Name: Prakram Rathore
Roll No: 20110141

DESIGN DOCUMENT

How to Run code:

1. Open two terminals in client and server folders.
2. Run python3 client.py and python3 server.py
3. Select encoding style from 0,1,2
4. Type commands in order
    a. Cwd [gives current working dir of server]
    b. Ls [list all content in server]
    c. Cd sys [move in folder name sys]
    d. Cd cont [move in folder name cont]
    e. Upd client.txt [upload client.txt from client to server cont dir]
    f. Dwd server,txt [download server.txt from server's cont dir to client]

For creating the file system I have used three modes of layering:

1. **File Service:** This is the top most layer of the system. It is like an application layer. This file service performs five commands through which client and server share information with each other. These five commands perform tasks like fetching current working directory, changing directory, fetching list of content and doing upload and download operations. This layer relies on the layer below it

which is a crypto service for encryption and decryption of data which needs to be exchanged between client and server.

2. **Crypto Service:** The second layer provides the encryption and decryption of the data sent between client and server. Whenever a client uses any command or sends any file it will use this service to encrypt the data using one of the any three encryption formats.
    a. Plain Text: This is a very simple form of service in which there is no need for encryption. The file is sent as a string as if there is no change.
    b. Caesar cipher: Not exactly like a caesar cipher but every lower or upper case letter is replaced by a letter with an offset of 2. I am maintaining two lists of alphabets and whenever we encounter upper or lowercase letters we set that letter with a letter with index equal to index of letter +2. Also we put a %26 which ensures that y and z matches to a and b. Theirs is a decryptor function which reverses these offset by actually giving an offset of 24 which will make sure that letters are back to their original form. Whenever we send data either from client to server or server to client we encode using caesar cipher and when reading anything on client or server we use the decrypt function to decrypt it.
    c. Transpose/Reverse: In this encoding style we split every word of the data and reverse it individually. The string "My name is Prakram" will be encoded as "yM eman si markarP". When sending any data we call reverse_encoding function which reverses the string in a given format and while reading data we again call the same function because reverse or reverse will give the actual data.
3. **Networking service:** The bottom most layer is the networking layer. It is responsible for the protocols for file transmission. I am using the TCP protocol in this layer to set up a connection between client and server. I have designed the client and server system in such a way that if a server is started once then it does

not need to shutdown and we can again and again connect the client to it. Clients on the other hand can continuously give commands until it exits by giving a “q” command.

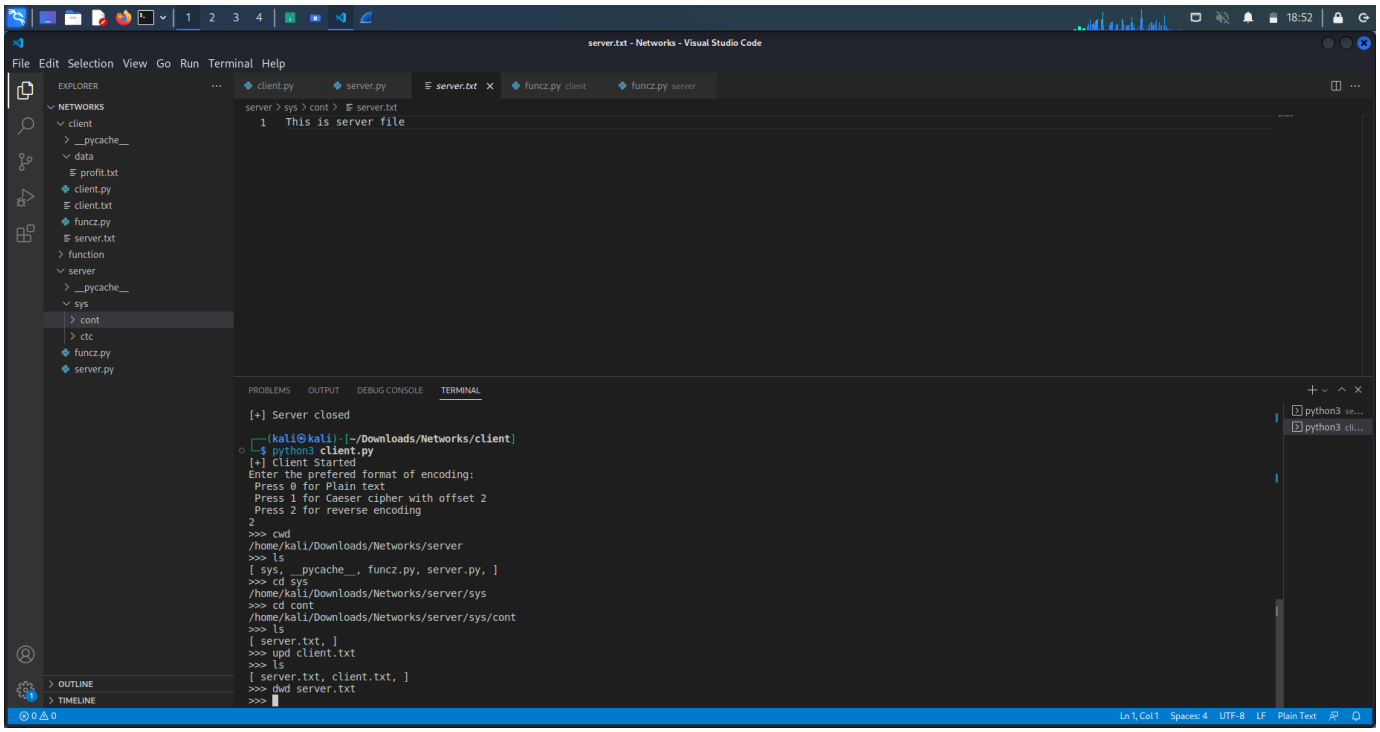
CHALLENGES FACED:

1. I tried implementing the file system using C language but was unable to do so because of the book which was not indexed properly. Compared to making the file system using python, C was a tougher choice as Python is more compatible and user friendly in doing such jobs.
2. Setting up the connection between client and server properly was a huge headache. When and how will the while loop end and how to make our server and client persistent so that the client does not close after giving one command. I tried making it look like an actual terminal.
3. To see the changes in file created while uploading and downloading commands, the connection between client and server needs to be closed in a proper fashion so that the changes in data of file created can be seen otherwise the data will not load into the file properly.
4. Error handling and debugging whenever there is an error was a difficult job to be done.
5. The encryption and decryption needed to be done properly. Writing code for it was a little bit difficult as every time we send or receive we need to take care of encryption and decryption and the encoding style that we are using in the starting of the system.
6. Downloading data from any directory of server and uploading data to any directory of the server was more challenging to achieve than simple downloading and uploading to server directory.

WIRESHARK SCREENSHOTS:

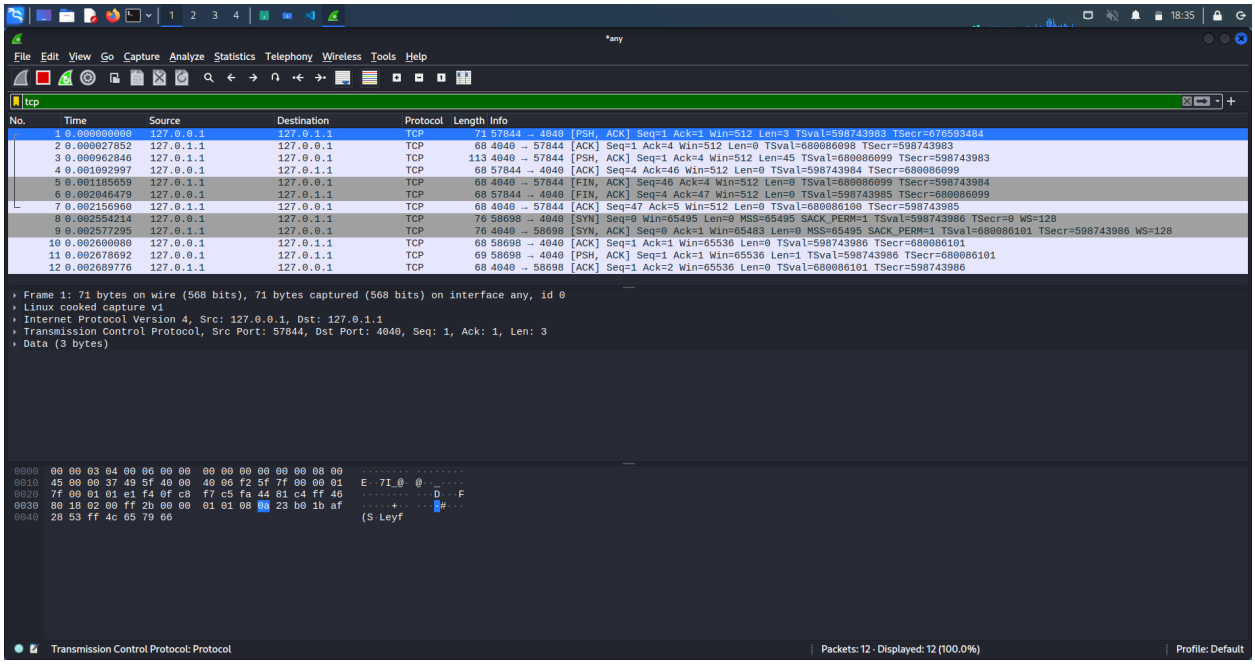
Encoding style: Caesar cipher

Note: Zoom in to see the commands clearly

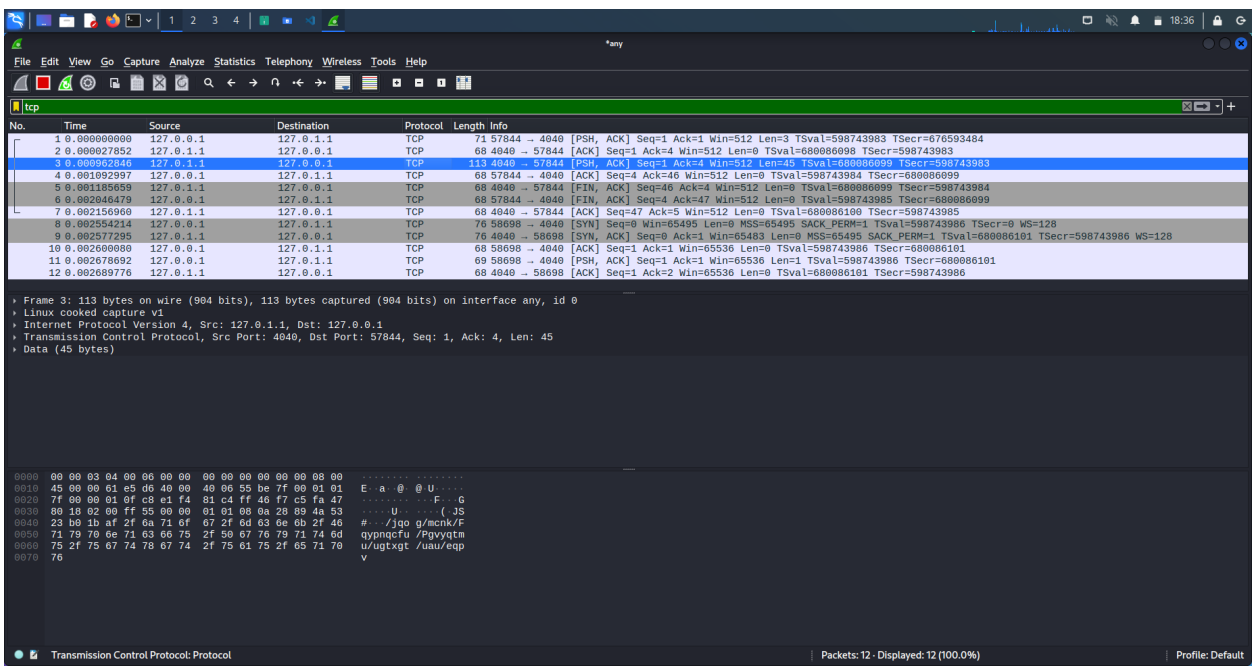


1. **Cwd**

   cwd is encrypted as eyf [offset of 2]

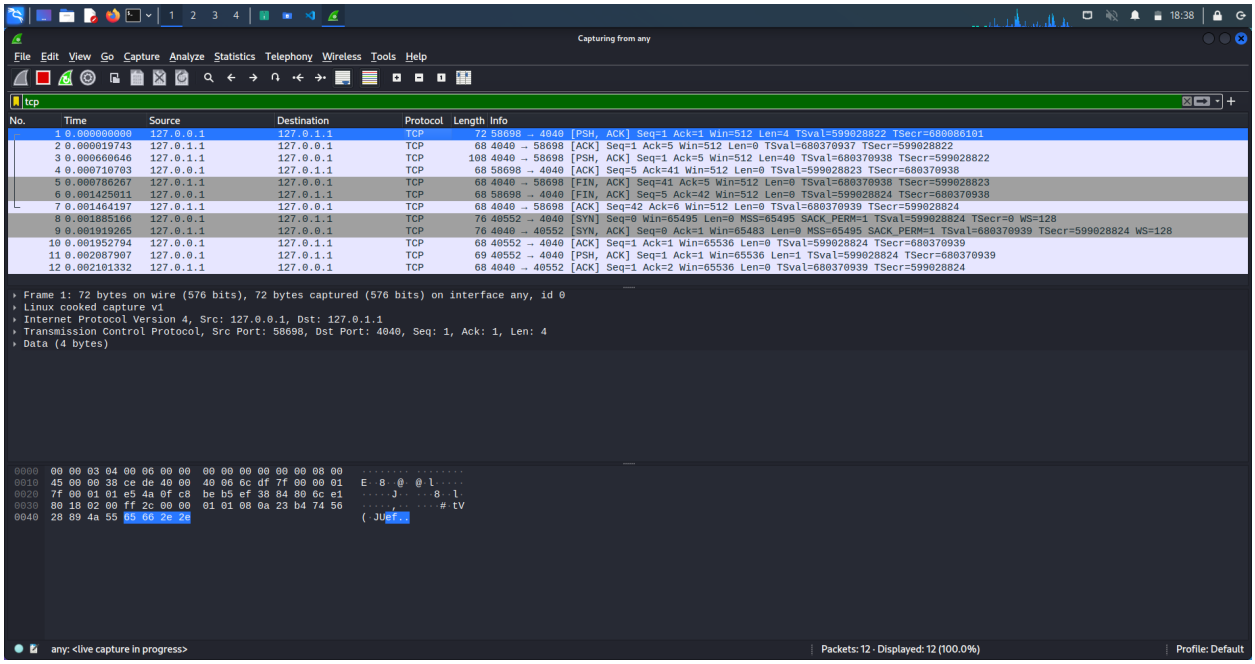


In response of cwd we are getting current working directory with offset of 2

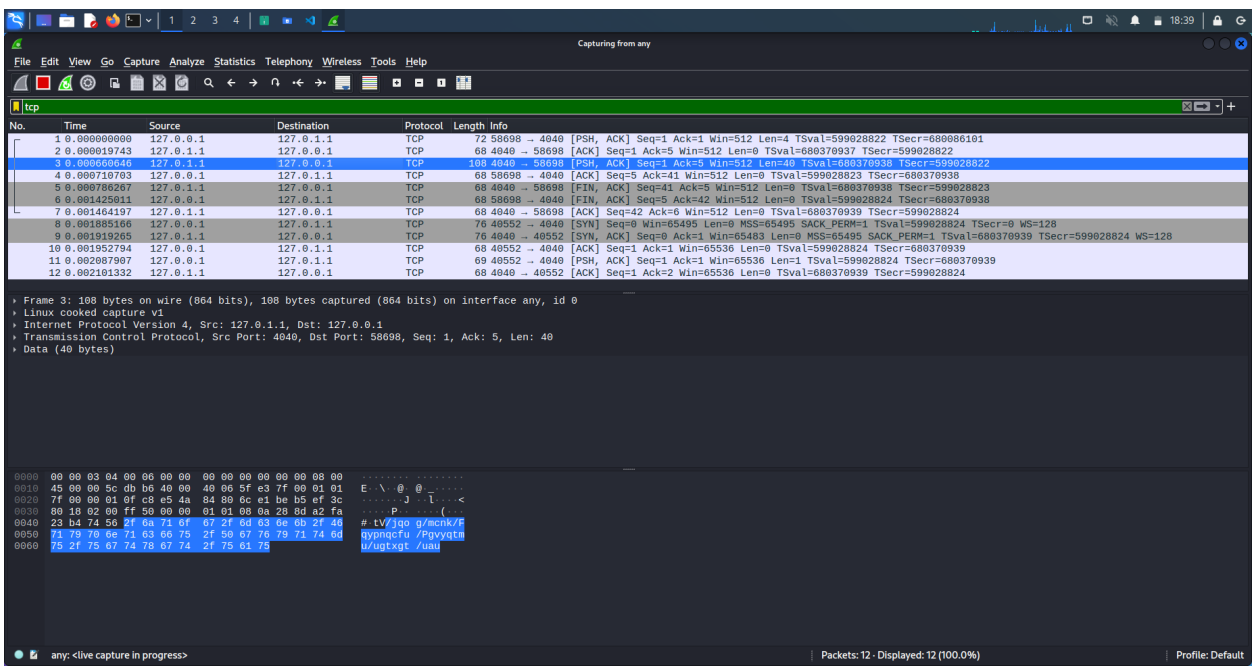


2. Cd

cd command is converted to ef

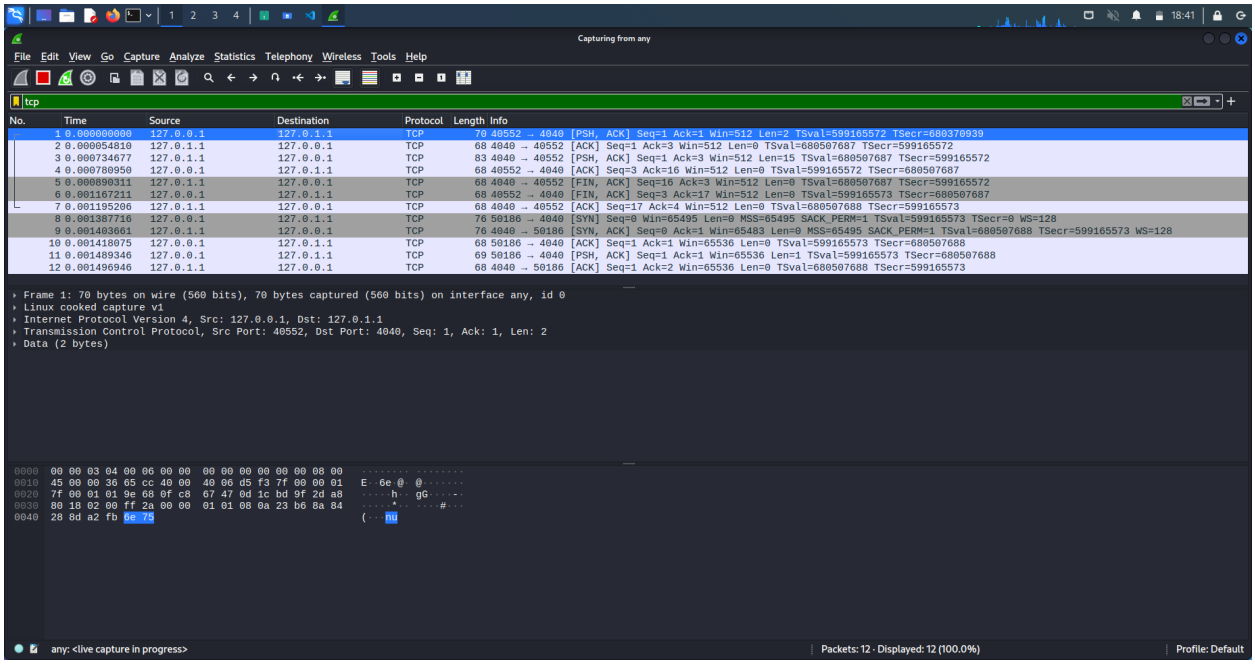


In response we are getting current working directory with offset of 2

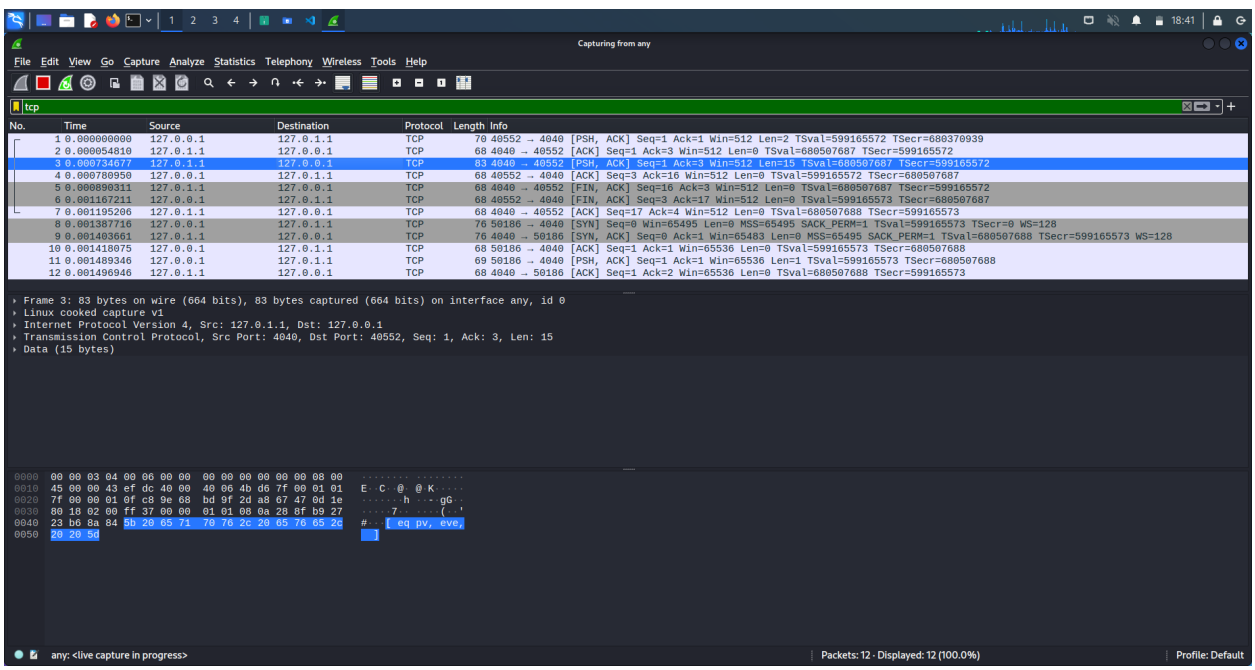


3. Ls

ls is encrypted as nu

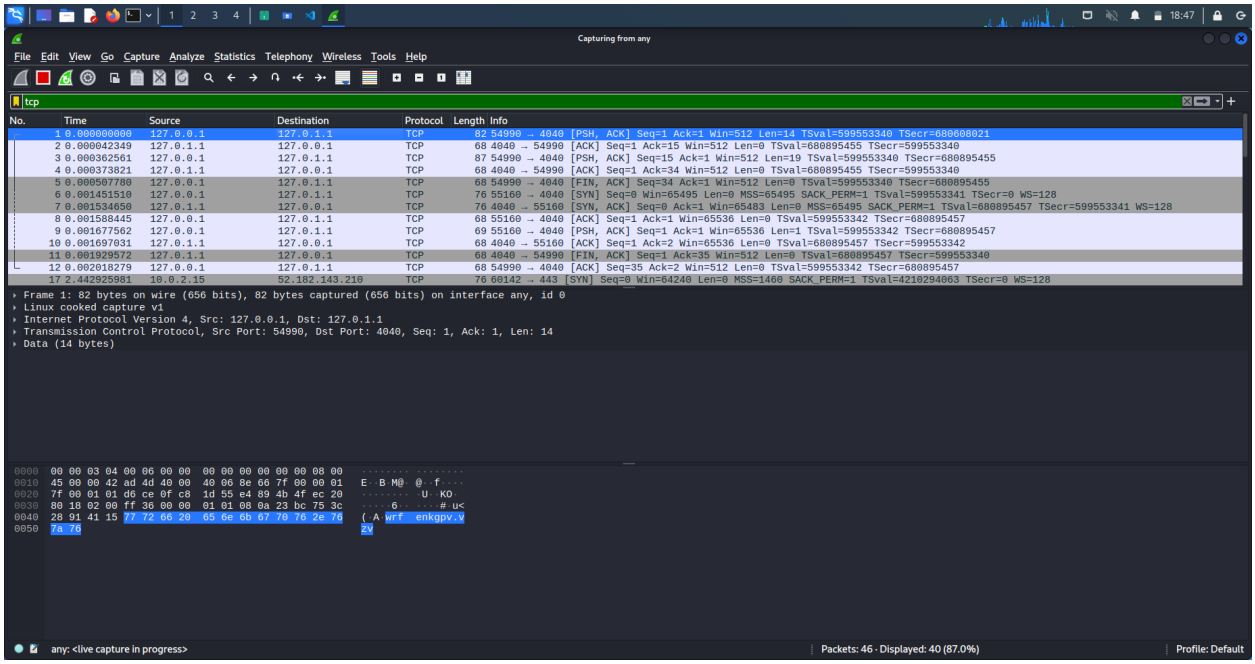


Getting the list of content with offset of 2

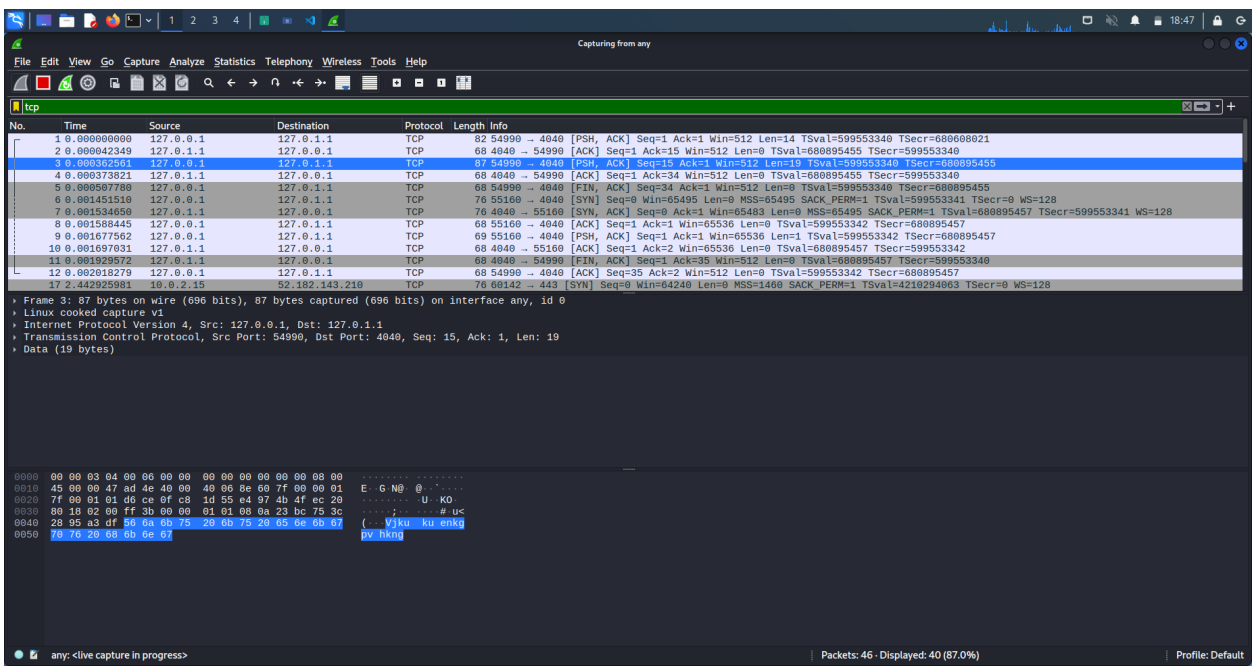


4. Upd

Upd client.txt is encrypted as wrf enkgpv.vzv

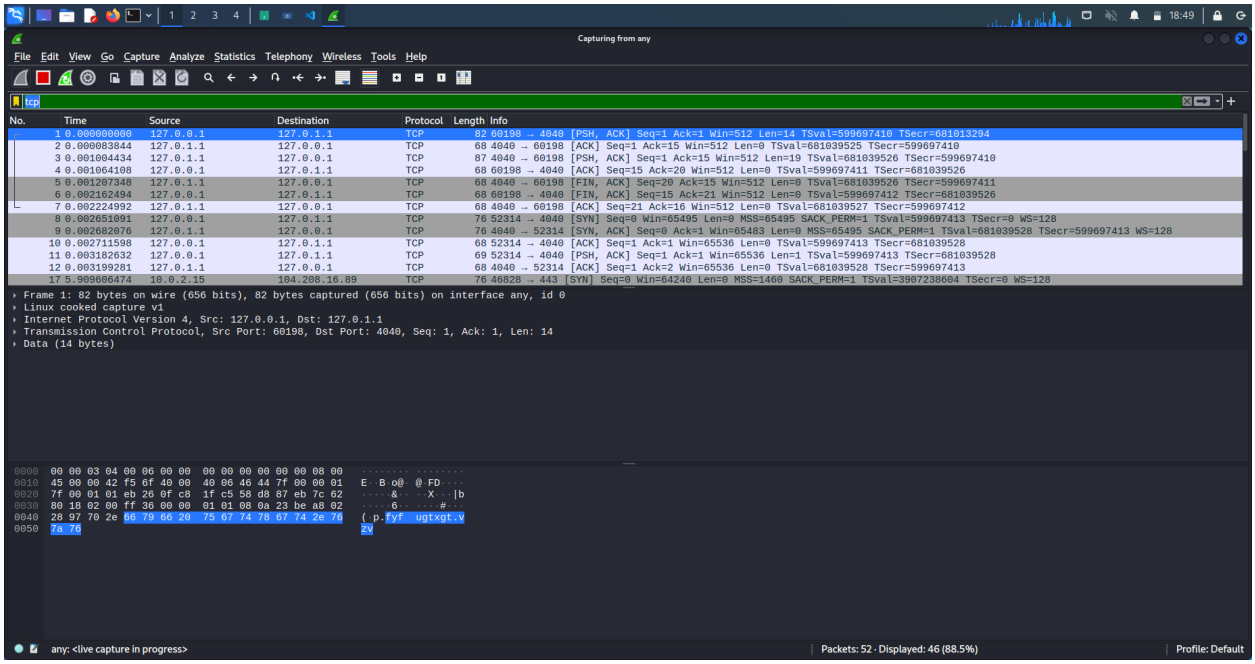


Passing the content of client.txt with an offset of 2

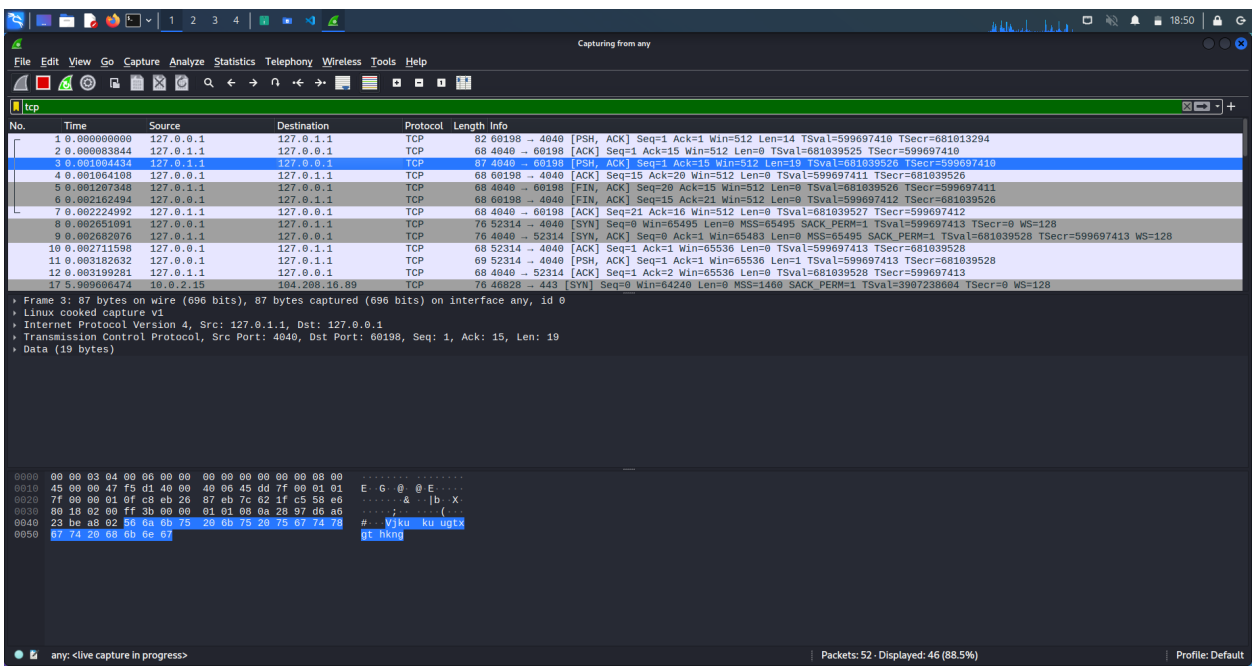


5. Dwd

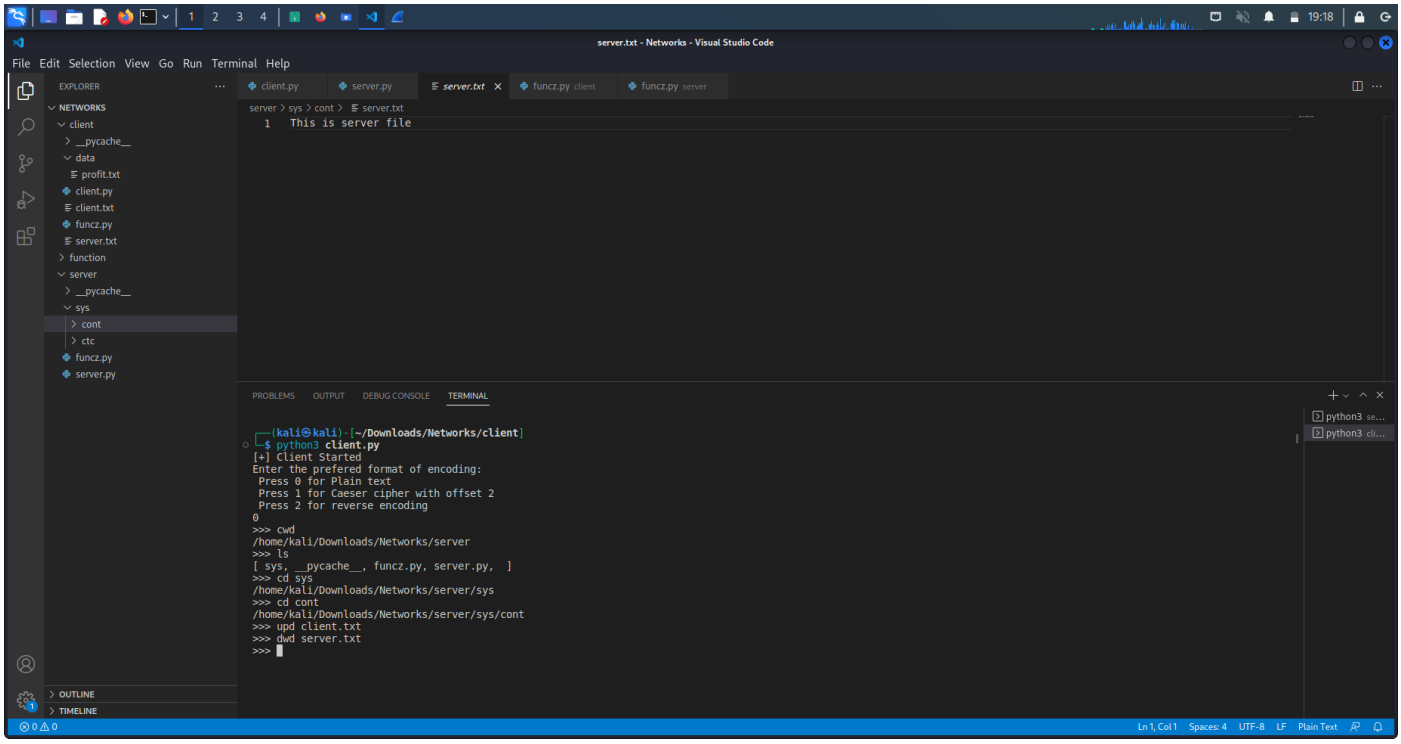
Dwd server.py is encrypted with offset of 2



Content of server.txt with an offset of 2

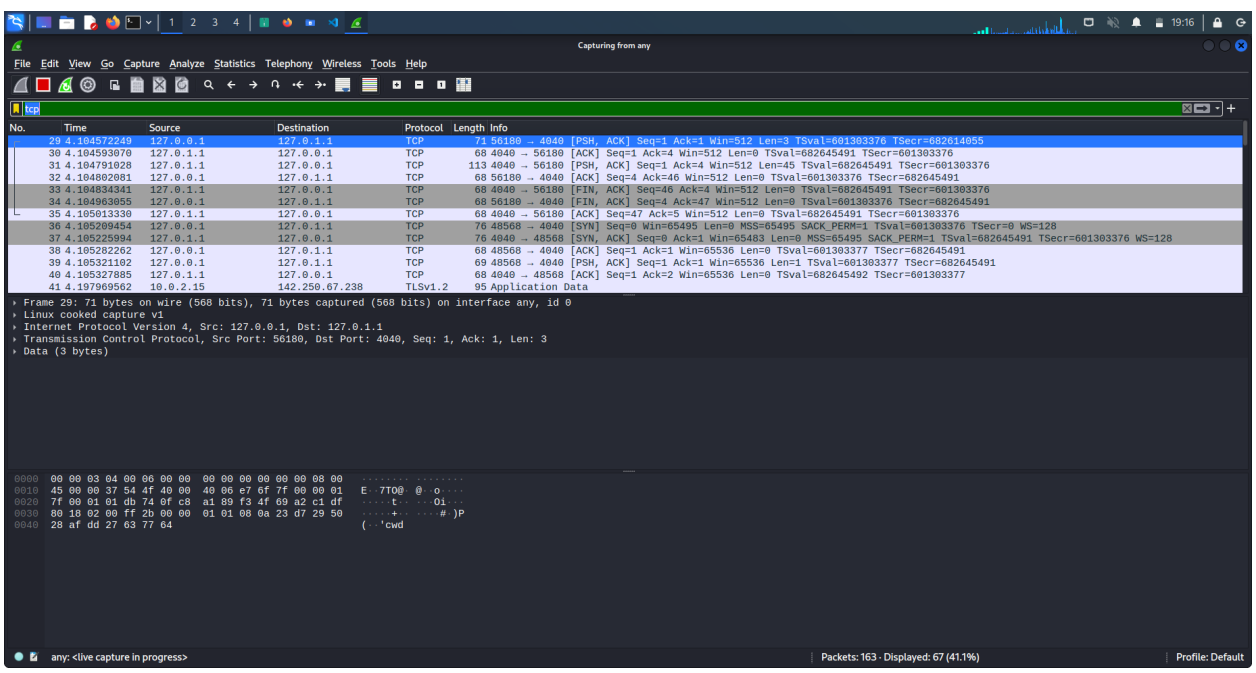


Encoding style: Plain text

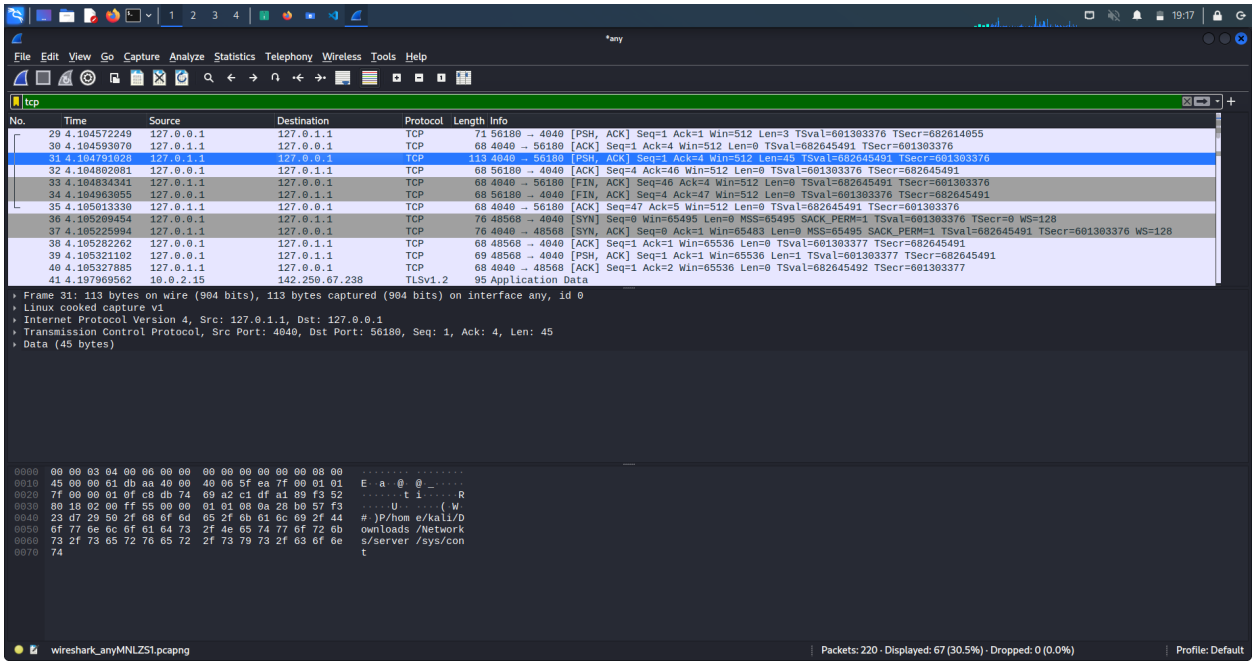


1. Cwd

Cwd is encrypted same as cwd

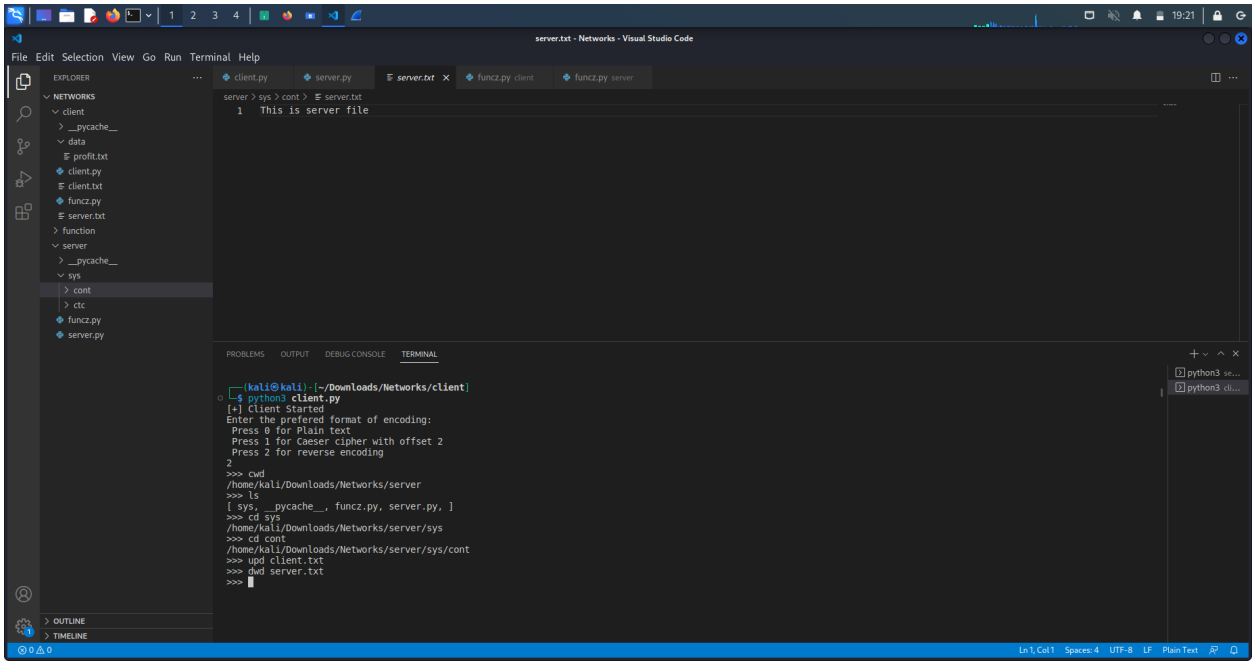


In response we are getting current working directory as plain txt.



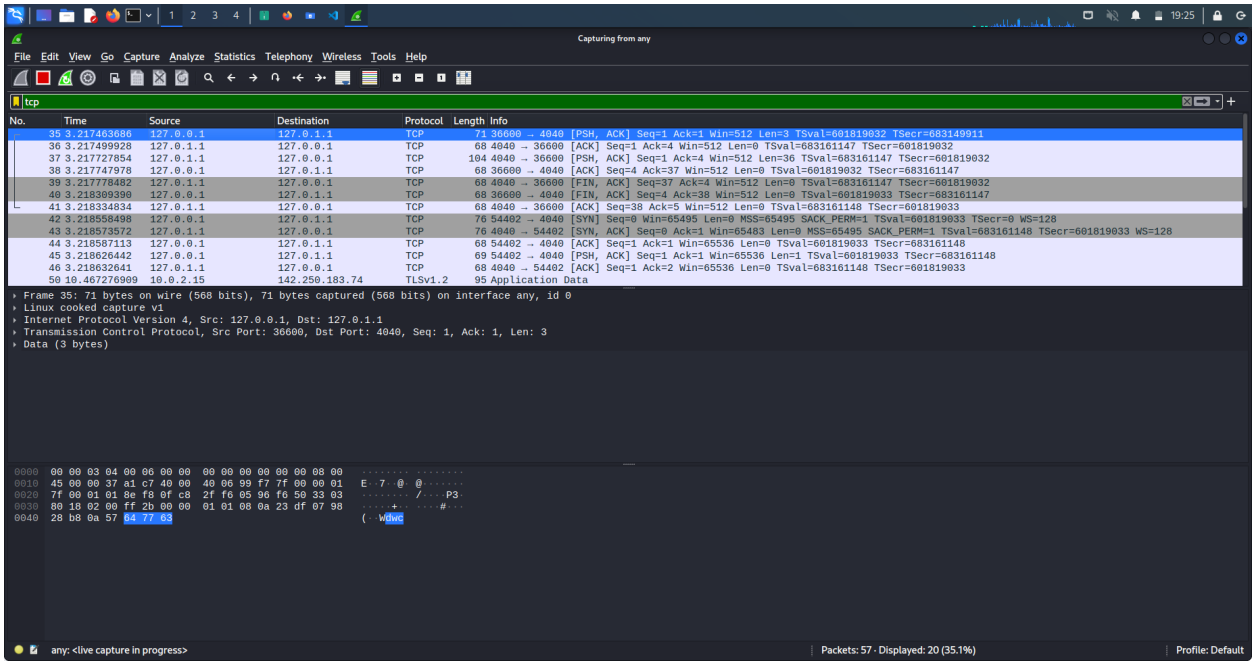
We have successfully analyzed caesar cipher encoding for all commands, we have analyzed request and response of all 5 commands and proved that it is working correctly. Now, we have showed that cwd is working fine with Plain text encoding style so by these two conditions it will also work same for commands like cd, ls, upd, dwd

Encoding style: Reverse encoding

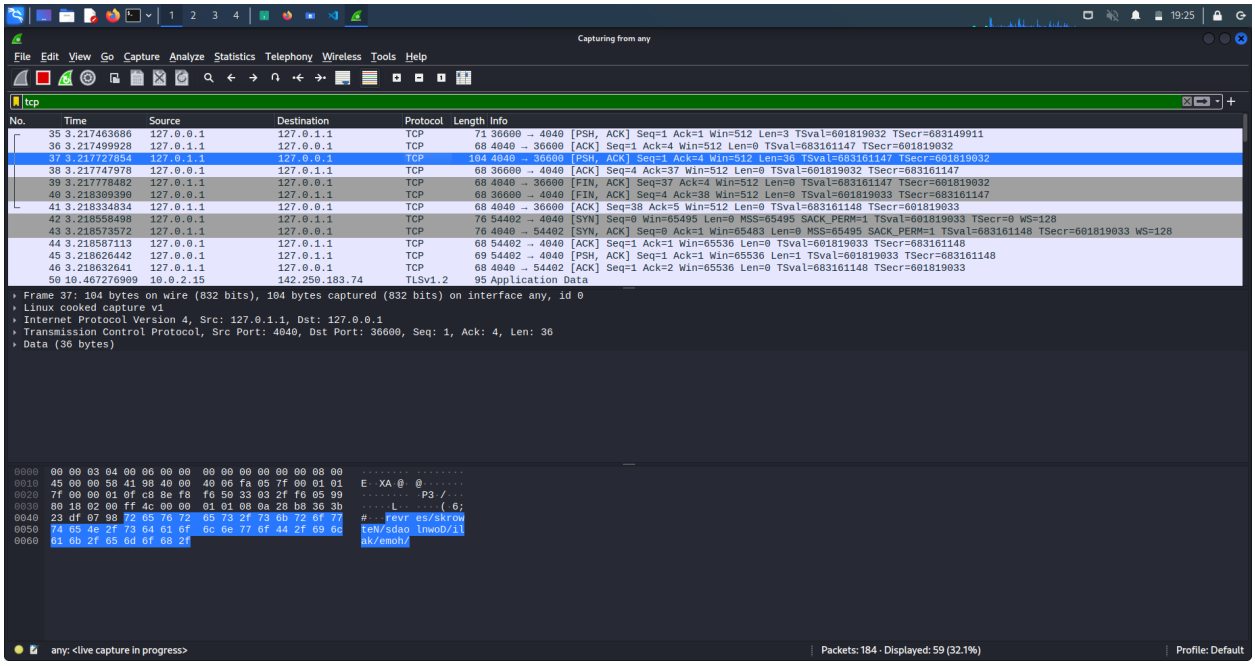


1. Cwd

Cwd is encrypted as dwc



In response we are getting current working directory in reversed fashion



We have successfully analyzed caesar cipher encoding for all commands, we have analyzed request and response of all 5 commands and proved that it is working correctly. Now, we have showed that cwd is working fine with reverse encoding style so by these two conditions it will also work same for commands like cd, ls, upd, dwd