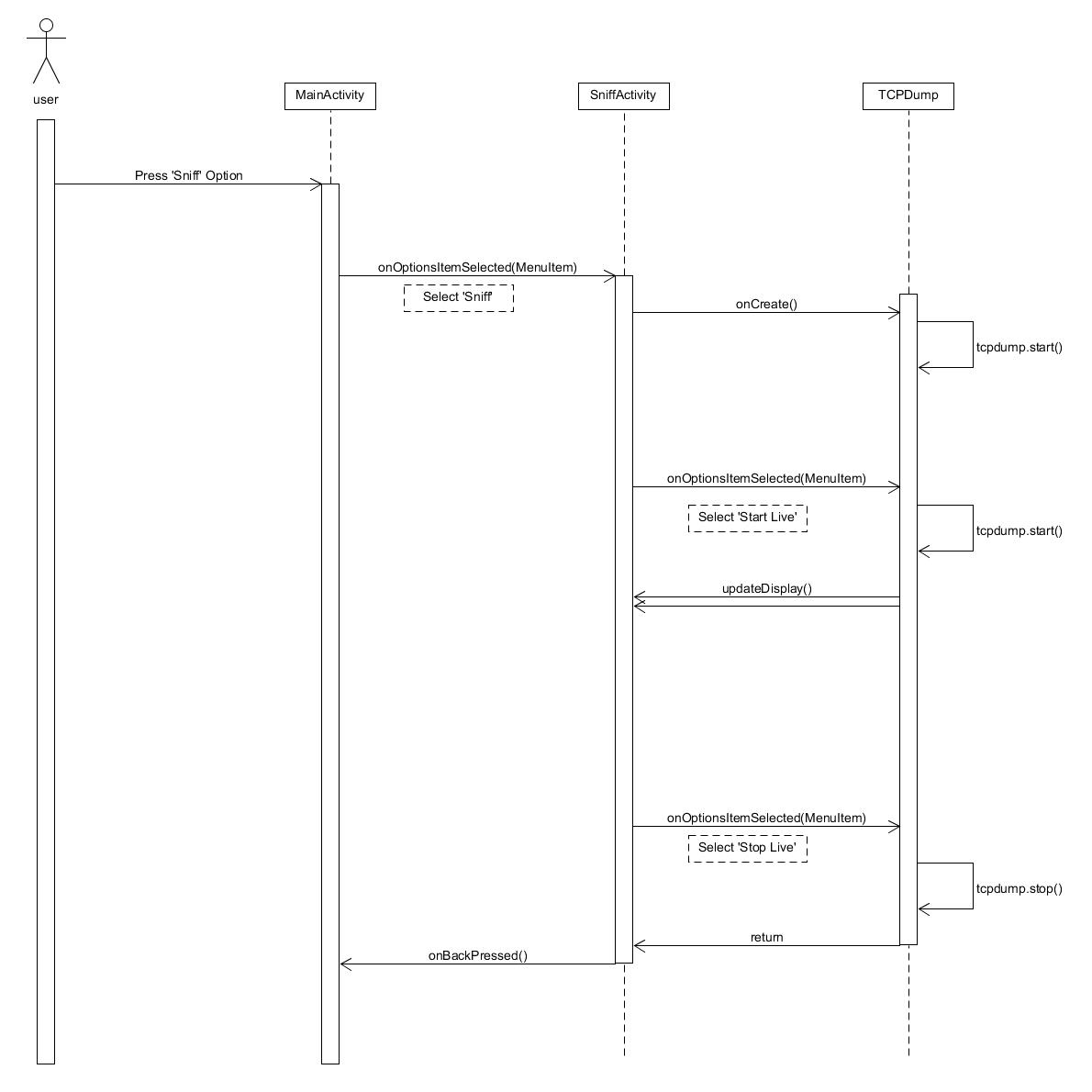


## Design Artefacts - Sequence Diagrams (Iteration 1)

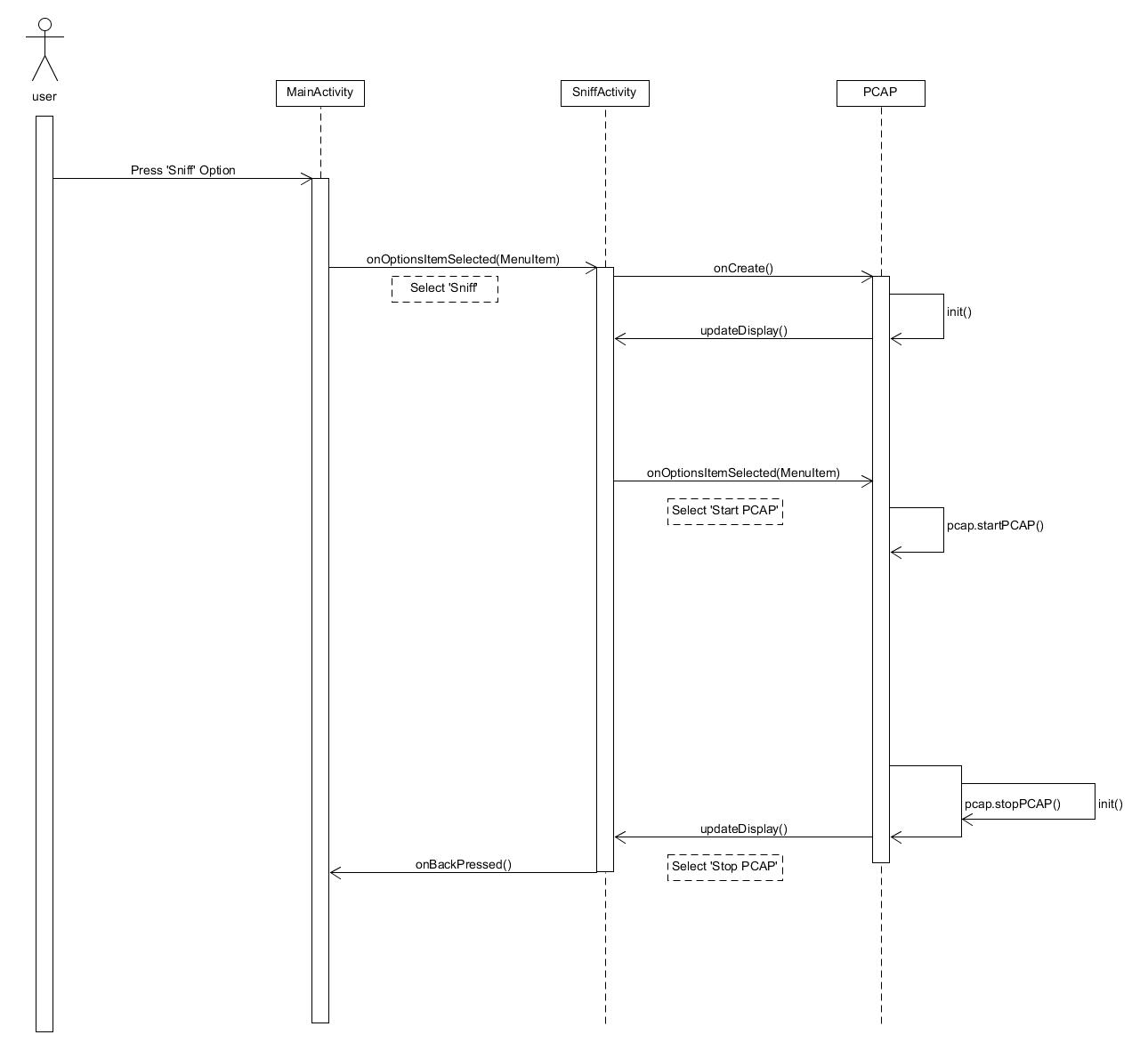
App initialize sequence diagram:



Start live sequence diagram:

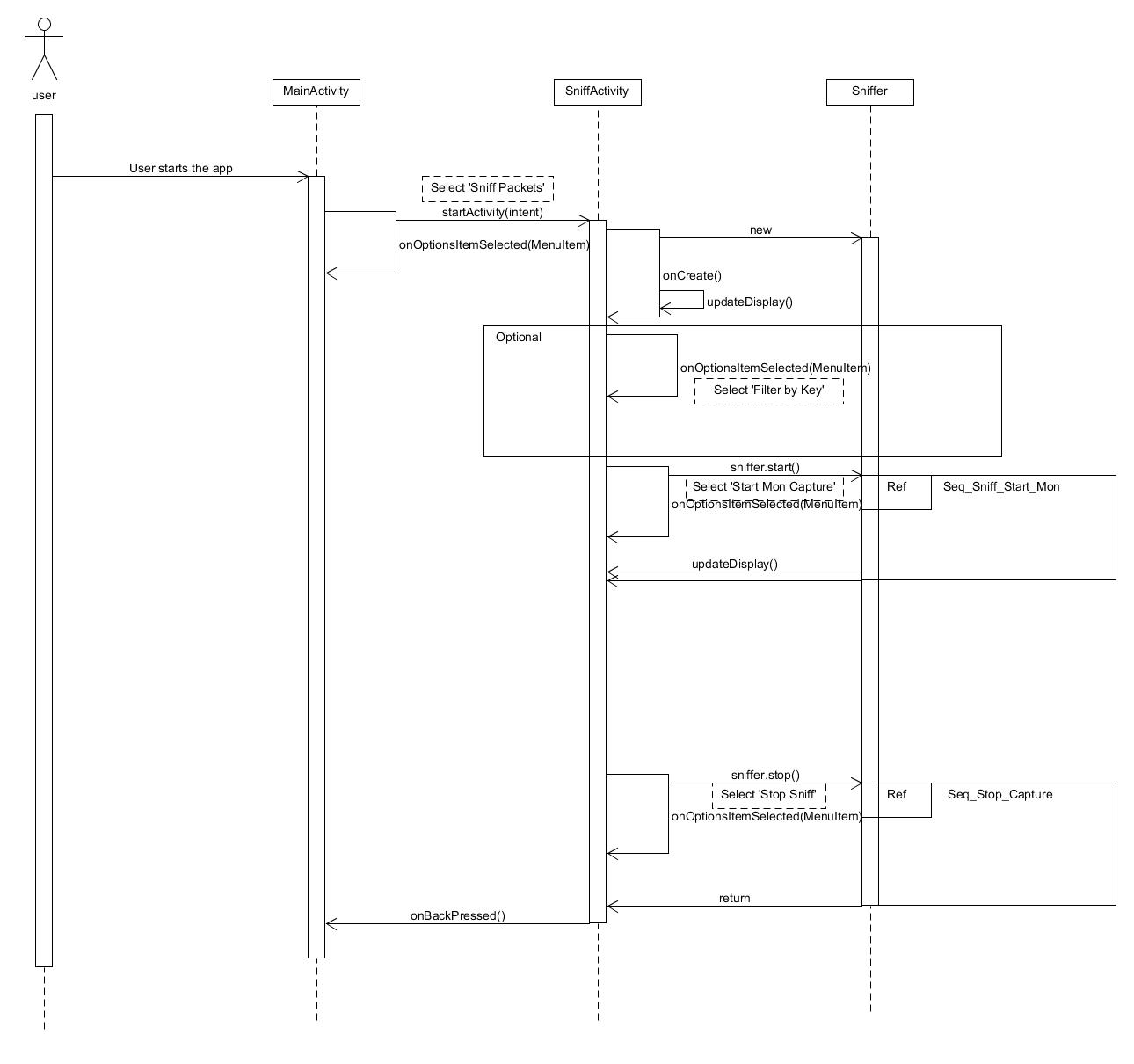


Start pcap sequence diagram:

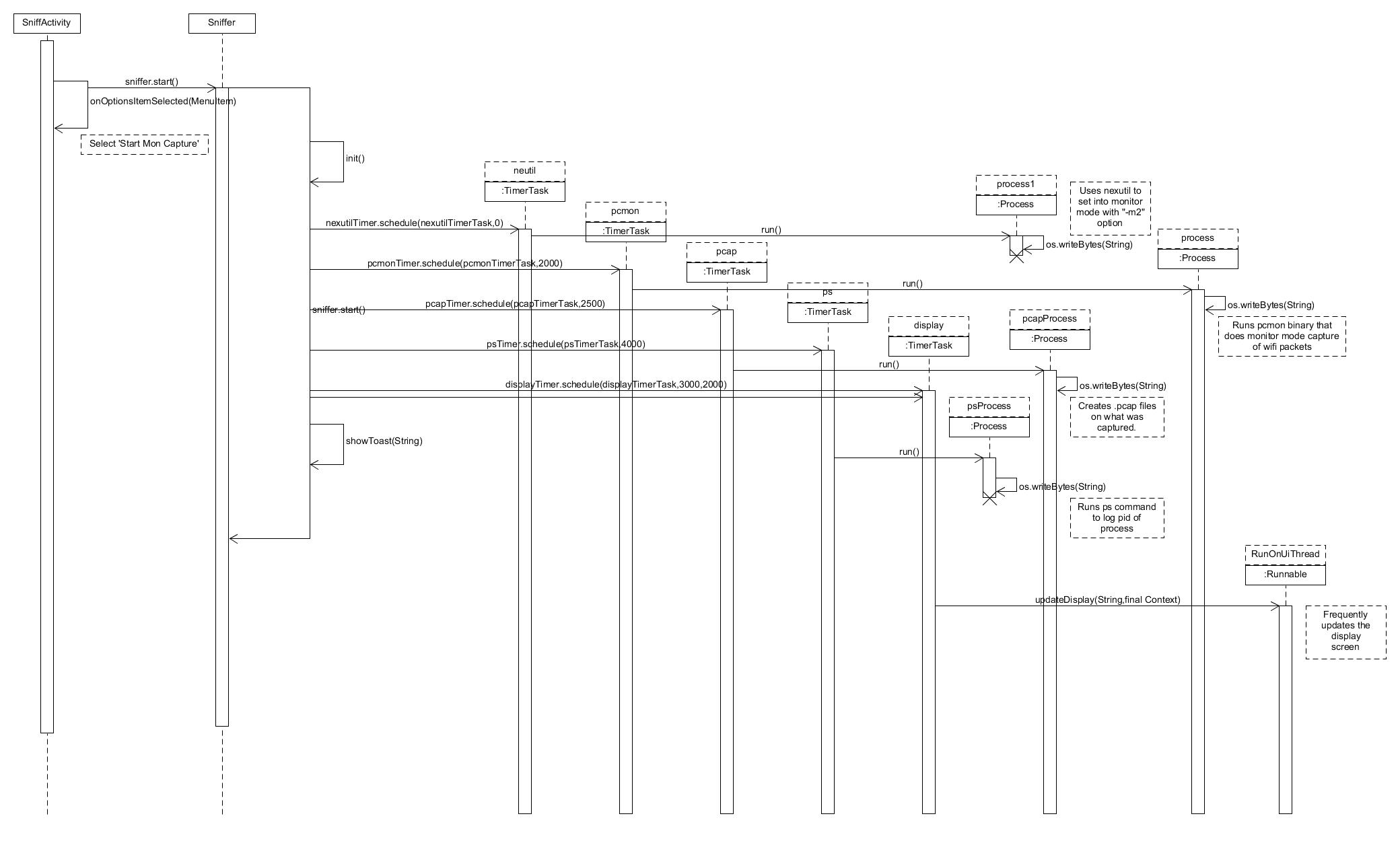


## Design Artefacts - Sequence Diagrams (Iteration 2)

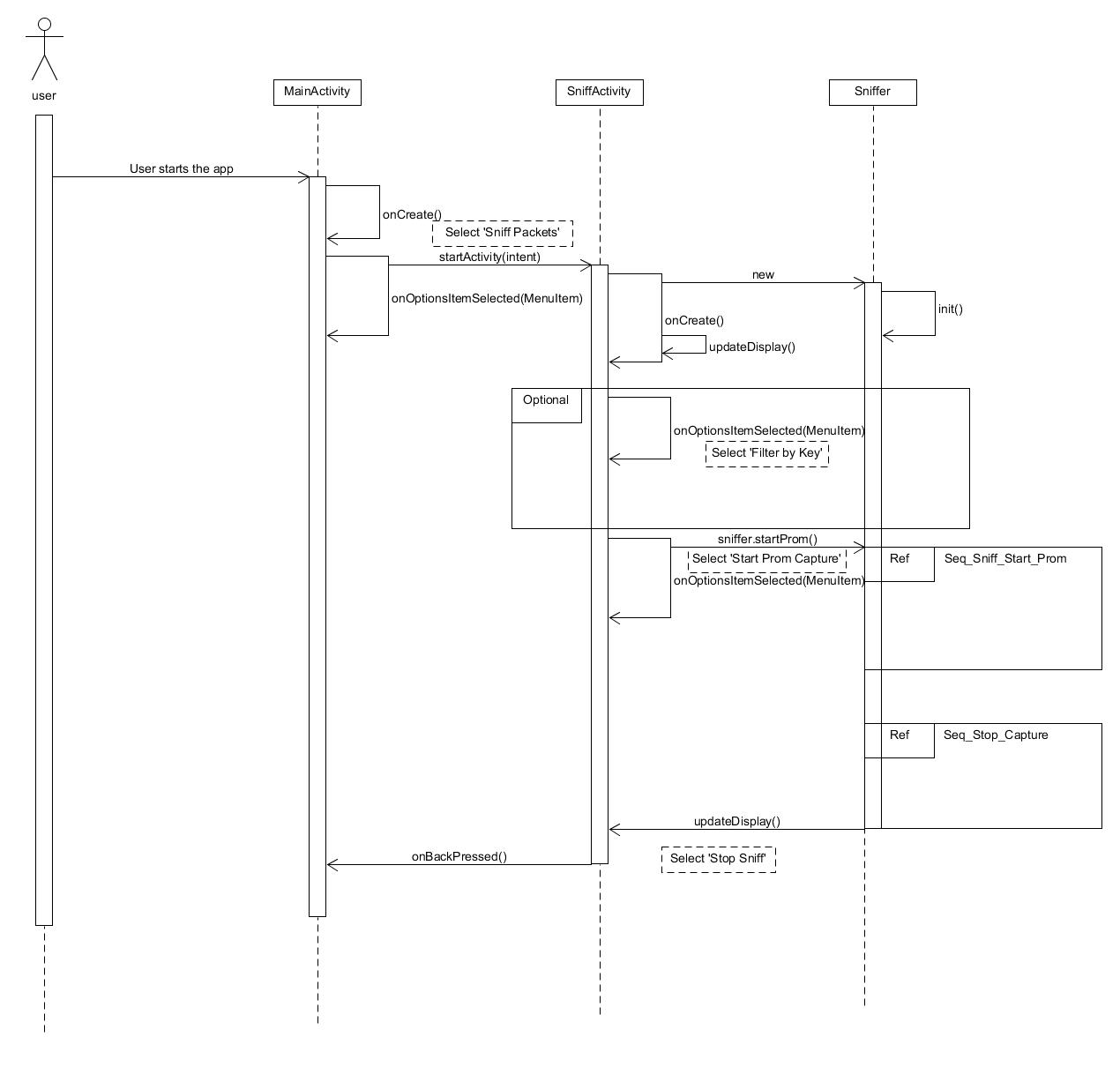
[UC-1.3] Start\_Mon\_Capture



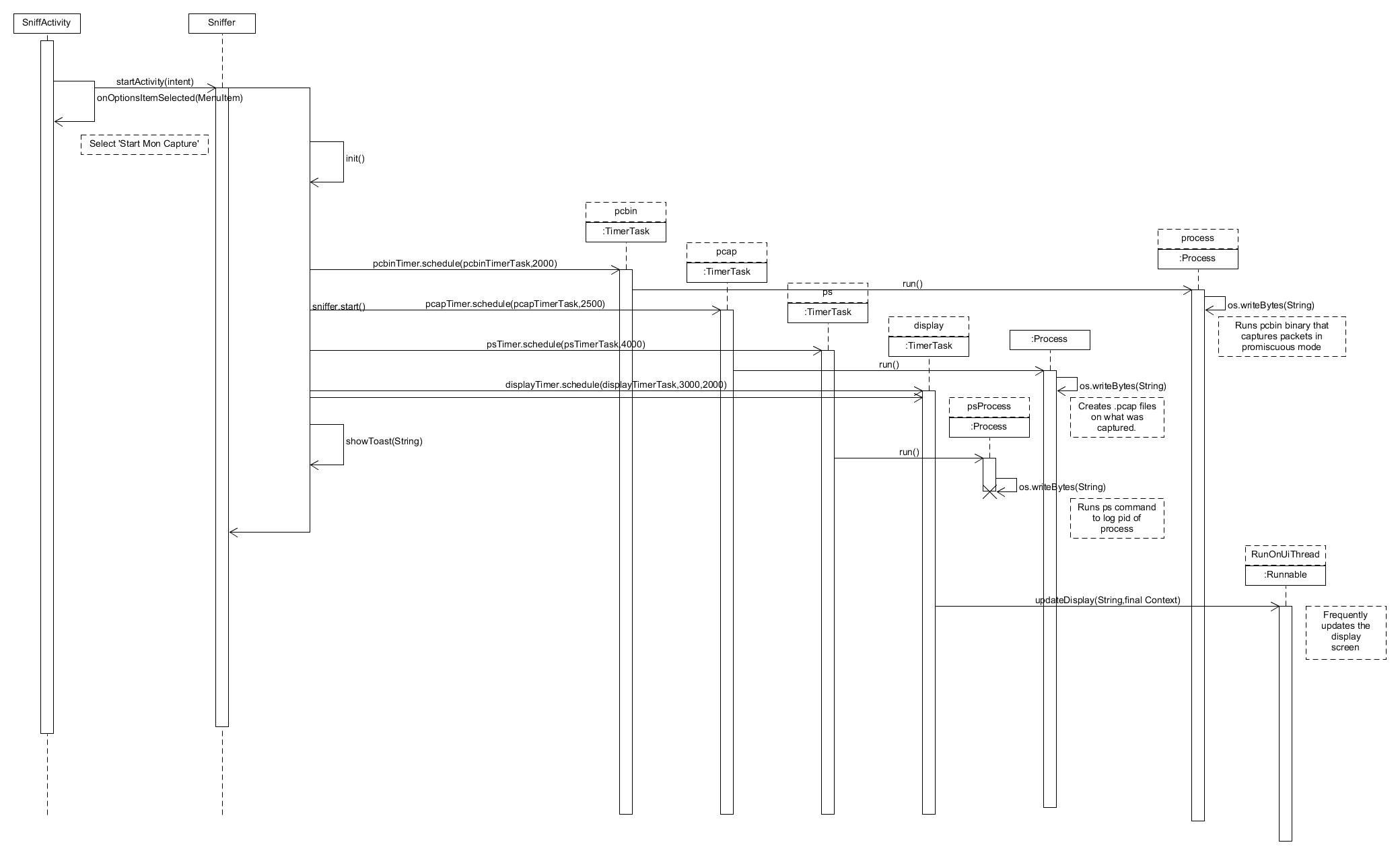
[UC-1.3] Sniffer\_Start\_Mon [ sniffer.start() ]



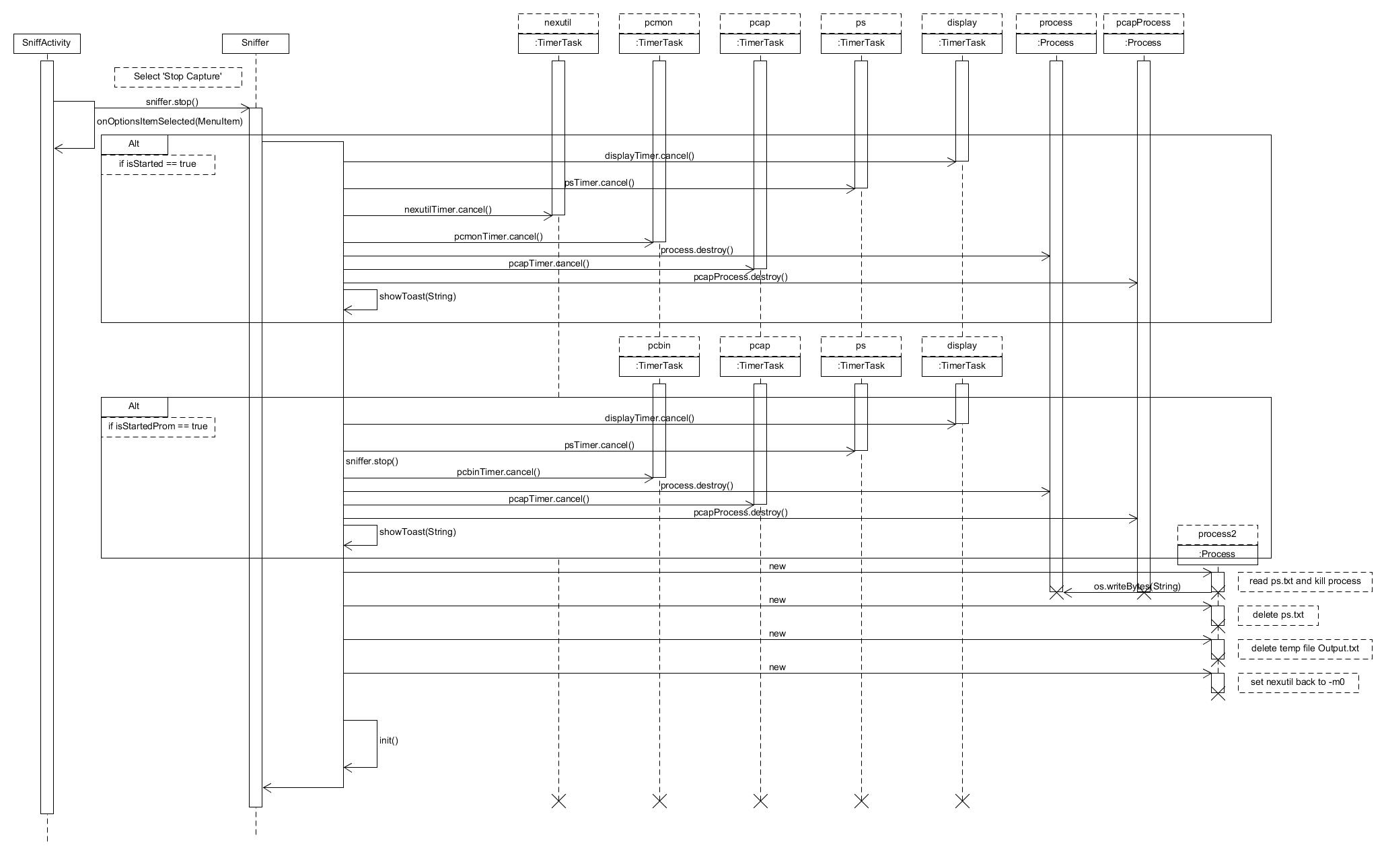
[UC-1.4] Start\_Prom\_Capture



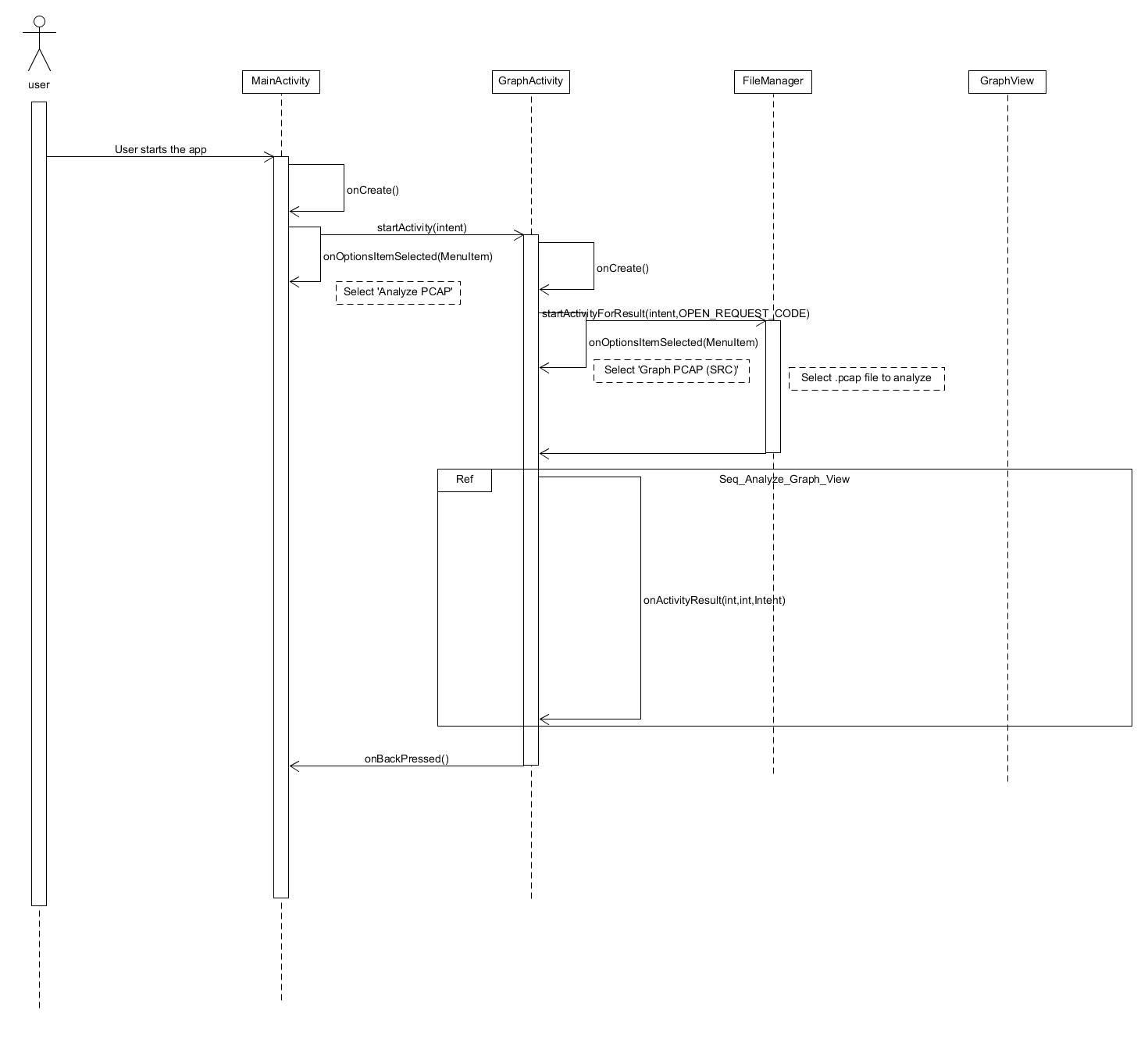
[UC-1.4] Sniff\_Start\_Prom [ sniffer.startProm() ]



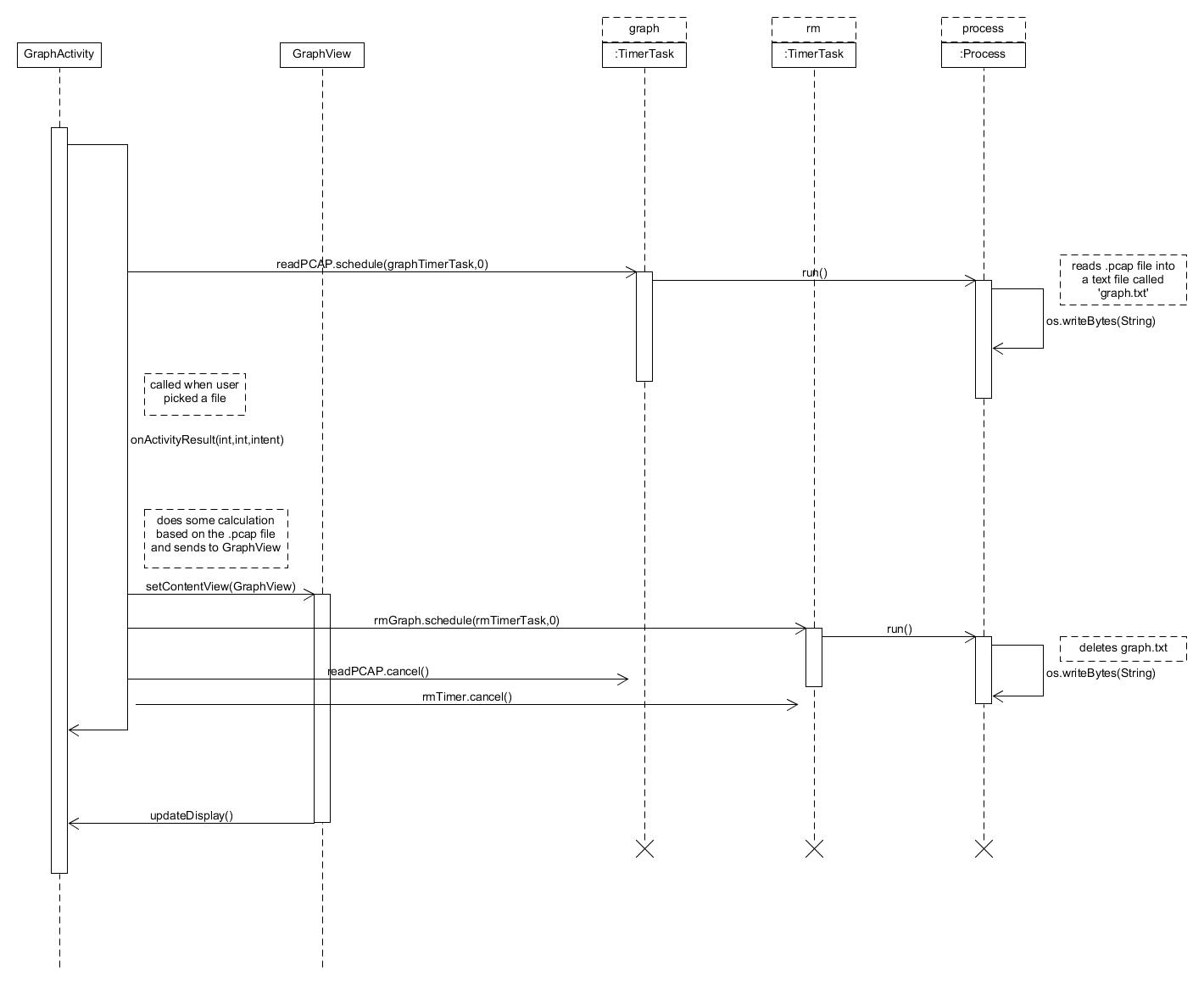
[UC-1.2] Stop\_Capture(Monitor & Promiscous)



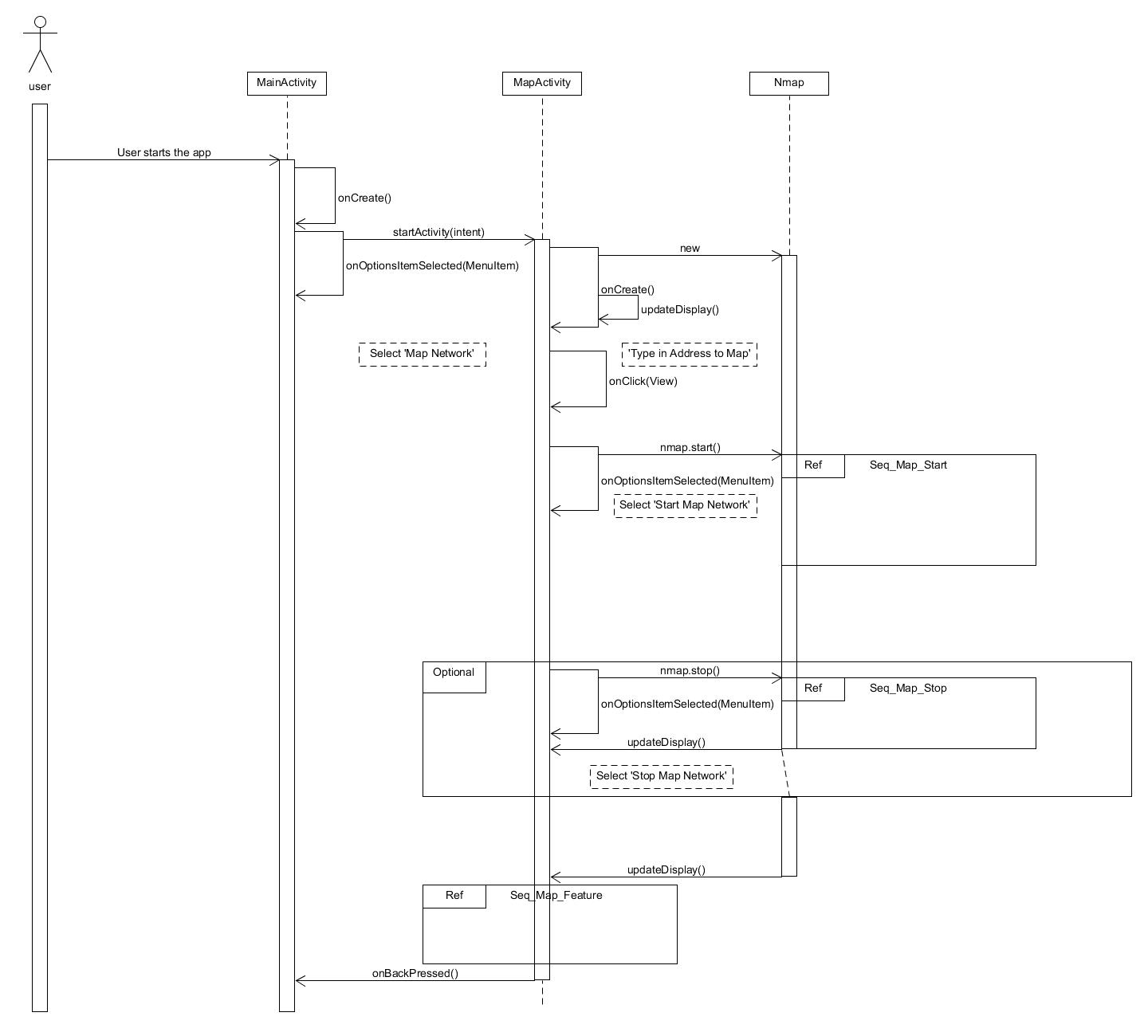
[UC-4] Analyze\_PCAP



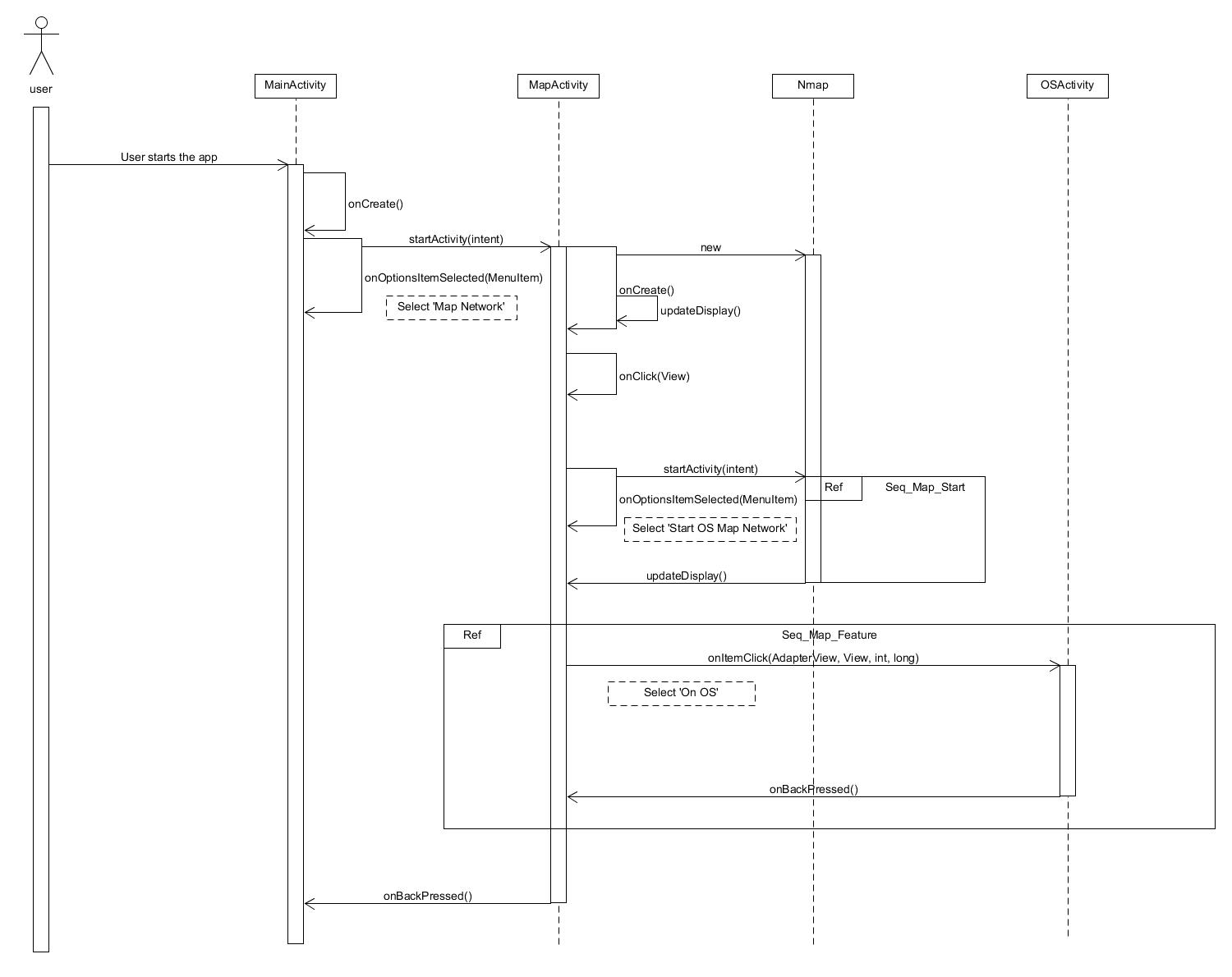
{UC-4] Analyze\_Graph\_View



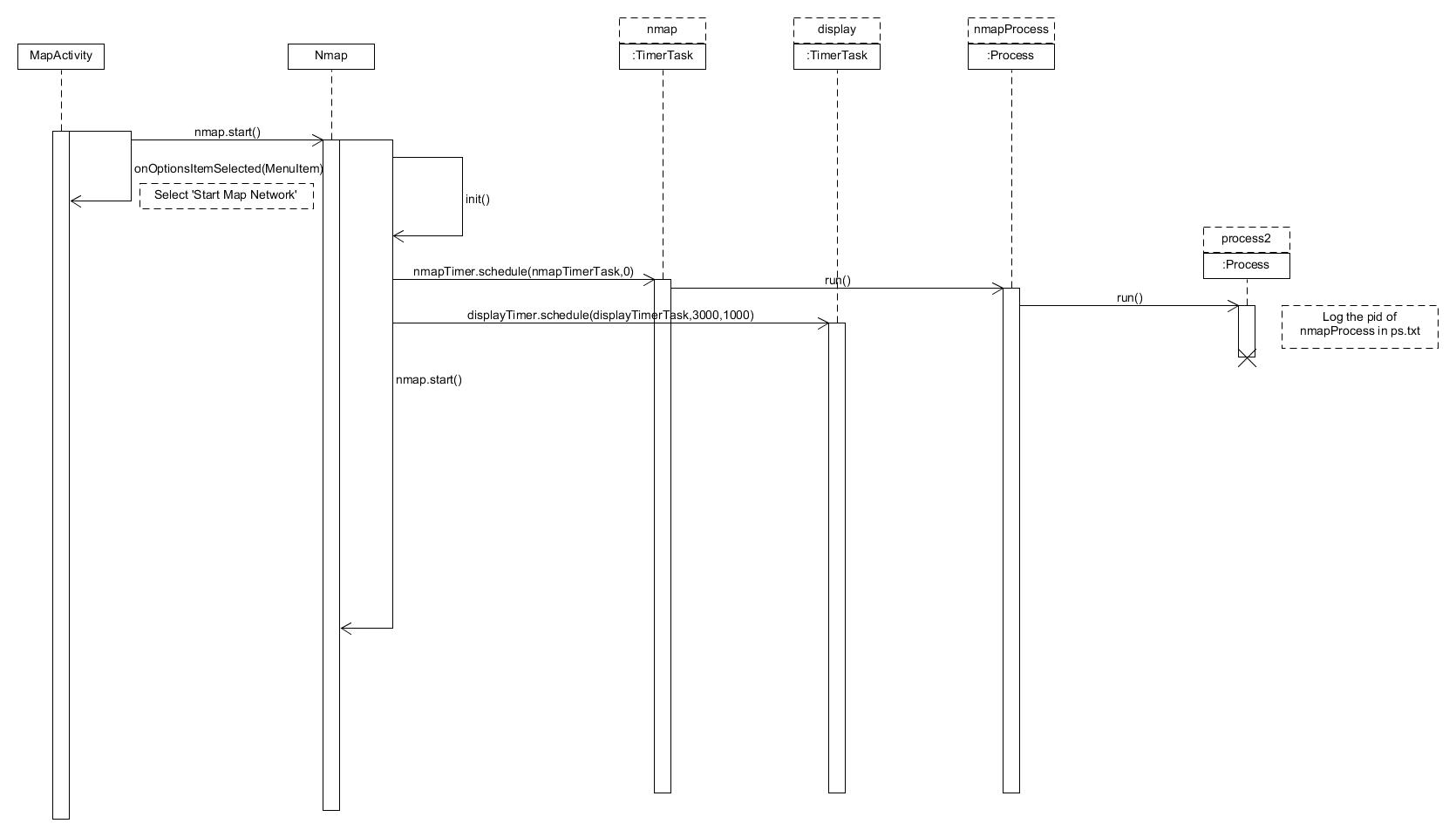
[UC-3.1] Start\_Map\_Network



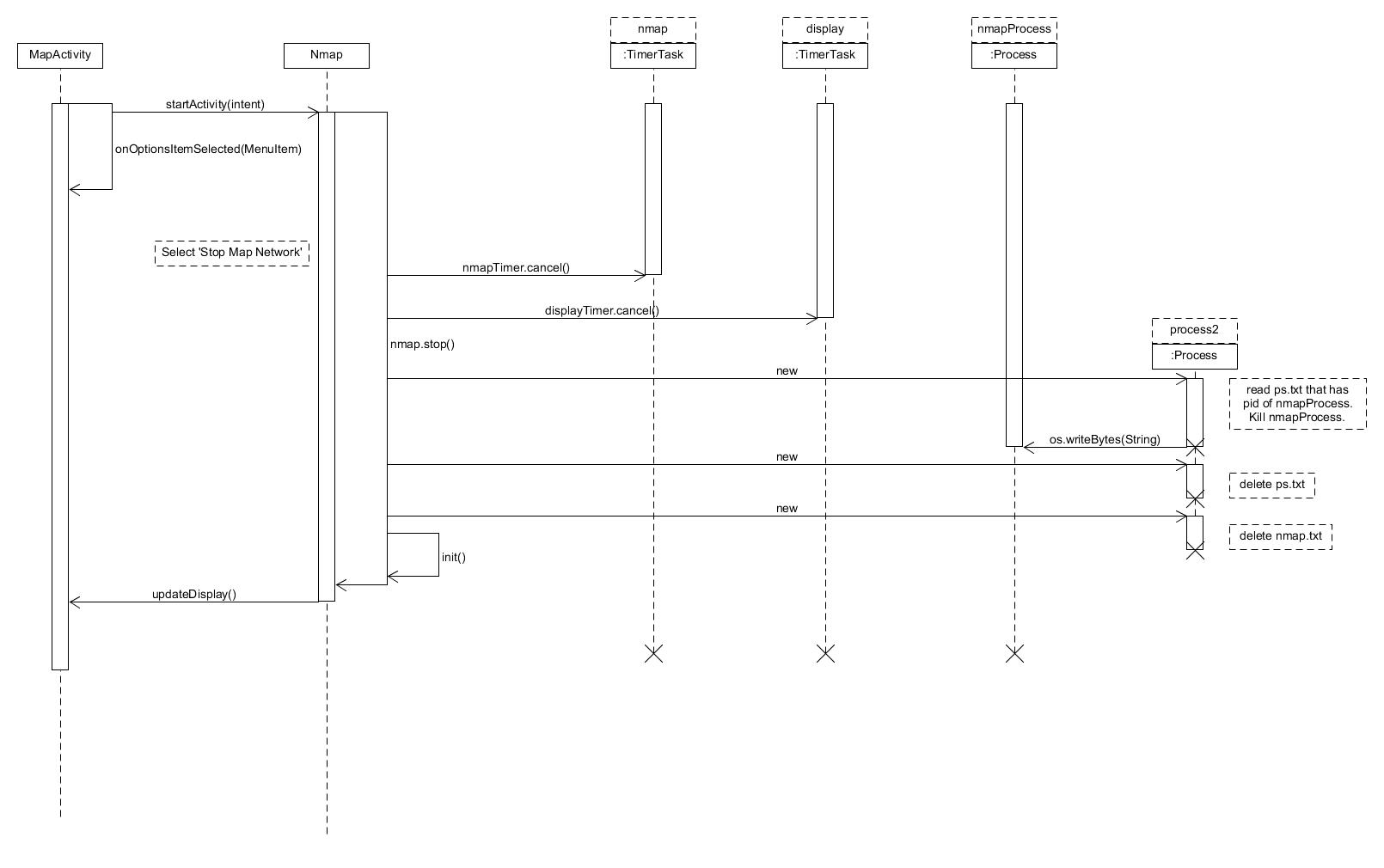
[UC-3.2] Start\_OS\_Map\_Network



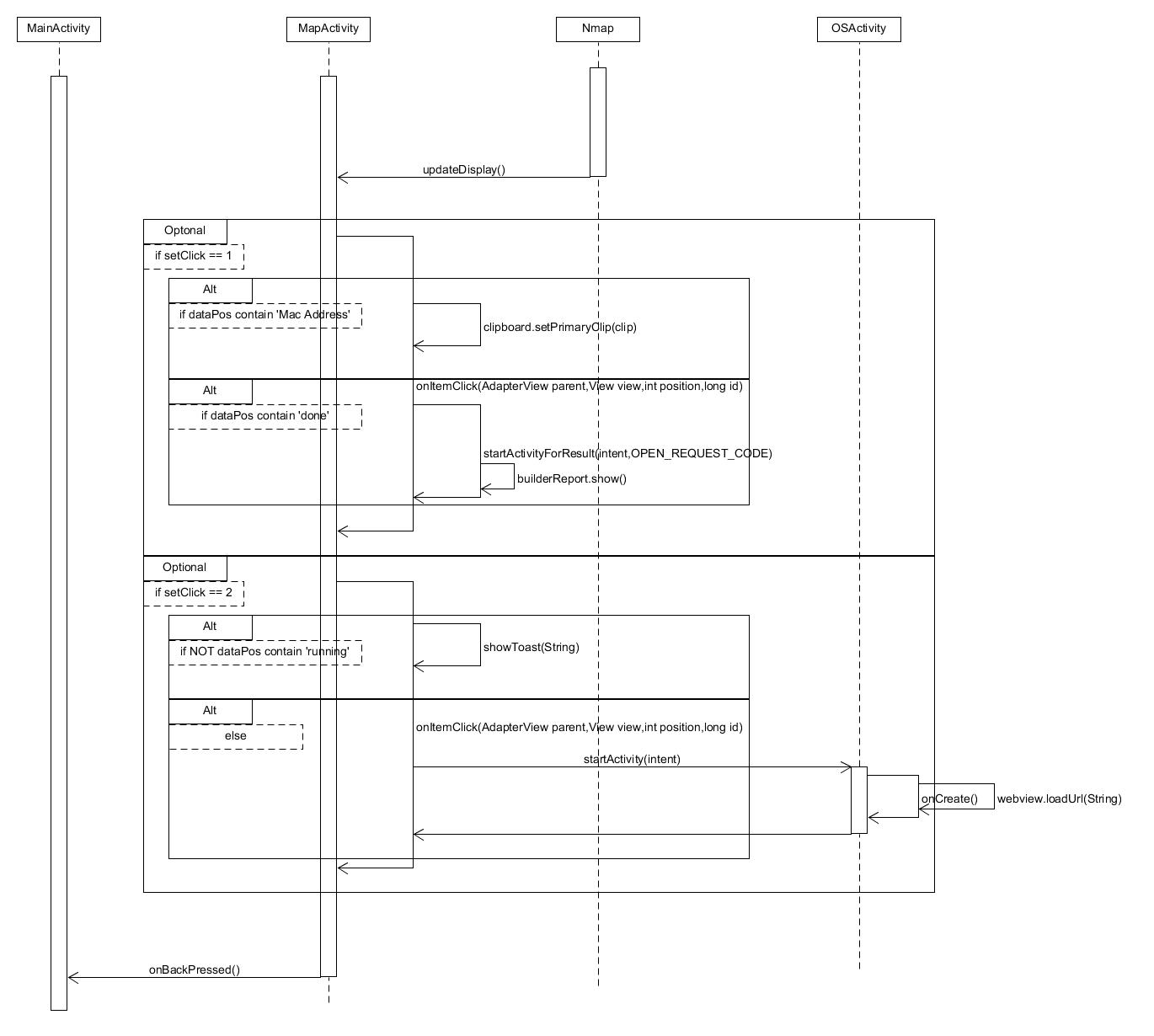
[UC-3.1/3.2] Map\_Start



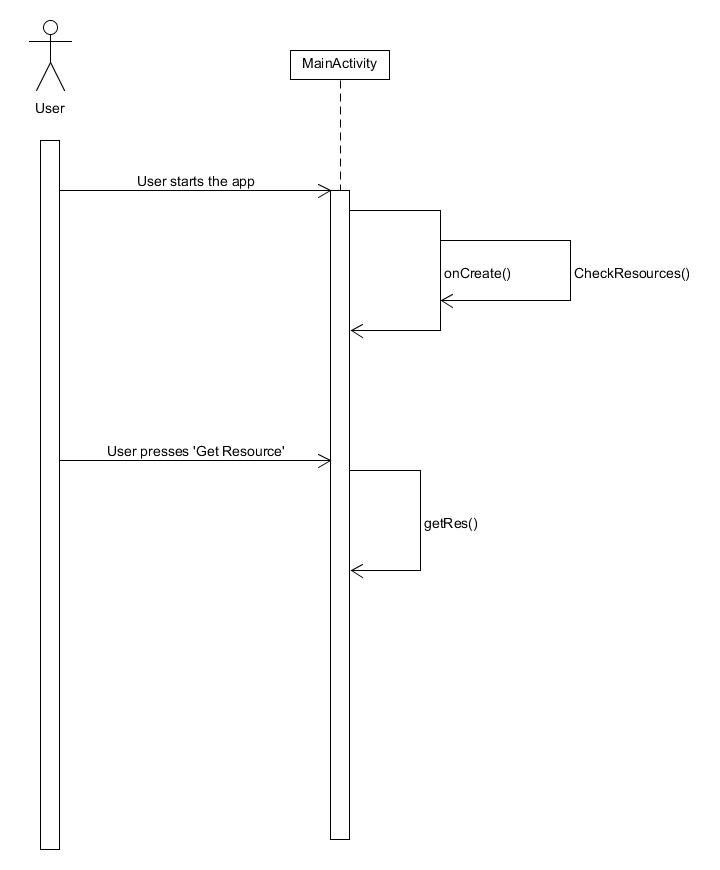
[UC-3.3] Map\_Stop



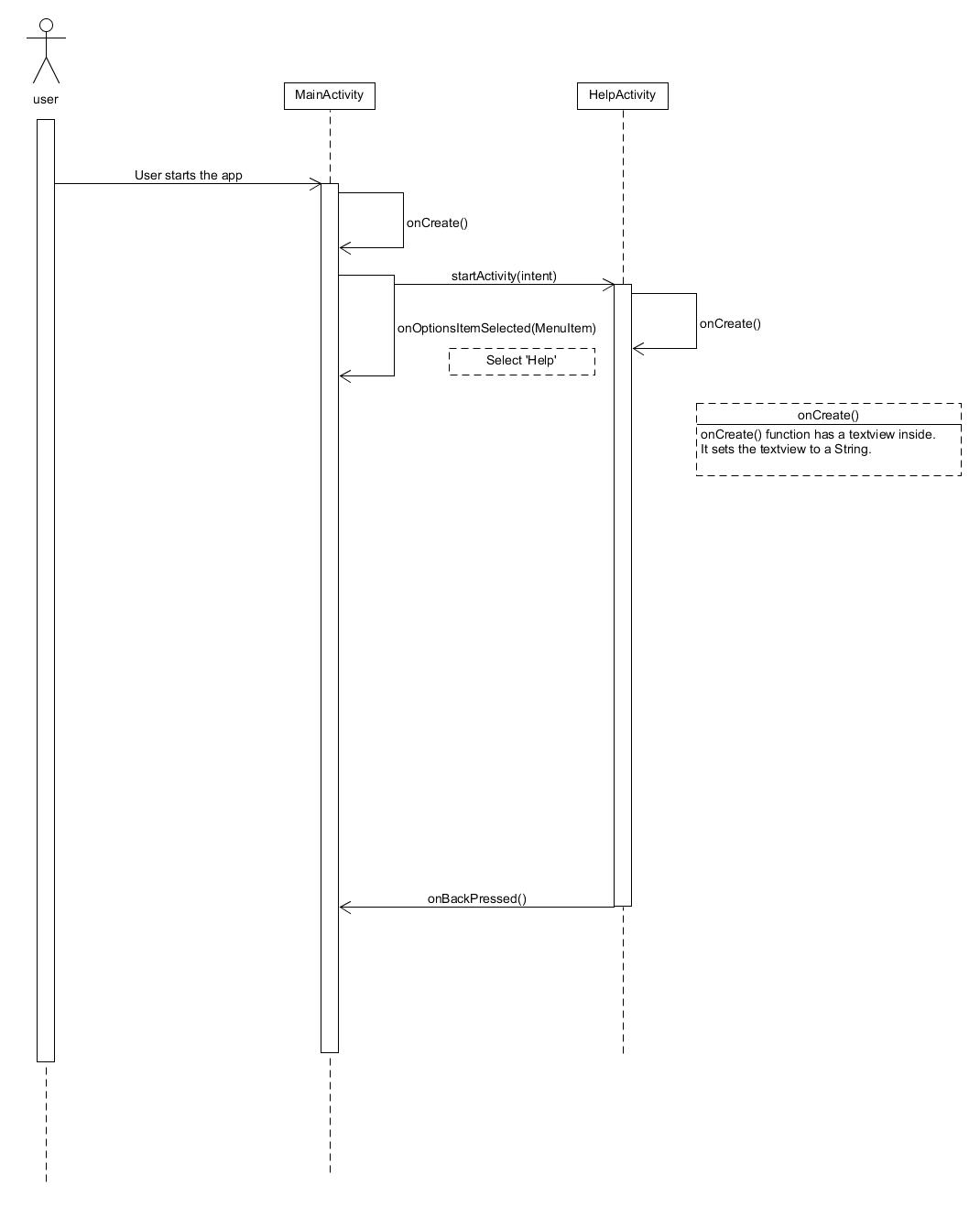
[UC-3.1.2/3.2.1] Map\_Feature



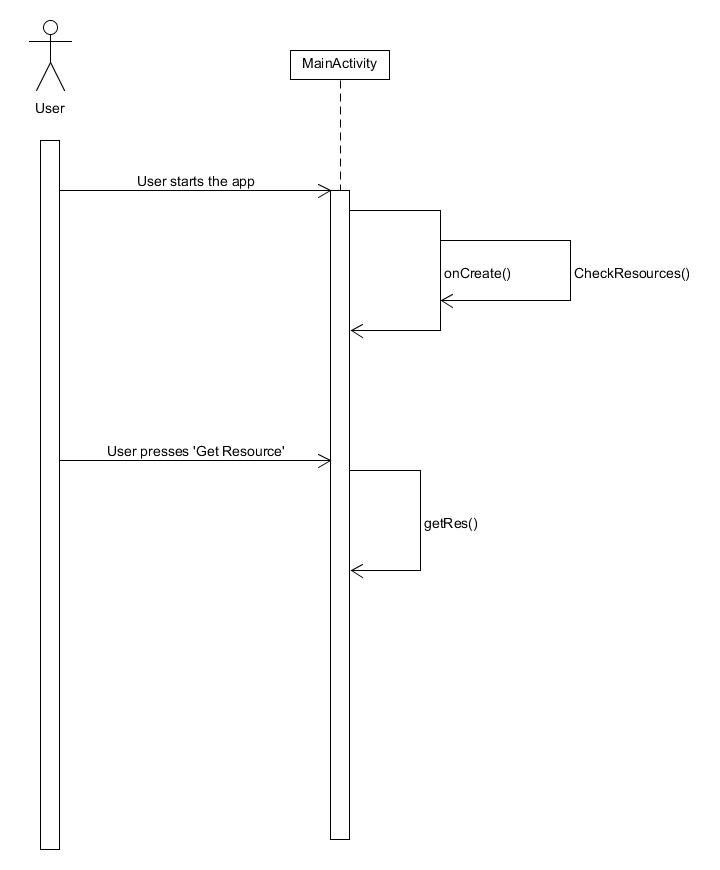
[UC-2] Check\_Resource



[UC-5] Help



[UC-2] Check\_Resource



# 5. Test Plan Design

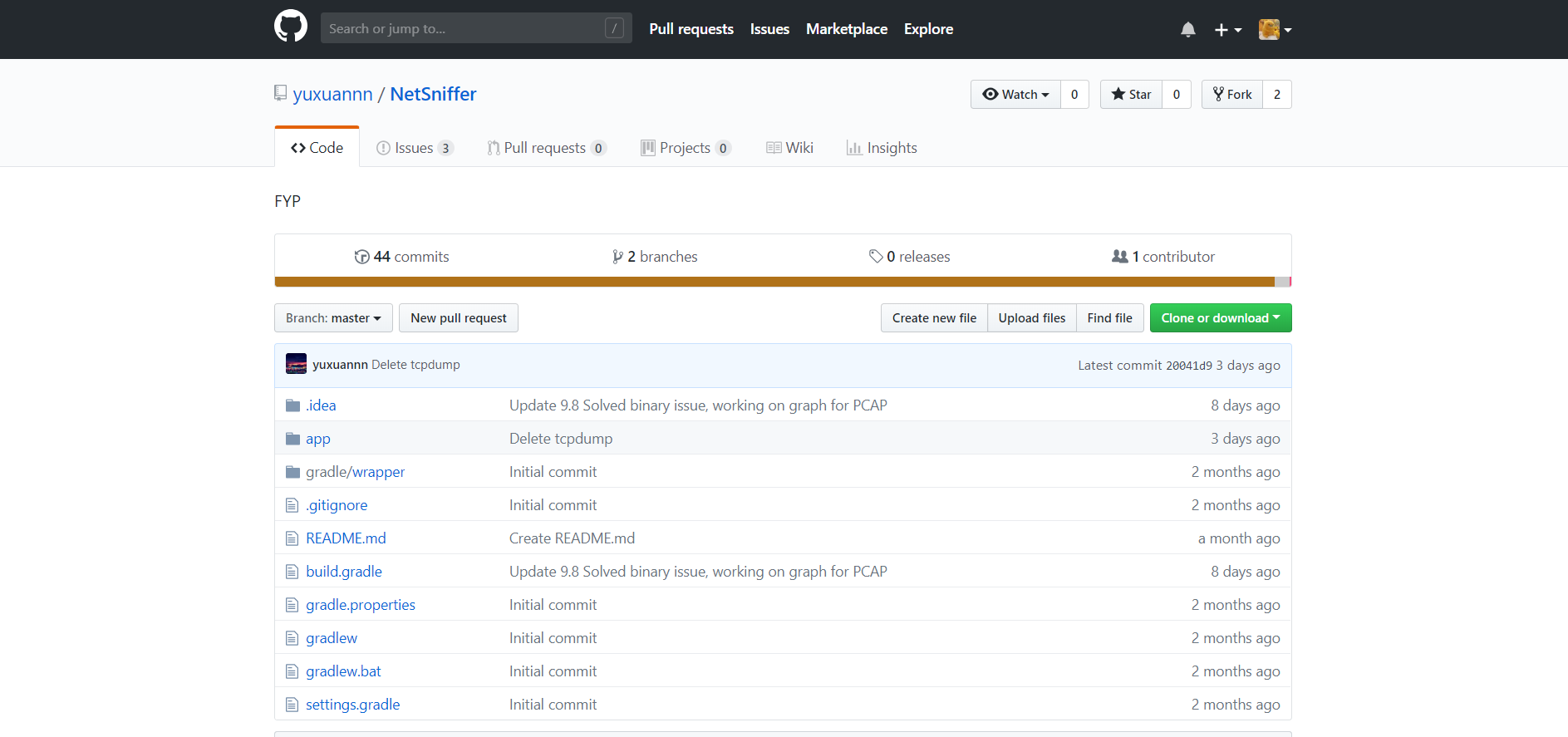
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Plan for :  Android Network Sniffer - Being able to capture/sniff packets | | | | |
| Test Case ID: | Description | Expected Outcome | Participating Program's Class & Method to be tested | Actual Outcome & If Issue was Solved |
| ANS-1 (Sniff) [UC-1.3] | To test if the Monitor mode(pcmon binary) packet sniffing works, being able to capture and see packets. Start Sniffing/Capture functions. | System should be able to receive and display packets to the display screen. The screen should show information relating to packets. | Class Name :  SnifferActivity, Sniffer.  Method Name :  void Sniffer.start() | At first, nothing was being output. Found the issue with threads, processes, update to ui.  Solved. |
| ANS-2 (Sniff) [UC-2] | To Test if the “Get Resource” button will trigger any errors if pressed multiple times, be error free. | System should not crash or have any errors if the button is pressed multiple times. | Class Name :  MainActivity  Method Name :  void getRes() | Pressing many times would “queue” the action up, resulting in efficiency issues. Now checks if has resources first.  Solved. |
| ANS-3 (Sniff) [UC-1.4] | To test if Promiscuous Mode(pcbin binary) packet sniffing works, see if can capture and see packets. | System should be able to receive and display packets to the display screen. The screen should show information relating to packets. | Class Name :  SnifferActivity, Sniffer.  Method Name :  void Sniffer.startProm() | Works. |
| ANS-4 (Sniff) [UC-1.3.1] | To test if sniffing of packets both Monitor and Promiscuous are able to create pcap files after sniffing is stopped. | System should have a pcap file created in its storage, at “sdcard/download” directory. | Class Name :  SnifferActivity, Sniffer.  Method Name :  void Sniffer.start() & void Sniffer.startProm() | Works. |
| ANS-5 (Sniff) [UC-1.1] | To test if sniffing filter works properly, be able to show filter only related results. | System should filter results based on keyword entered in the text and display those to screen. | Class Name :  SnifferActivity, Sniffer.  Method Name :  void Sniffer.start() & void Sniffer.startProm() | Was case-sensitive.  Solved. |
| ANS-6 (Map) [UC-3.1] | To test if Map Network works properly and displays to the screen, be able to show devices connected to network. | System should be able to generate the devices connected to the network and show them to the display screen. | Class Name :  MapActivity, Nmap  Method Name :  void Nmap.start(String) | Did not work, nothing displayed to screen. Figured out needed nmapservices to run.  Solved. |
| ANS-7 (Map) [UC-3.1.2] | To test if “done” choose file doesn’t crash when trying to generate report, being able to generate report. | System should be able to generate a report displayed to the screen showing Mac Addresses. | Class Name :  MapActivity, Nmap  Method Name :  void updateDisplay(final String, int mode, final Context), void onItemClick( AdapterView, View, int, long ) | Report generated had weird output.  Solved. |
| ANS-8 (Map) [UC-3.1] | To test if “Copy Mac Address” works properly, be able to copy address to clipboard. | System should be able to copy the Mac Address properly and be able to paste elsewhere. | Class Name :  MapActivity,Nmap  Method Name :  updateDisplay(final String, int mode, final Context), onItemClick( AdapterView, View, int, long ) | Works. |
| ANS-9 (Map) [UC-3.2] | To test if Map OS works properly, being able to show OS of device. | System should be able to obtain the OS of the device and display to screen. | Class Name :  MapActivity, Nmap  Method Name :  void Nmap.start(String) | Works. |
| ANS-10 (Map) [UC-3.2.1] | To test if Map OS can properly launch OS vulnerability search, be able to show OS vulnerability to screen. | System should be able to launch vulnerability list depending on OS received with no error. | Class Name :  Map Activity, Nmap  Method Name :  updateDisplay(final String, int mode, final Context), onItemClick( AdapterView, View, int, long ), webView(String) | Crashed when didn’t match “Running”, did not launch webpage related to OS vulnerabilities.  Solved. |
| ANS-11 (Binary) [UC-1.4] | To test that binary Pcbin can get IP packet info properly. | System be able to receive IP packets smoothly. | Class Name :  Pcbin.c  Method Name :  void sniffPacket(char\*)  void getPacket(u\_char \*, const struct pcap\_pkthdr \*, const u\_char\*)  void print\_IP(const struct Ip \*ip,char\*) | Works. |
| ANS-12 (Binary) [UC-1.4] | To test that binary Pcbin can get TCP packet info properly. | System be able to receive TCP packets smoothly. | Class Name :  Pcbin.c  Method Name :  void sniffPacket(char\*)  void getPacket(u\_char \*, const struct pcap\_pkthdr \*, const u\_char\*)  void print\_tcp(u\_char\* , const struct pcap\_pkthdr\* , const u\_char \*, char \*, const struct Ip \*) | Works. |
| ANS-13 (Binary) [UC-1.4] | To test that binary Pcbin can get UDP packet info properly. | System be able to receive UDP packets smoothly. | Class Name :  Pcbin.c  Method Name :  void sniffPacket(char\*)  void getPacket(u\_char \*, const struct pcap\_pkthdr \*, const u\_char\*)  void print\_udp(u\_char\* , const struct pcap\_pkthdr\* , const u\_char \*, char \*, const struct Ip \*) | Works. |
| ANS-14 (Binary) [UC-1.4] | To test that binary Pcbin can get DNS packet info properly. | System be able to receive DNS packets smoothly. | Class Name :  Pcbin.c  Method Name :  void sniffPacket(char\*)  void getPacket(u\_char \*, const struct pcap\_pkthdr \*, const u\_char\*)  void print\_IP(u\_char\* , const struct pcap\_pkthdr\* , const u\_char\*, char\*, const struct Ip\*,const struct Dns\*) | Weird Output, wrong packet structure, compare .pcap file in wireshark, analyse structure.  Solved. |
| ANS-15 (Binary) [UC-1.4] | To test that binary Pcbin can get ARP packet info properly. | System be able to receive ARP packets smoothly. | Class Name :  Pcbin.c  Method Name :  void sniffPacket(char\*)  void getPacket(u\_char \*, const struct pcap\_pkthdr \*, const u\_char\*)  void printARP(u\_char\*, const struct pcap\_pkthdr\*, const u\_char\*, const struct ether\_header\* ,const struct Arphdr\*) | Works. |
| ANS-16 (Binary) [UC-1.3] | To test that binary Pcmon can output correct packet info. | System be able to receive 802.11 packets smoothly. | Class Name :  Pcmon.c  Method Name :  void pcapHandler(u\_char\* ,const struct pcap\_pkthdr\*, const u\_char\*)  void printAddress(const u\_char\*)  void printHexAsciiValueOfPayload(const u\_char\*)  void handleFrames(const u\_char\*, int )  void handleControlFrames(const u\_char\*) | Weird output, figured out contained radiotap header, analyze in wireshark.  Solved. |
| ANS-17 (Binary) [UC-1.3] | To test that Pcmon is reading in radiotap headers correctly. | System be able to interpret radiotap headers correctly, resulting in correct packet info. | Class Name :  Pcmon.c  Method Name :  printHexAsciiValueOfPayload(const u\_char\*)  void writeToPcapFile(char\*, char\*) | Weird Output, Figured out radiotap header length, and why different length.  Solved. |
| ANS-18 (Analyze) [UC-4.1.3] | To test that graph is generated correctly based on .pcap file. | System be able to read pcap file and generate graph smoothly to screen. | Class Name :  GraphActivity, AddressPair, GraphView  Method Name :  void onActivityResult(int, int, Intent )  void onDraw(Canvas)  float getMax()  float getMin()  float getAvg()  float[] calculateYCoord(float[], float[]) | Graph first generation does not show up. Subsequent graphs shows up.  Solved. |
| ANS-19 (Analyze) [UC-4] | To test that the choosing of files other than pcap files will not crash app. | System be able to not crash or any errors upon choosing non-pcap files. | Class Name :  GraphActivity  Method Name :  void onActivityResult(int, int, Intent) | Works. |
| ANS-20 (Analyze) [UC-4] | To test that pcap analysis can both read in Monitor and Promiscuous pcap files. | System be able to detect and use appropriate code based on file content. | Class Name :  GraphActivity, AddressPair  Method Name :  void onActivityResult(int, int, Intent) | Works. |

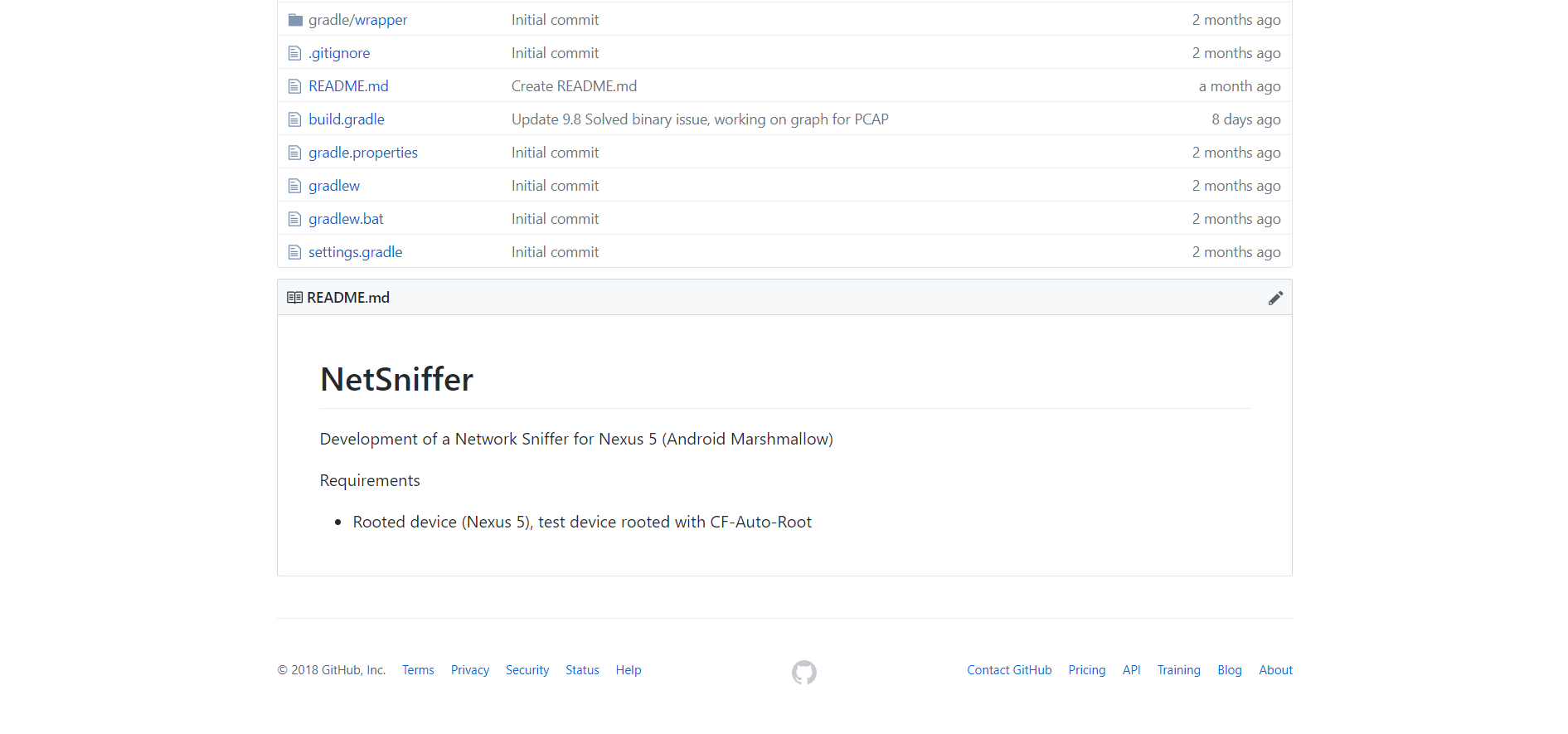
# 6. Evidence of using VCS (Version Control Software)

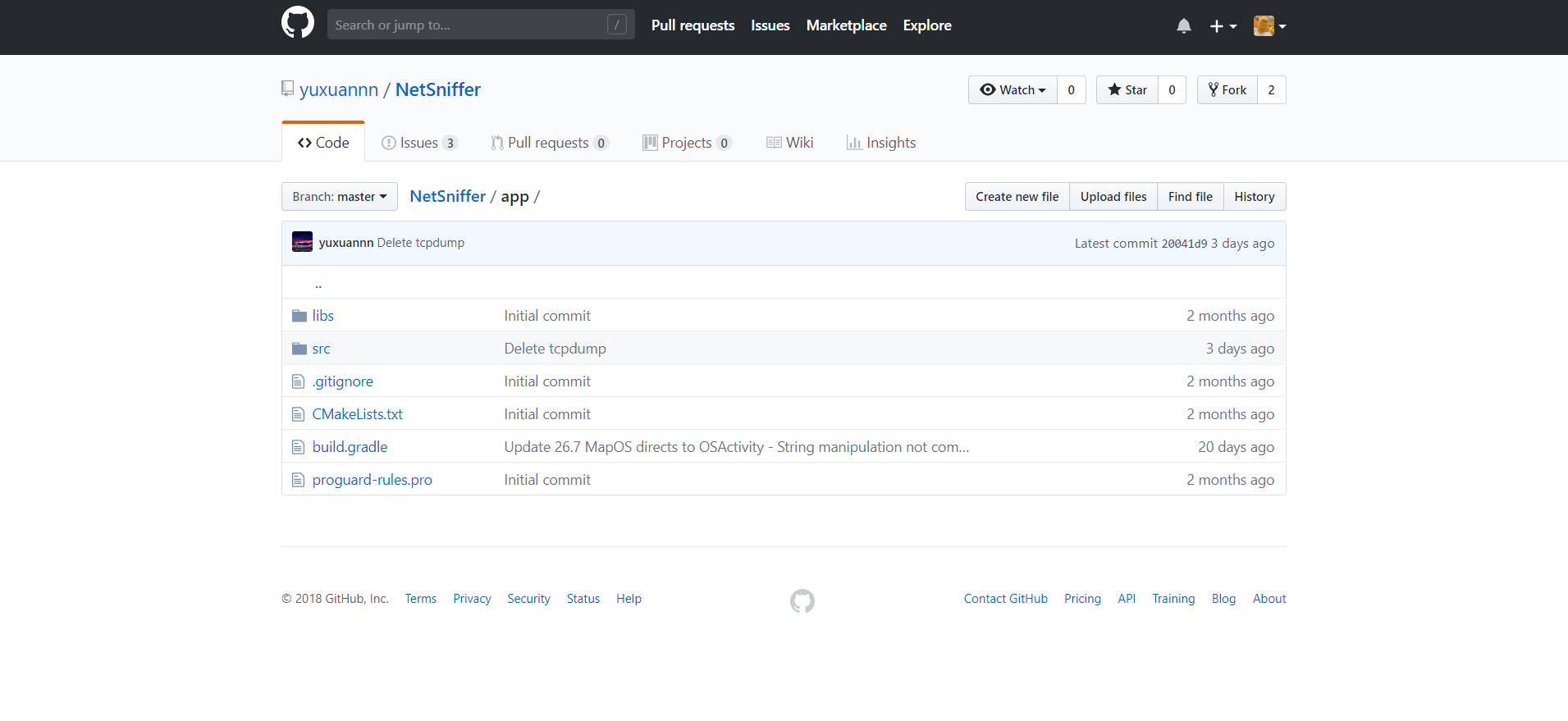
Version Control Software used : Github

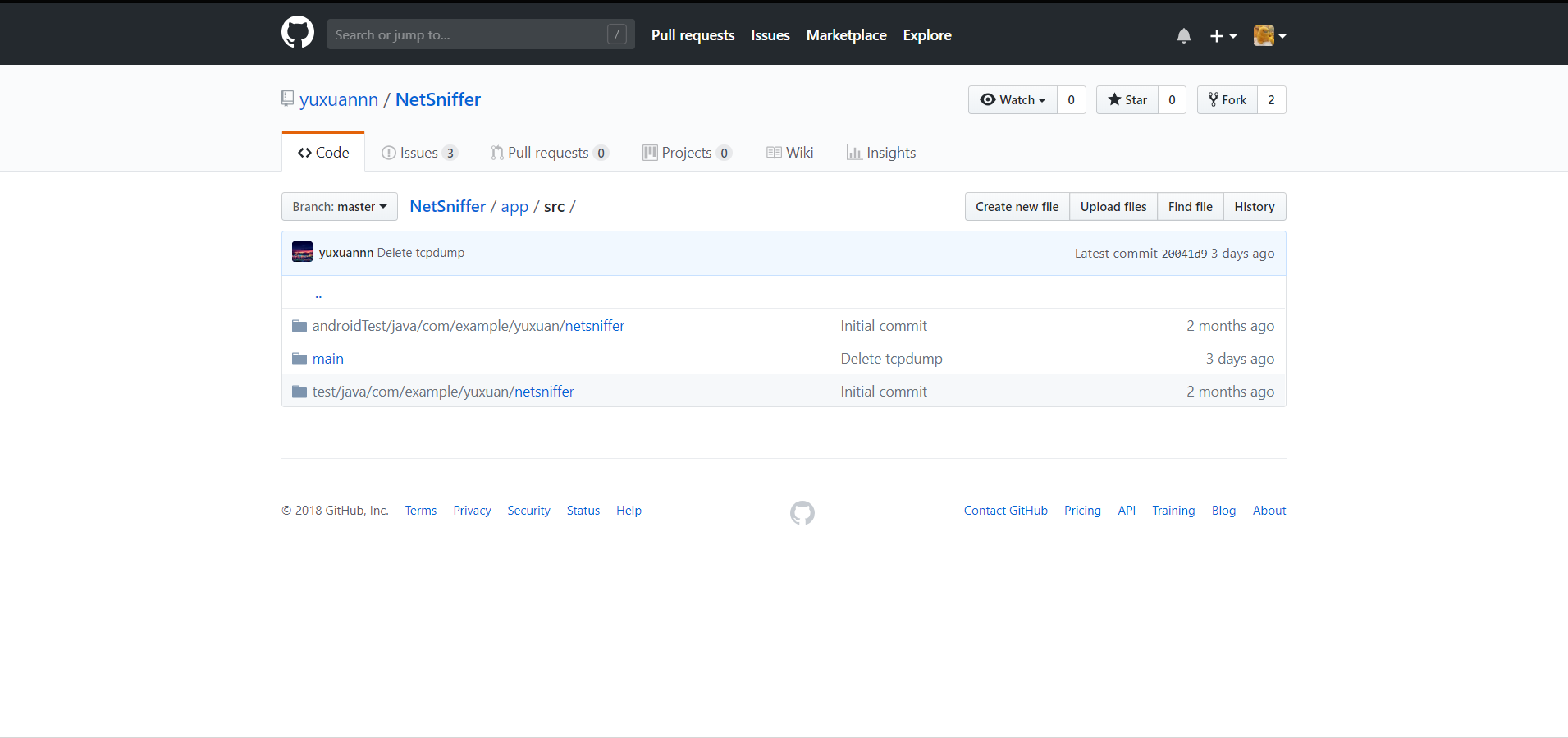
Screenshot #1 - Contents in the VCS's **Root Folder** containing all **Repository Project Files**

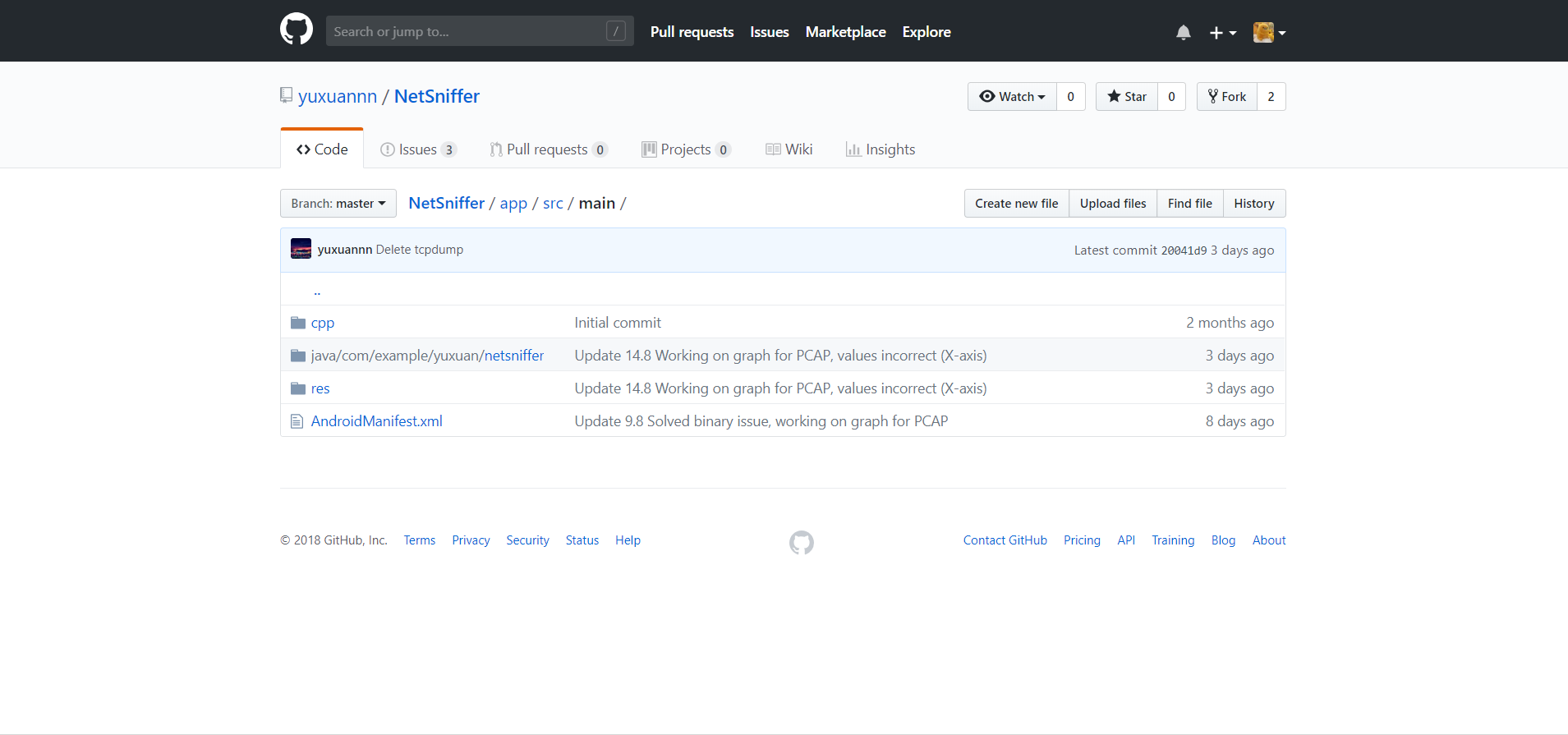
Netsniffer Repository:

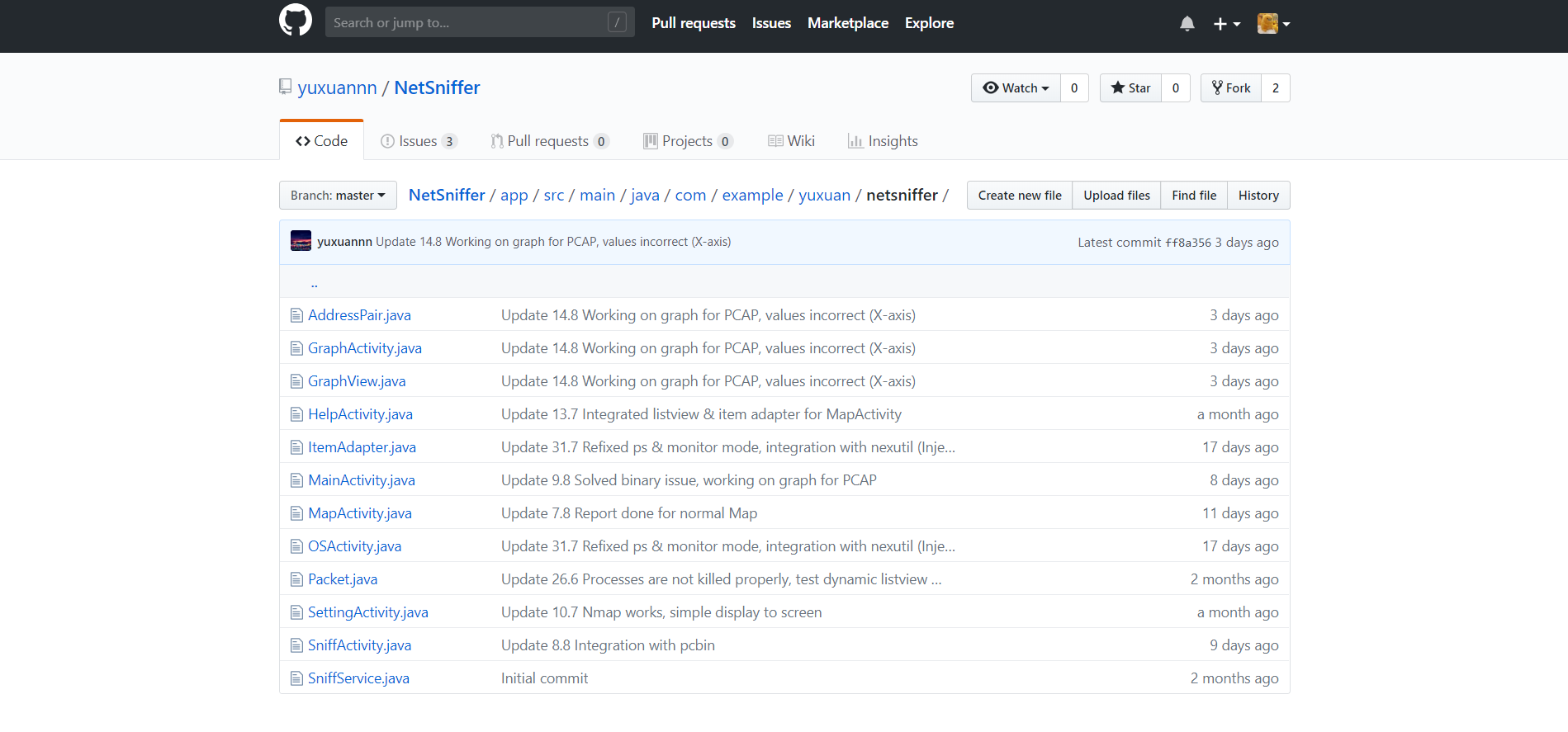




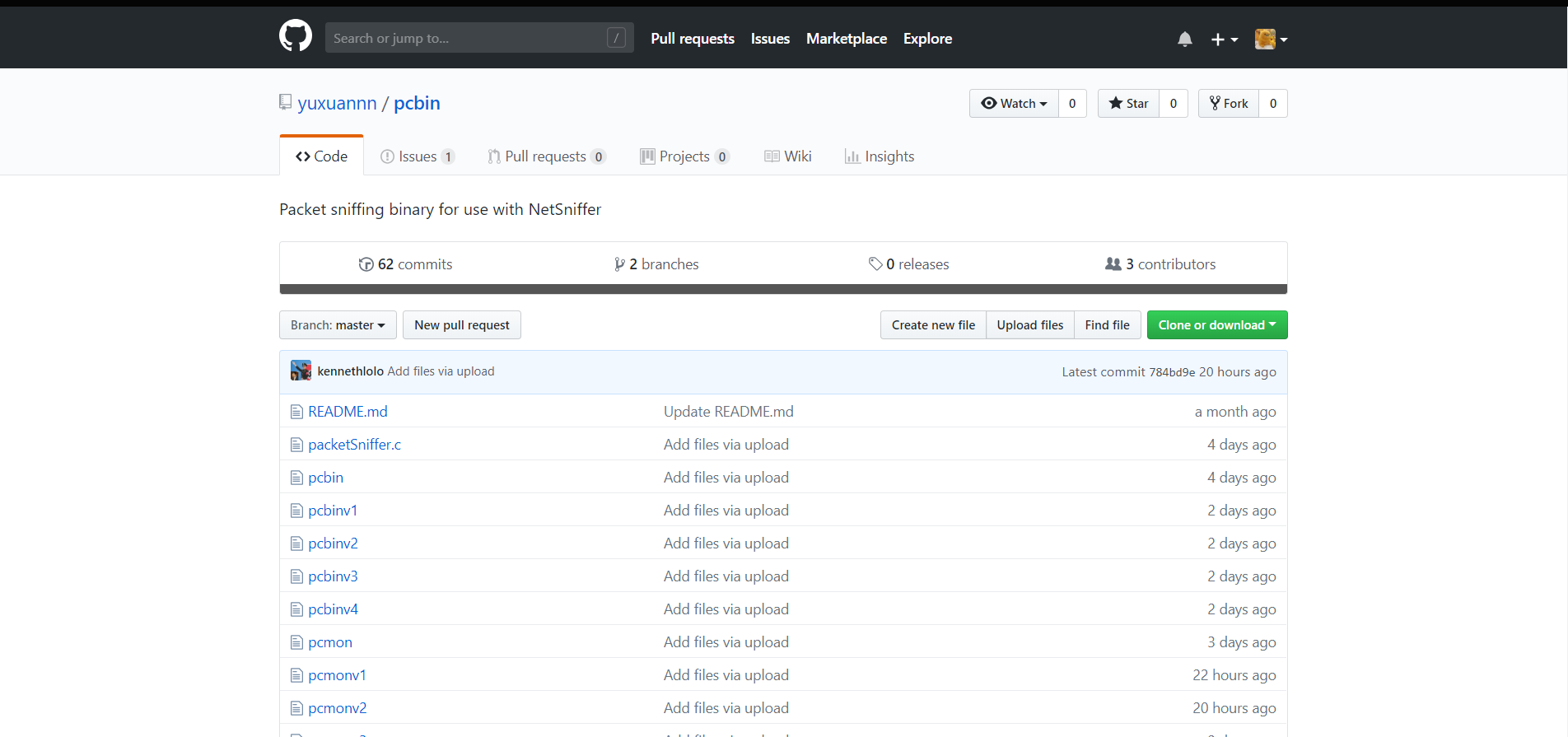




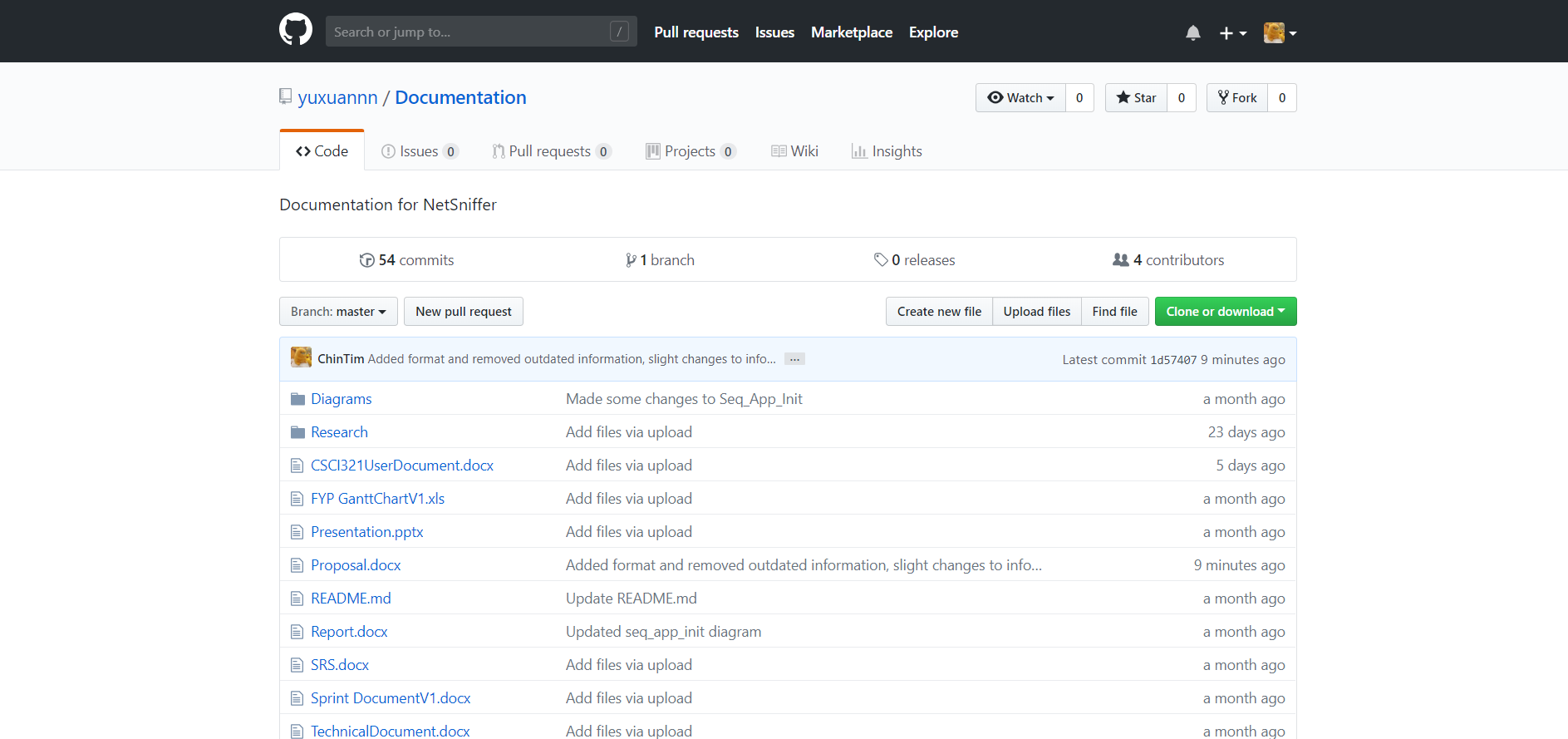




Pcbin Repository:



Documentation Repository:

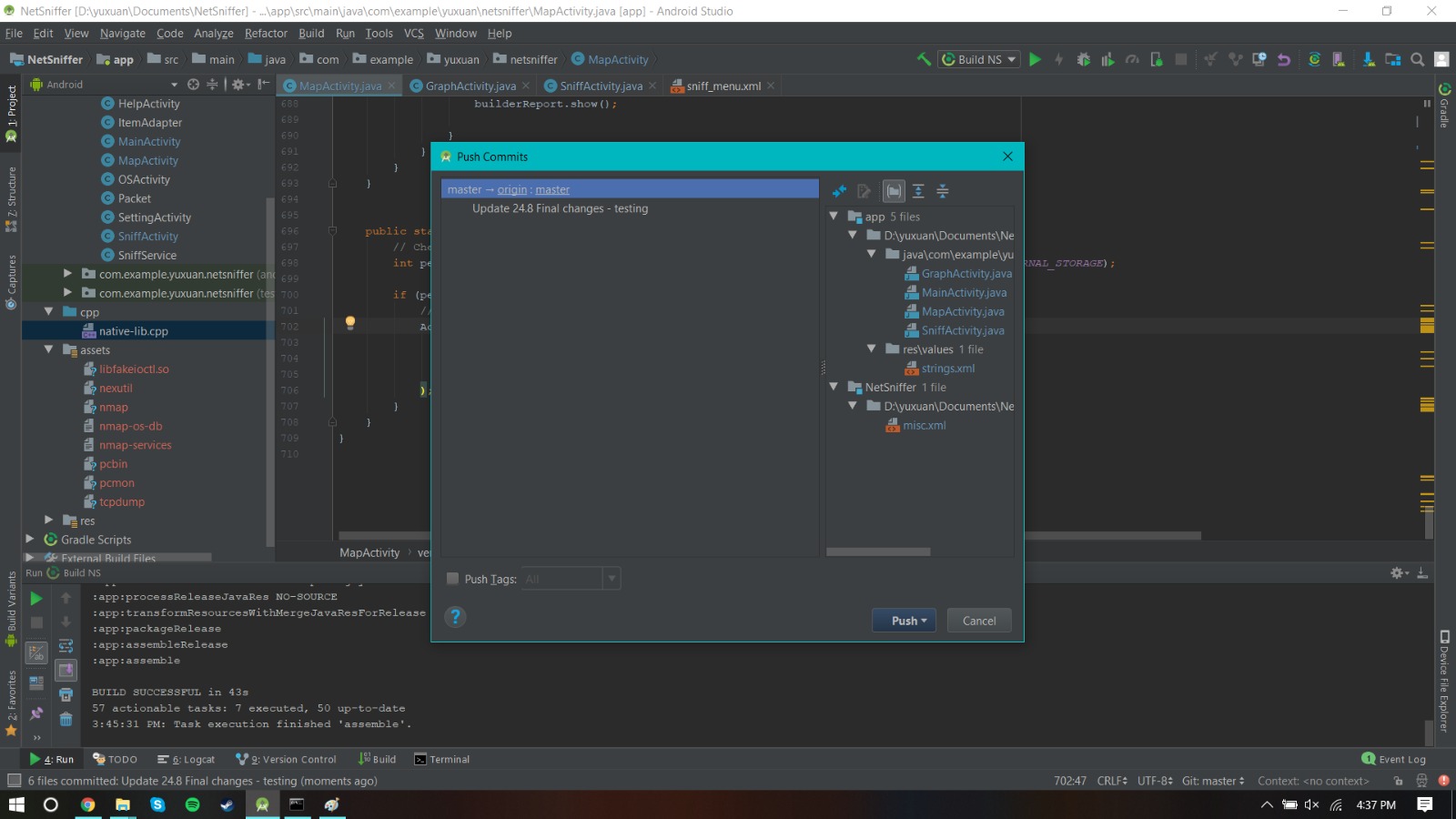
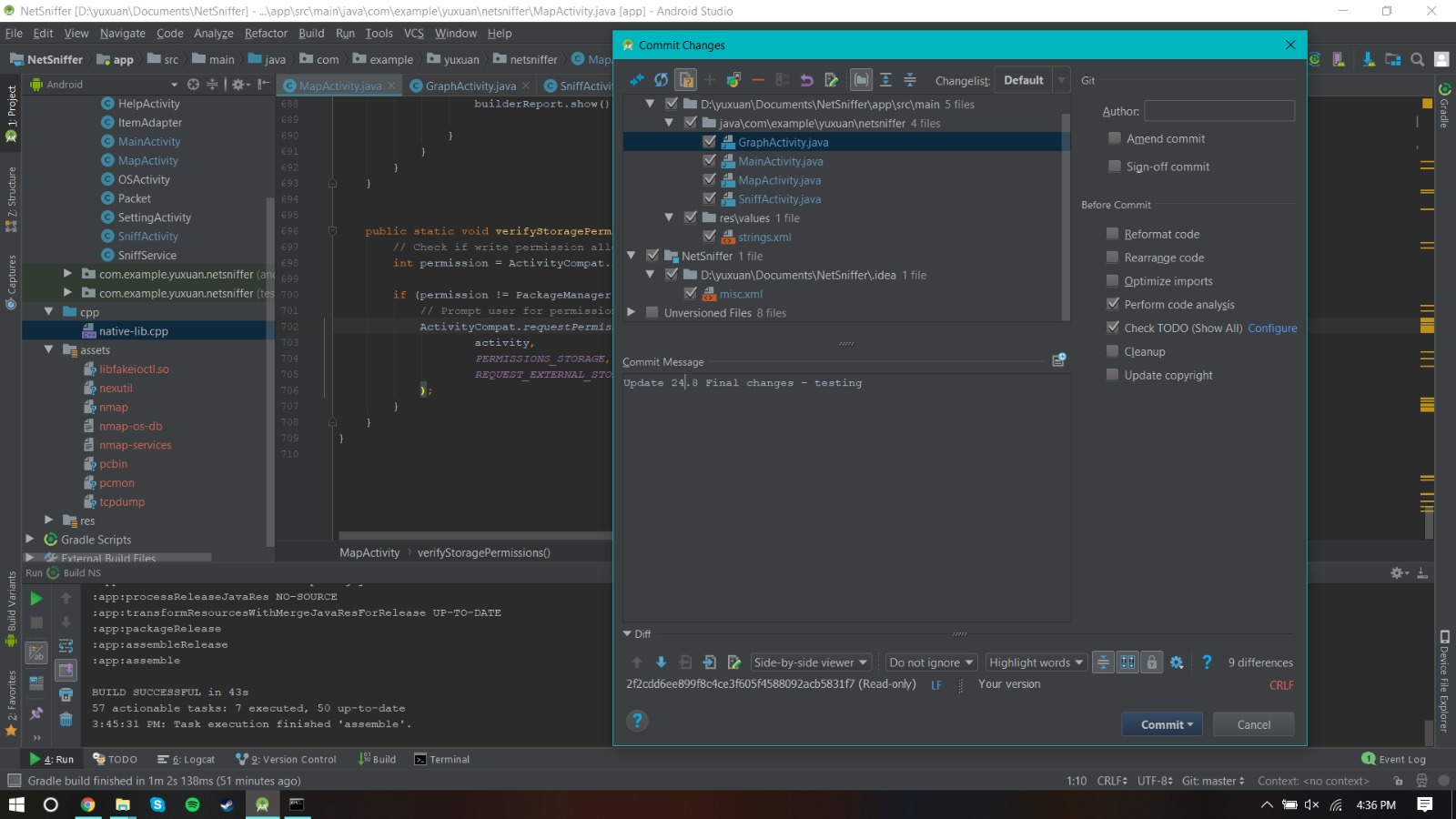


Screenshot #2 - VCS's listing of all the **latest source files currently being managed**

Screenshot #3 - Example using VCS's to **check-out source files** (it may be necessary to do >1 screen capture, depending on the software used)

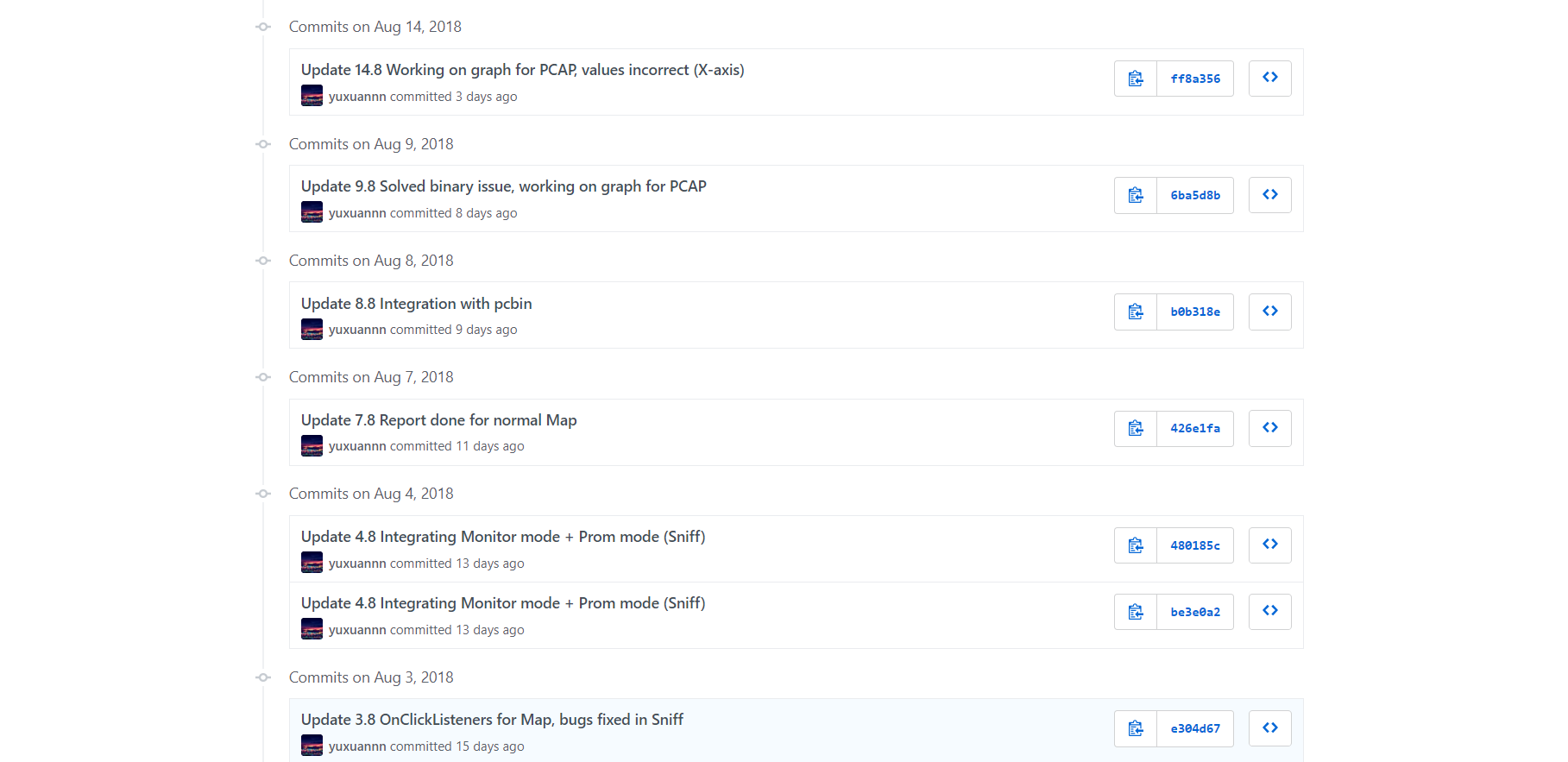
< Paste Screenshot #3 image**(s)** here >

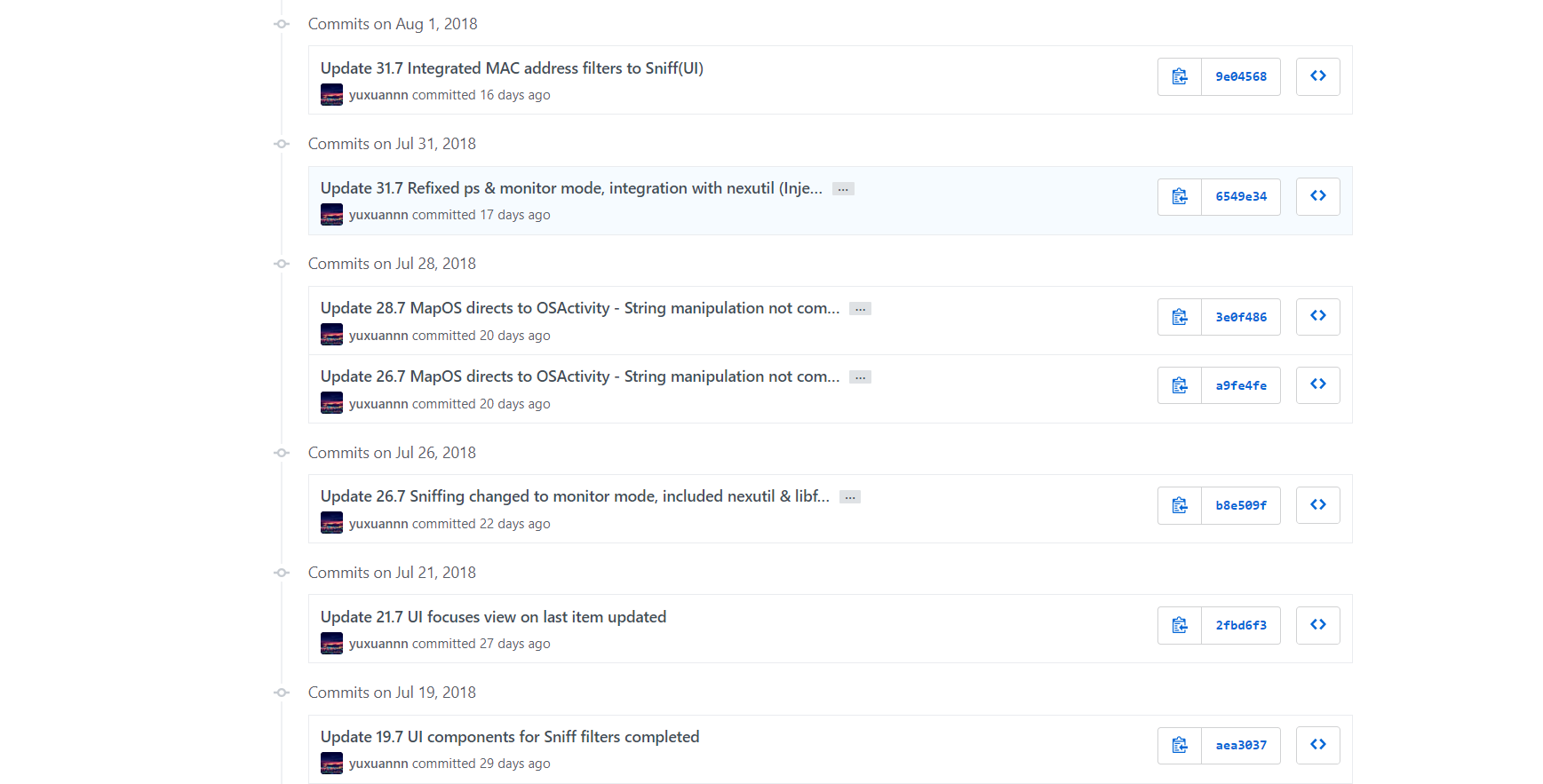
Screenshot #4 - Example using VCS's to **check-in source files** (it may be necessary to do >1 screen capture, depending on the software used)



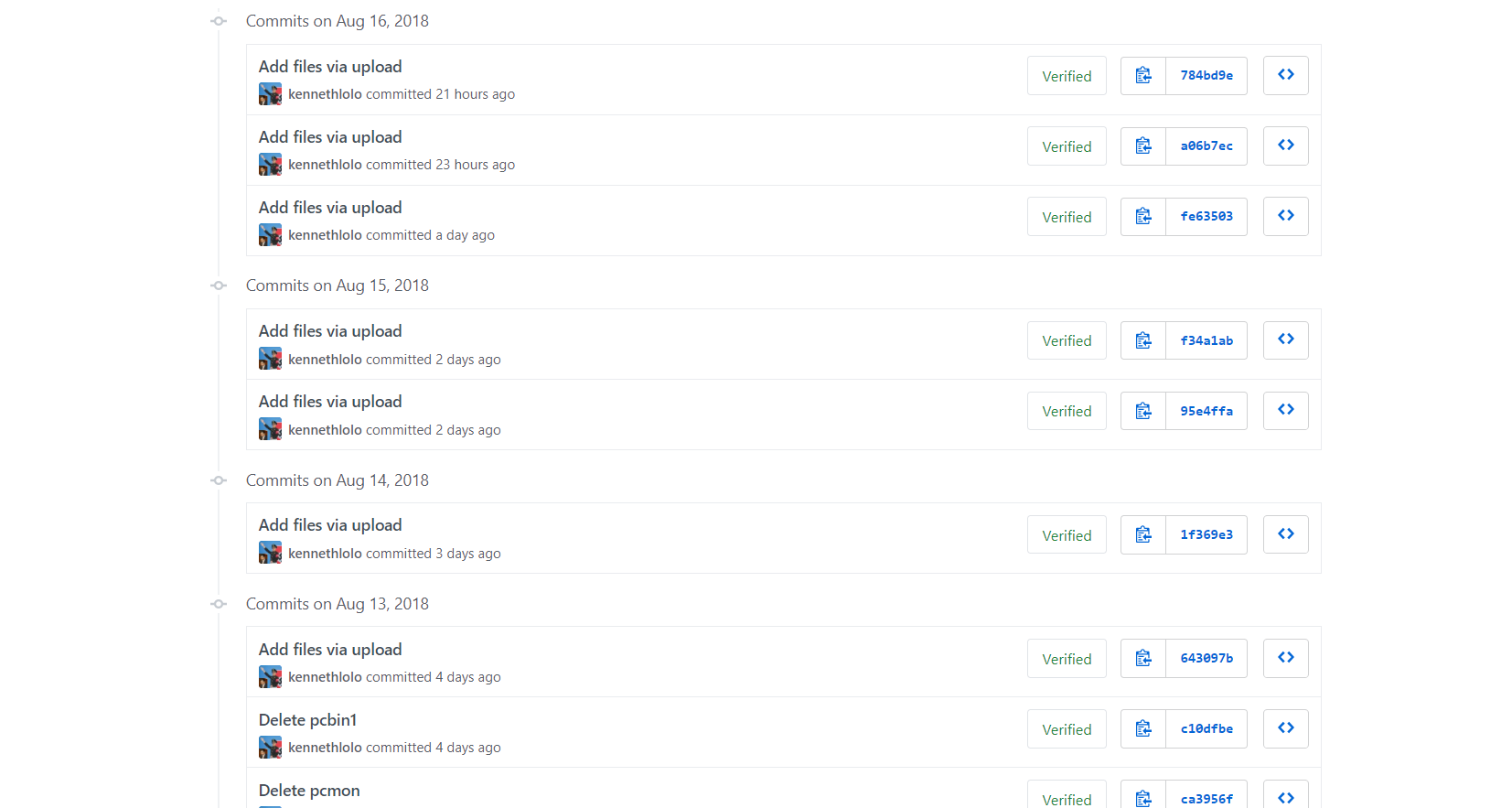
Screenshot #5 - Example using VCS's to **display the change history / log** (it may be necessary to do >1 screen capture, depending on the software used)

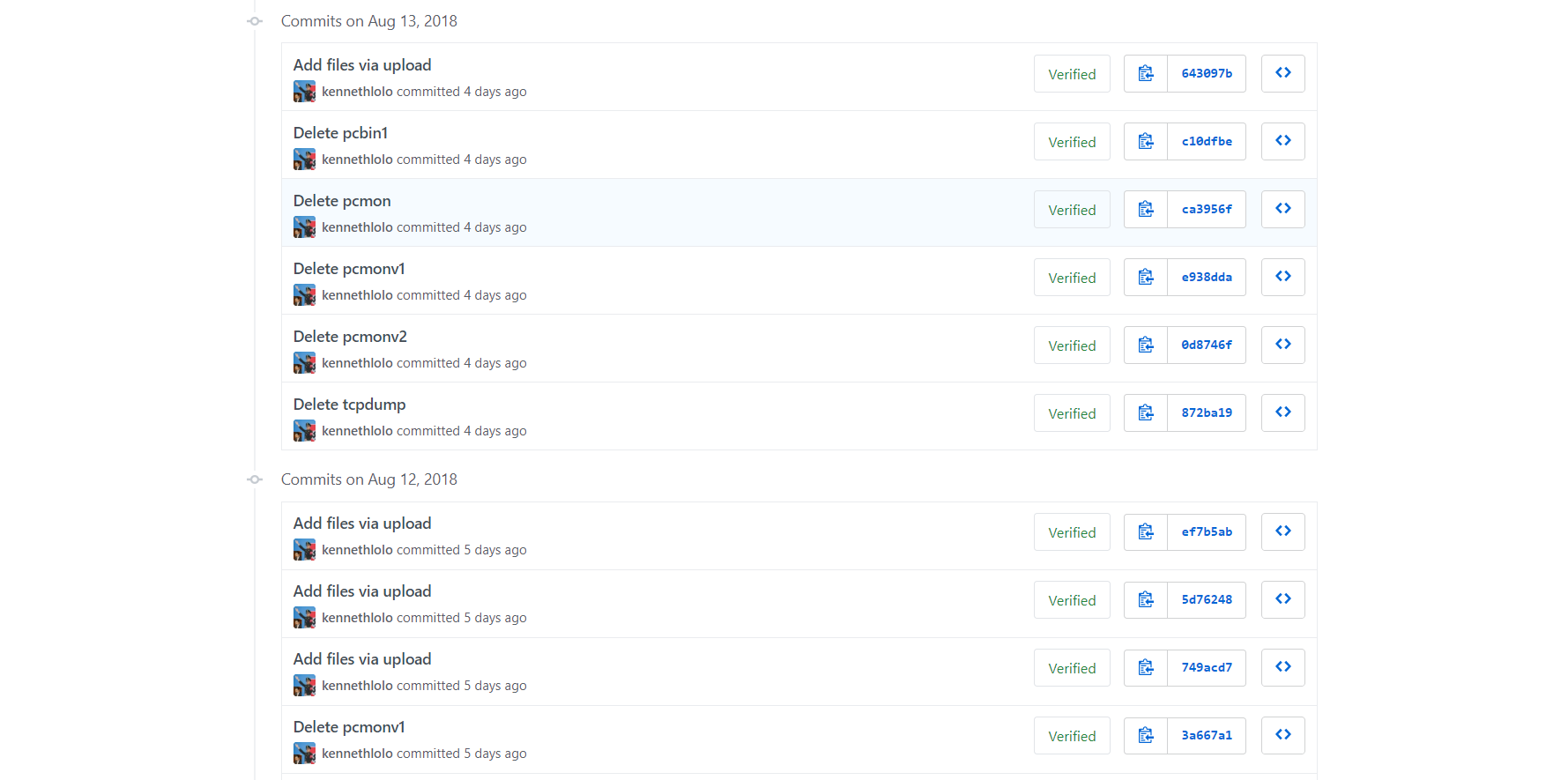
NetSniffer History Log(Commits):





Pcbin History Log(Commits):





Documents History Log(Commits):

