



ASSIGNMENT

Zawar khan



SEPTEMBER 13, 2024

XFLOWRESEARCH
Islamabad

Table of Contents

1. Objectives	3
2. Introduction	3
2.1. Brief Overview:.....	3
3. Approach/Methodology	3
3.1. Task 0:	3
3.1.1. Approach:.....	3
3.1.2. Results:.....	3
3.1.3. Challenges Faced:	4
3.1.4. Solution:	4
4. Task 0:.....	5
4.1.1. Approach:.....	5
4.1.2. Result:	5
4.1.3. Prerequisites:	6
4.1.4. Analysis:	6
5. Task 1 division.....	7
5.1. Sub Task 1: READ Task	7
5.1.1. Approach.....	7
5.1.2. Steps taken	7
5.1.3. Challenges.....	8
5.2. Sub Task 2: Filter Task	8
5.2.1. Approach.....	8
5.2.2. Steps taken	8
5.2.3. Challenges.....	8
5.2.4. Results:.....	9
5.3. Sub Task 3:.....	9
5.3.1. Approach.....	10
5.3.2. Challenges.....	10
5.4. Subtask 4	10
5.4.1. Steps taken.	10
5.4.2. Approach.....	10

6.	Task 3.....	11
6.1.1.	Approach.....	11
6.1.2.	Steps taken	11
6.1.3.	Challenges.....	12
7.	Task 4.....	12
7.1.1.	Approach.....	12
7.1.2.	Steps taken	12
8.	Last task	12
8.1.1.	Challenges.....	12
9.	Recommendations:.....	12
10.	References.....	13

1. OBJECTIVES

1. Read, filter, modify and write packets to and from pcap file using libpcap.
2. Use C++17 to code.
3. Perform unit tests.
4. Packaging the application with well-versed documentation.

2. INTRODUCTION

2.1. Brief Overview:

In this assignment we are going to use Libpcap with a proper C++ code to read, modify and write packets from and to .pcap file. Save to a pcapFile and create a database for extracted fields. Further, reasonable google framework unit tests are required for the written application.

3. APPROACH/METHODOLOGY

3.1. Task 0:

3.1.1. Approach:

Since I already have idea of git, github and git version control. This task will follow step outlined below for completion.

Tools: VS code, Browser for Github, Git for windows.

1. Create a private repository on Github.com account for the task with title xFlowResearch_Assignment.
2. As I have already set up environment and application of GIT installed in my desktop so I will not need this step.
3. In VS Code, in new work space I will enter clone the repo.
4. After cloning I will apply git fetch and git pull commands.
5. Will import .pcap file into the workspace as well.
6. Readme file will be edited with brief description of the project.
7. Commit the changes.
8. Task 0 completed.

3.1.2. Results:

1. Git hub link : https://github.com/ZawarKhan97/xFlowResearch_Assignment.git
2. I can also provide SSH link but that will require sharing the SSH key.

3. **Figure 1** shows the screenshot of the VS code setup git repository. Further validation is confirmed by checking the git remote URL version. Which is depicted in the **Figure 2**

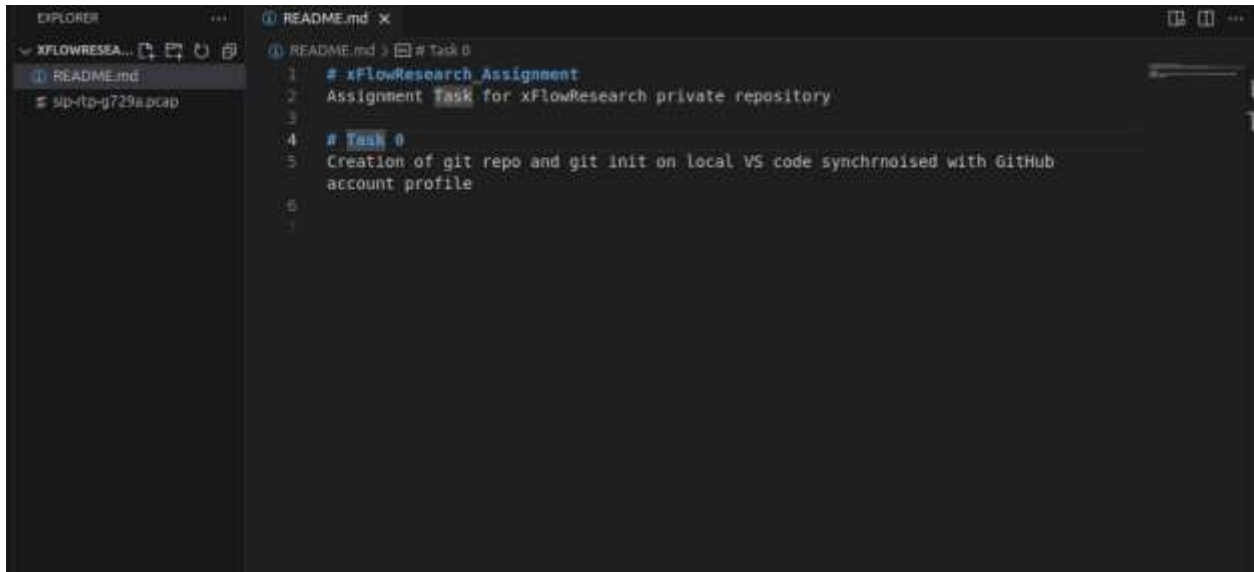


Figure 1: Read me file of git hub repository snapshot

```
zawar@Zawar:~/Zawar_Personal/xFlowResearch_Assignment$ git remote -v
origin  https://github.com/Zawarkhan97/xFlowResearch_Assignment.git (fetch)
origin  https://github.com/Zawarkhan97/xFlowResearch_Assignment.git (push)
zawar@Zawar:~/Zawar_Personal/xFlowResearch_Assignment$
```

Figure 2: Git remote version for validation of git hub repository

3.1.3. Challenges Faced:

1. Since it was private repository, I faced sign in issue.

3.1.4. Solution:

1. I updated the github account password since it was exposed in recent data breach. And then added the credentials.

4.1.1. Approach:

First Installing **Wireshark** on Ubuntu to observe the packets structure.

Tools: Wireshark

4.1.2. Result:

Figure 3 and Figure 4 shows the packet details in Wireshark. The analysis is down below.

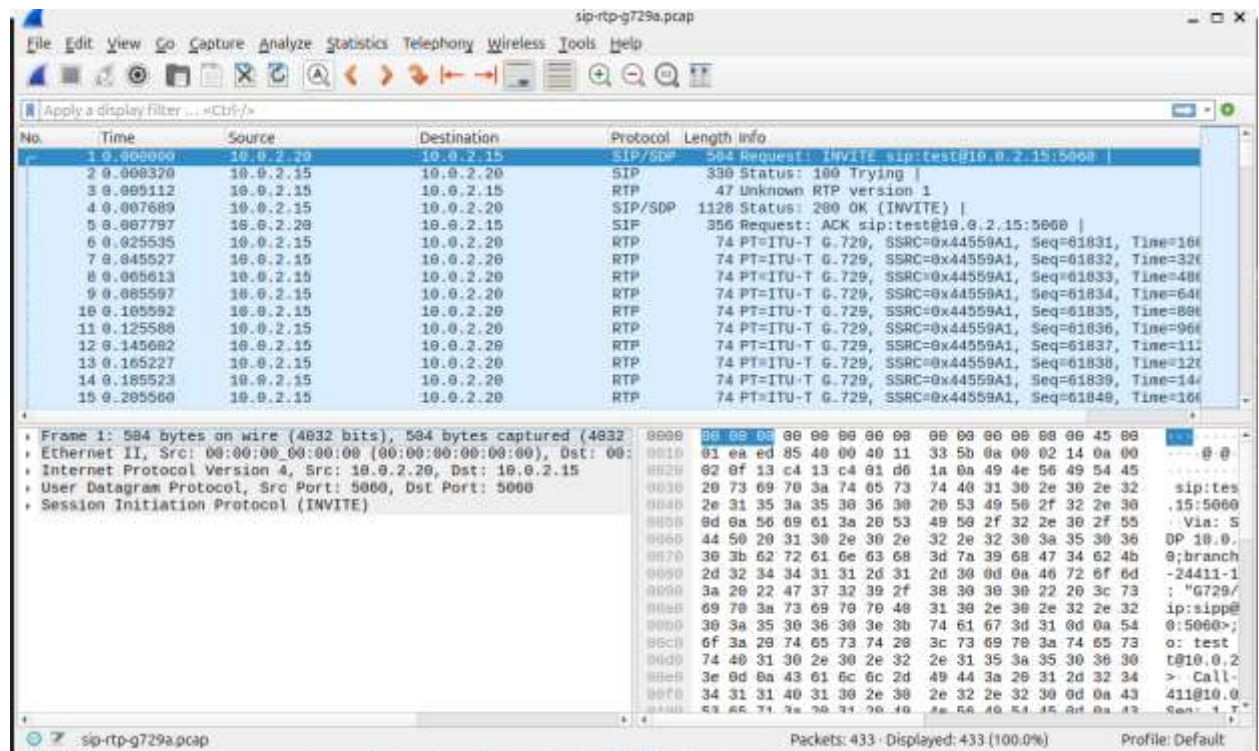


Figure 3: SIP Packet

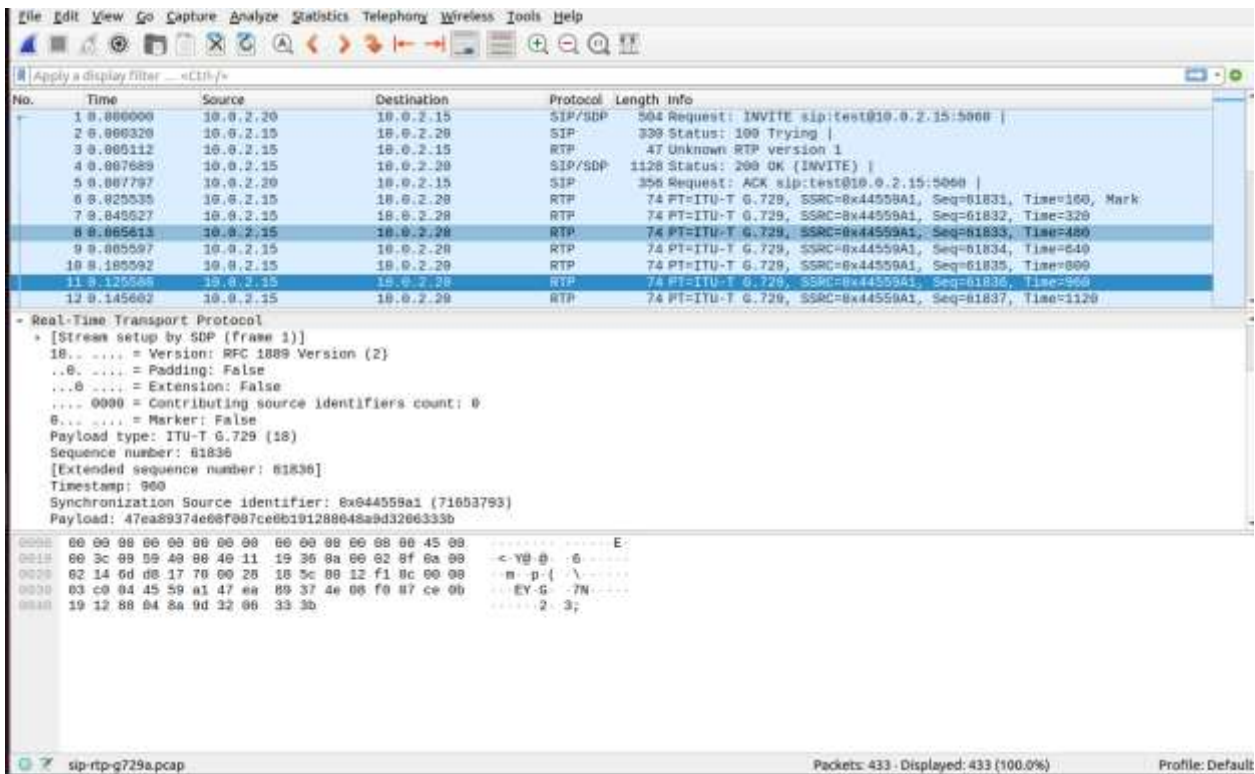


Figure 4: RTP Packet

4.1.3. Prerequisites:

Review of the following concepts to understand the packet structure and get a rough idea of basic functionality.

SDP Protocol: Session Description Protocol for session announcement and informing multicast participants to join [1]. Generally conveys the format of the session.

SIP Protocol: A session initiation Protocol is signaling protocol used for call communication as well creating multimedia session. It relies on other protocols such as RTP for data transmission [2].

RTP Protocol: A real time transport protocol that works on top of UDP protocol. It serves to enable video and audio transmission with interactive or real time features. [3]

G729: A voice audio compression technique that allows 8kbps/sec compression of the audio packet [4].

4.1.4. Analysis:

As per Wireshark analysis and review of the relevant protocols, it is observed that A session is initiated by host (source IP: 10.0.2.20) to client (Destination IP: 10.0.2.15) to request the initiation of SIP call with caller ID : 1-24411 on UDP port 5060, with G729 voice encoding scheme with 8kbps compression and RTP media protocol method. Whereas the invite

request for the caller request has been acknowledged by the destination node (10.0.2.15). While the transmission of Audio packets are done using RTP protocol which can be observed in **Figure 4**. Additionally RTP packets include RTP packets sequence number for proper encoding and time stamps for real time audio.

The call has been ended with SIP bye packet.

5. TASK 1 DIVISION

5.1.Sub Task 1: READ Task

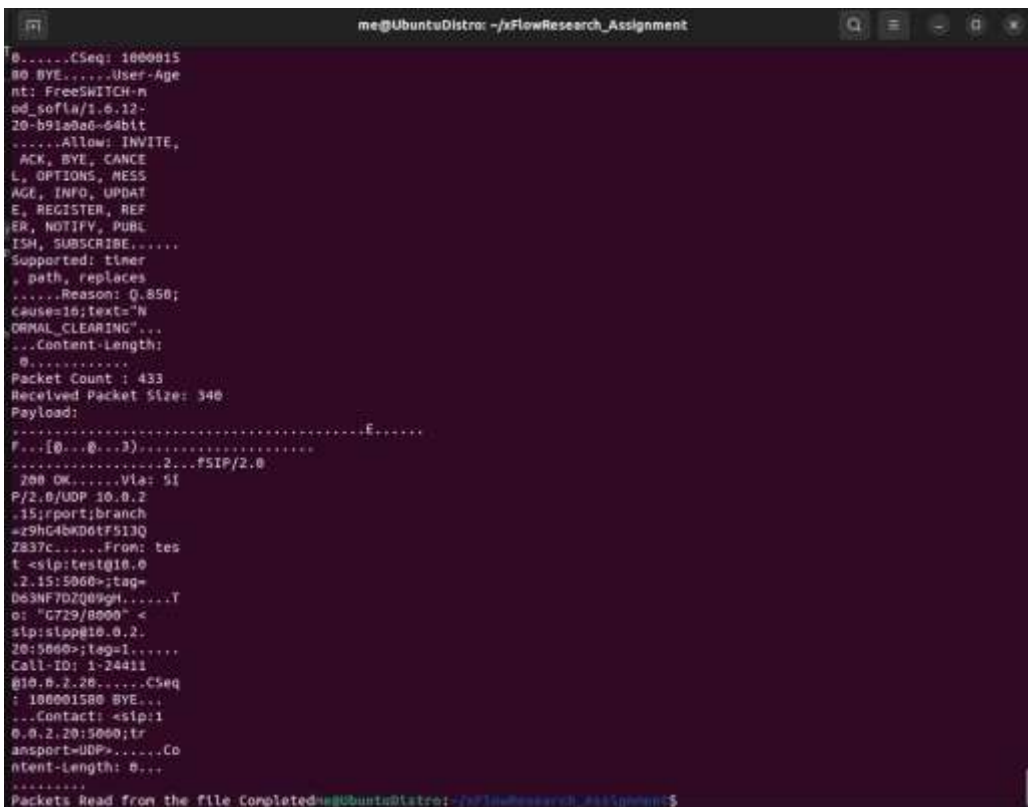
1. Read all the packets from the pcap file and extract the SIP portion of the packet.

5.1.1. Approach

1. Search and read about LibPCAP on TCPDump website [5].
2. See an example and test it.

5.1.2. Steps taken

1. Git clone the libpcap repository to my workspace, built the package.
2. Created a new .cpp source file.
3. Creating a simple sniffer application using function read from manual pages of TCPCDUMP MAN documentation.



```
me@UbuntuDistro: ~/FlowResearch_Assignment
0.....CSeq: 1000015
00 BYE.....User-Agent: FreeSWITCH-n
od_sofia/1.0.12-
20-b91a0a0-64bit
.....Allow: INVITE,
ACK, BYE, CANCEL,
OPTIONS, MESSAGE,
INFO, UPDATE,
REGISTER, REFER,
NOTIFY, PUBLISH,
SUBSCRIBE.....
Supported: timer
path, replaces
.....Reason: 0.850;
cause=10;text="N
ORMAL CLEARING"....
...Content-Length:
0.....
Packet Count: 433
Received Packet Size: 340
Payload:
.....E.....
F...[0...0...2).....
.....2...fSIP/2.0
200 OK.....Via: S/I
P/2.0/UDP 10.0.2
.15;port;branch
=z9hG4bK0d0f513Q
Z837c.....From: tes
t <sip:test@10.0
.2.15:5060>;tag=
063MF70Z009GH.....T
o: "C729/8000" <
sip:slop@10.0.2.
20:5060>;tag=1.....
Call-ID: 1-24411
@10.0.2.20.....CSeq
: 100001500 BYE...
...Contact: <sip:1
0.0.2.20:5060>;tr
ansport=UDP>.....Co
ntent-Length: 0...
.....
Packets Read from the file Completedme@UbuntuDistro: ~/FlowResearch_Assignment$
```

Figure 5: Packet read using Cpp application

4. Successfully built the application and coded a Make file for automatic build.
5. The application was tested which gave stats of all the packets inside pcap file as analyzed by Wireshark. The results are shown in **Figure 5**.

5.1.3. Challenges

1. Some dependency were not found while configuring the libpcap, so they were search and downloaded to apt package list.
2. Git history of the lipcap was also found in order to remove it I had to find how to remove the git history
3. Make file generated some syntax issues. I used chat gpt to fix the script.

5.2.Sub Task 2: Filter Task

1. Extract the sip portion of the packet from the data.
2. Extract fields TO, FROM, CALL ID
3. Save these fields data in text file as well

5.2.1. Approach

1. Search and read the documentation of pcap_compile and pcap_filter function in TCPDump manual Pages
2. Review of concepts of file handling for storing the packet filtered data.
3. Implement the functions for extracting SIP packets.

5.2.2. Steps taken

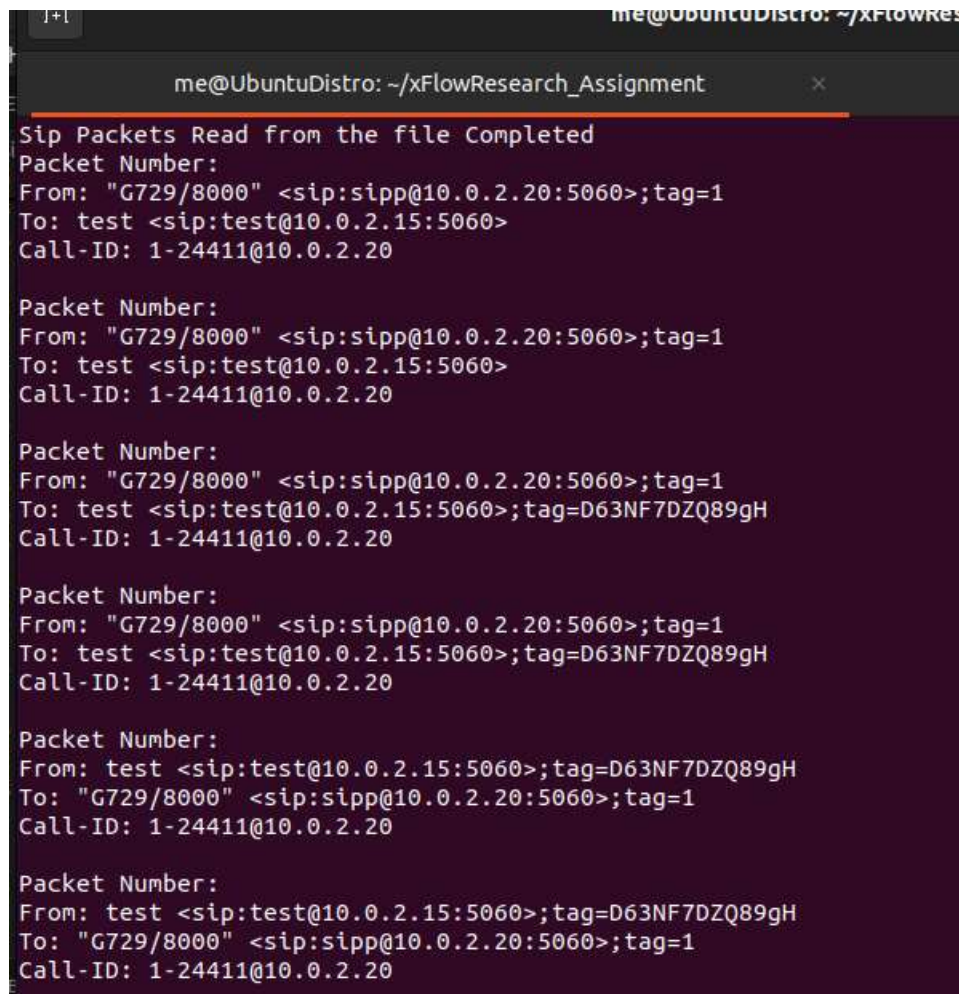
1. Wrote pcap_compile function and pcap_filter function to filter SIP packets
2. Use pcap_loop to read packets
 - a. TCMPDump magazine about sniffer(Research)
3. Used call back function to parse the packet fields
 - a. Learned from stack exchange to parse or split strings [6]
4. Save the packets fields
 - a. Saved it using vectors.

5.2.3. Challenges

1. During pcap-compile function, network IP posed a problem since I was using the file. I used Wireshark to determine the IP and then use default gateway IP for the net variable and used Apra library to convert to net format.
2. Faced another challenge of filtering expression that SIP portion was not parsed successfully. Tried UDP string but did not work. Then I read the pcap_filter MAN page to know if I am using incorrect expression. Learned I was using wrong syntax. Udp worked though. Sip failed. I read the assignment it said fixed header length. So

- instead of using Offset and writing a function to parse the payload for “SIP “Keyword. I used udp port 5060 as determined in Wireshark. And as expected 6 packets of sip were counted and extracted.
3. Challenged faced in correctly parsing the payload. Resolved it using Wireshark analysis of the fixed size header lengths.
 4. Huge number of bugs results from definition and declaration of functions. Priority was found in the function definitions

5.2.4. Results:

A terminal window titled 'me@UbuntuDistro: ~/xFlowResearch_Assignment' displays the output of a script. The script has completed reading SIP packets from a file. It lists six packets, each with its packet number and SIP header details. The first four packets are from 'G729/8000' to 'test', and the last two are from 'test' to 'G729/8000'. All packets have a Call-ID of '1-24411@10.0.2.20'.

```
me@UbuntuDistro: ~/xFlowResearch_Assignment
Sip Packets Read from the file Completed
Packet Number:
From: "G729/8000" <sip:sipp@10.0.2.20:5060>;tag=1
To: test <sip:test@10.0.2.15:5060>
Call-ID: 1-24411@10.0.2.20

Packet Number:
From: "G729/8000" <sip:sipp@10.0.2.20:5060>;tag=1
To: test <sip:test@10.0.2.15:5060>
Call-ID: 1-24411@10.0.2.20

Packet Number:
From: "G729/8000" <sip:sipp@10.0.2.20:5060>;tag=1
To: test <sip:test@10.0.2.15:5060>;tag=D63NF7DZQ89gH
Call-ID: 1-24411@10.0.2.20

Packet Number:
From: "G729/8000" <sip:sipp@10.0.2.20:5060>;tag=1
To: test <sip:test@10.0.2.15:5060>;tag=D63NF7DZQ89gH
Call-ID: 1-24411@10.0.2.20

Packet Number:
From: test <sip:test@10.0.2.15:5060>;tag=D63NF7DZQ89gH
To: "G729/8000" <sip:sipp@10.0.2.20:5060>;tag=1
Call-ID: 1-24411@10.0.2.20

Packet Number:
From: test <sip:test@10.0.2.15:5060>;tag=D63NF7DZQ89gH
To: "G729/8000" <sip:sipp@10.0.2.20:5060>;tag=1
Call-ID: 1-24411@10.0.2.20
```

Figure 6: Filter Packets of SIP

5.3.Sub Task 3:

1. save the content into pcap file
2. Reconstruct the packet by adding back the modified parts.

3. Save the packet to pcap file.

5.3.1. Approach

1. Read manual pages on API to dump a pcap file

5.3.2. Challenges

1. Segmentation fault error: upon searching I found I need to allocate memory before memcpy for error buffer, otherwise it cannot decide its memory.
2. Found that no data is being written to pcap file. Found out that I was passing value to pcap_loop rather than pointer or address.

5.4.Subtask 4

1. Reading from Wireshark the modified Pcap file
2. Find and correct the error.

5.4.1. Steps taken.

1. Read the packet from Wireshark, it correctly showed my packets, they are shown in Figure 8. Also I did not need to correct the length of the packet or checksum.

5.4.2. Approach

Check checksum and frame length for the difference in size of lengths. As shown in Figure 7. Analysis it found that it is missing the spaces. Corrected it and it worked. Figure 8.

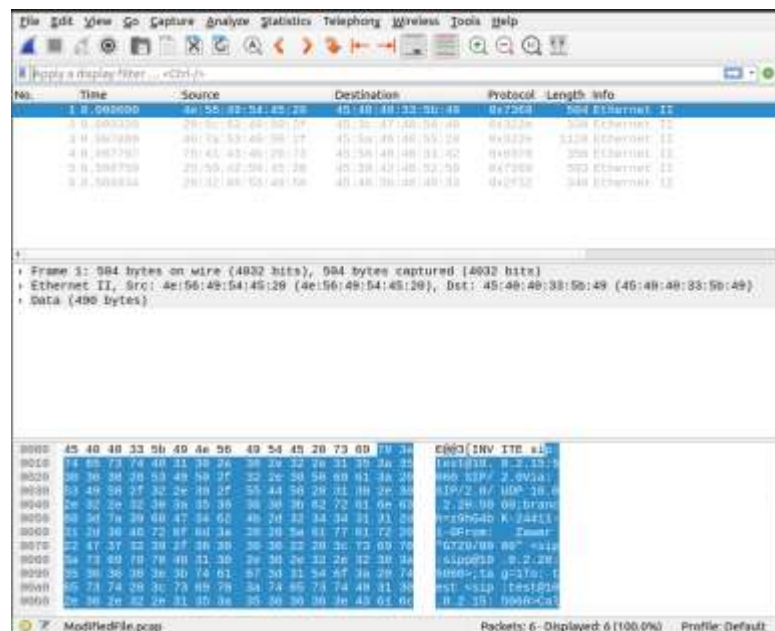


Figure 7: Packets not read completely by the Wireshark

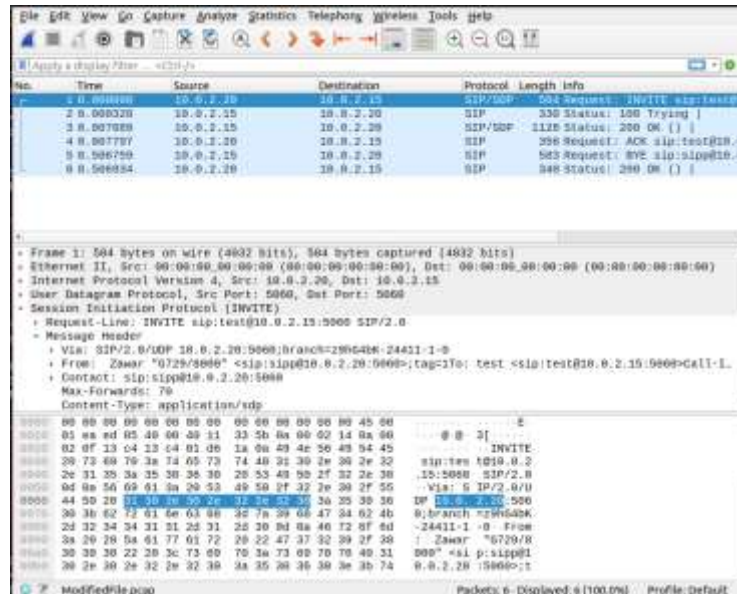


Figure 8: Packets written correctly and determined by Wireshark

6. TASK 3

Writing some database manipulation code

1. You need to write the 'To', 'From', and 'Call ID' fields extracted in the first part to a database.
2. You can use any database you like and can design the database however you like.

6.1.1. Approach

1. Learn how to install SQLite 3 in Linux and create a database
2. <https://www.prisma.io/dataguide/sqlite/setting-up-a-local-sqlite-database#setting-up-sqlite-on-linux>
3. Watch tutorial how to insert, read and modify to a database using cpp.
4. <https://www.youtube.com/watch?v=IBtRhG9ycY&t=12s>
5. Create your own database and write a cpp code to manipulate it to add my entries.
6. https://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm

6.1.2. Steps taken

1. Installed Libraries for SQLite in Linux so that I can use sqlite3.h and its APIs. Used SQLite library Documentation for functions available and found functions of my use.

6.1.3. Challenges

It was not easy since I had no prior knowledge of database. Spent lot of time in tutorials.

7. TASK 4

1. Testing the code through unit testing.
2. You need to use the google test framework to write at least three unit tests for your code.
3. One required unit test is for the database

7.1.1. Approach

1. Understand Google Test Framework
2. Write Unit Test case or test Suite

7.1.2. Steps taken

1. Read google test framework <https://google.github.io/googletest/primer.html>
2. `sudo apt install libgtest-dev`
3. Installed the package
4. Link the package with cpp source code

8. LAST TASK

1. Code refinement
2. Classes and organizing into structure
3. Proper exceptions error dealings
4. Modular approach
5. Challenges static and non-static members
6. Some function needs to be defined in order so that they are found
7. Struct must be defined outside class

8.1.1. Challenges

I faced a lot of challenges while code refinement it may have taken several hours. I structured the code in OOPs concept. Found that linking was not easy, members cannot be static, and member's manipulation with object is very difficult. Knew some concept of self-instances from python but was not able to find and mirror same approach in cpp. More can be told in interview.

9. RECOMMENDATIONS:

1. Please do also show that from field also start from test string at the end of series packet.
2. It will be good if you could explain the tests task.
3. I will appreciate if data base task could be explained more as well.

4. References:

10. REFERENCES

- [1] M. Handley and V. Jacobson, “SDP: Session Description Protocol,” *Ietf.org*, 2024. <https://www.ietf.org/rfc/rfc2327.txt> (accessed Sep. 13, 2024).
- [2] B. S. KU, “CHAPTER 12 - Internet Telephony Technology and Standards Overview,” in *In Communications, Networking and Multimedia, Multimedia Communications*, Academic Press, 2001, pp. 191–219. Available: <https://www.sciencedirect.com/science/article/pii/B978012282160850013X>
- [3] H. Schulzrinne, S. Casner, R. Frederick, and v. Jacobson, “rfc3550,” *datatracker.ietf.org*, Jul. 2003. <https://datatracker.ietf.org/doc/html/rfc3550>
- [4] Sangoma Technologies, “G729 codec ★ Asterisk,” *Asterisk*, May 14, 2024. <https://www.asterisk.org/products/add-ons/g729-codec/> (accessed Sep. 13, 2024).
- [5] L. M. Garcia, “VOIP Storming IP Security,” *Practical Protection Hard core IT Security Magazine*, vol. 3, no. 1733–7886, pp. 38–46, 2008. Available: <http://recursos.aldeabaknocking.com/libpcapHakin9LuisMartinGarcia.pdf>
- [6] StackExchange, “parsing - Parse (split) a string in C++ using string delimiter (standard C++),” *Stack Overflow*. <https://stackoverflow.com/questions/14265581/parse-split-a-string-in-c-using-string-delimiter-standard-c>