CS433: Computer Networks
Design Document
Assignment 1

KANISHK SINGHAL
20110091

Overview:
To understand the concepts of computer networking and learn how communication happens progress between two systems, a python based server-client system is implemented in this assignment. It uses the TCP protocol for communication and supports five different commands on the client side.

CMD	Description
CWD	Retrieve the path of the current working directory for the user
LS	List the files/folders present in the current working directory
CD <dir>	Change the directory to <dir> as specified by the client
DWD <file>	Download the <file> specified by the user on server to client
UPD <file>	Upload the <file> on client to the remote server in CWD

Files used at the server side:

- server.py
- crypt.py

Files used at the client side:

- client.py
- crypt.py

Mode of Layering:

1) Introduction

Communication between computers to form a network is a complex process. To ease the communication process, it is better to divide the task into several tasks and work on them individually. Network layering is an implementation of such an idea of divide and conquer. Layered Architecture adds its service to a higher layer building the packet that travels over networks to reach its destination.

2) For this implementation, there are three essential layers:

- ***File service layer*:** The topmost layer or the layer closest to the user handles requests given by the user on the client side and generates a proper response on the client side. Commands like **ls**(list files), **cd**(change directory), **cwd**(current directory) are implemented by OS APIs of python.

```
# ? COMMAND ls: LIST FILES IN CURRENT DIR
def listfiles(path):
    send(" ".join(os.listdir(path)))

# ? COMMAND cd: CHANGE CURRENT DIR, SEND STATUS OK or NOK
def changeDir(path):
    if os.access(path, os.F_OK):
        os.chdir(path)
        send(f"OK: Directory changed to {os.getcwd()}")
    else:
        send("NOK: This directory does not exist")

# ? COMMAND cwd: RETURN CURRENT WORKING DIRECTORY
def currentDir():
    send(f"Current Directory is {os.getcwd()}")
```

Upload and download commands read and write in files, thus using file handling functions. Data is transferred and received in chunks which have to be managed over the network for lossless transfer of files.

This is achieved by sending confirmation of the received packet or chunk of the file. If the packet is not received correctly, it is resent over the network.

```
# SENDING FILEDATA IN CHUNKS TO ACCOMODATE VARIABLE FILE SIZE
while filesize > 0:
    # SENDING 2048 BYTES OF DATA
    filedata = file.read(2048)
    client.send((cryptInd.encode("utf-8") + cryption[cryptInd](filedata, 2)))
    filesize = filesize - 2048
    # RECEIVING CONFIRMATION
    res = recv(1024)
    while res[0:3] == "NOK":
        client.send((cryptInd.encode("utf-8") + cryption[cryptInd](filedata, 2)))
        res = recv(1024)
```

- ***Crypto layer***: The second layer handles the safety of the data protecting it from other users or hackers trying to access information not meant for them. The crypto layer takes input from the File service layer as data to be sent and encrypts in one of three manners mentioned in the assignment. The user can change the mode of encryption using **chcrypt <0-2>** command. The **shcrypt** command displays the current mode of encryption.

```
#? COMMAND "chcrypt <0-2>": CHANGE CURRENT MODE OF
#? ENCRYPTION TO INTEGER BETWEEN 0-2
# 0 -> Plain Text
# 1 -> Caesar Cipher
# 2 -> Transpose
if command[0:7] == 'chcrypt':
    if ' ' in command:
        cryptInd = command.split()[1]
        print("Crypt mode changed to " + cryptInd)
    else:
        print("NOK: Give a integer between 0 to 2 with command")
    continue

#? COMMAND "shcrypt": SHOW CURRENT MODE OF ENCRYPTION
if command[0:7] == 'shcrypt':
    print("Crypt mode is " + cryptInd)
    continue
```

The output of the crypto layer builds over the File service layer. It takes the command or file data from a higher layer as input and adds the mode of encryption(‘0’, ‘1’, or ‘2’) as a header to it when sending the message. While receiving the message, the crypto layer extracts the header, which determines the mode of encryption of the message and decrypts it using functions defined in **cryption** dictionary in the crypt.py file.

```
# Utility Functions to send and recieve data
def send(toSend):
    # Adding "cryptInd" as header and encypting the message before sending
    client.send((cryptInd + cryption[cryptInd](toSend, 2)).encode("utf-8"))

def recv(bufferSize):
    # Extracting "cryptInd" from header and decypting the message on receiving
    res = client.recv(bufferSize).decode('utf-8')
    res = cryption[res[0]](res[1:], -2)
    return res
```

For the encryption of data, the user is provided with three choices:

- *Plain Text*: In this mode, no encryption is performed. The data is sent as it is combined with the mode of encryption.
- *Substitution*: Substitution or Caesar Cipher shifts each letter or byte by fixed places. For example, if the shift is of 2 letters, ‘a’ will be shifted to ‘c’, ‘b’ will be shifted to ‘d’, and so on. While receiving the message, a shift of negative character is done with the same amount in encryption.
  For encrypting bytes by this method, first, bytes are converted into list objects and the shift amount is added to each list member and subtracted while decrypting.
- *Transpose*: The transpose method reverses the data sent, and while receiving the data, another reversal is done, which nullifies the effect.

- ***Networking layer*:** The network layer is the layer that enables communication between the client and the server. This layer is majorly implemented inside python and abstraction of this layer is used using sockets provided in the language. For this assignment, the TCP layer is used to send and receive messages. This is mentioned when using SOCK_STREAM while defining the socket for server and client.

Execution of commands in File service layer

1) ***LS*: List Files**
   The 'ls' command lists the files in the current folder of the server. It uses OS APIs to find the response.

2) ***CD*: Change Directory**
   The 'cd' command changes the current working directory of the server.

3) ***CWD*: Change Working Directory**
   The 'cwd' command returns the current working directory of the server.

4) **UPD: Upload File**

The ‘upd’ command uploads a file to the server. The file to be uploaded is passed as an argument with command, for example, “ upd test1.jpg ”.

First, the client checks if the file to upload is available in the system. After checking, the client sends the size of the file to the server and waits for a confirmation message.

For uploading file regardless of size, the file is uploaded in chunks. A part of the file(2048 bytes) is read at a time and is sent to the server. Then the client waits for a confirmation message after the server successfully receives the file chunk.

Reading in chunks and sending the data happens inside of a loop that runs for (filesize/2048+1) times. This ensures that files of any size can be sent over the socket.

The server opens a file with the name received from the command in write mode. It writes the data in a loop that runs for same number of iterations as above. After completion the file is closed and “OK: Complete File Uploaded Successfully” message is sent to the client.

```
[CLIENT]: upd test0.txt
[SERVER]: OK: Complete File(test0.txt) Uploaded Successfully
```

5) **DWD: Download file**

The ‘dwd’ command downloads a file from server to the client. The file to be uploaded is passed as an argument with command for example, “dwd test1.jpg”.

Upon receiving the command the server first checks if the requested file is available to download. If the file is available a similar method of sending the file in chunks which is used in “upload files” is used. If the file is not available a “NOK:...” message is sent to client. To prevent any data loss or corruption at the server side a confirmation message is also sent to the server by the client after each packet.

```
[CLIENT]: dwd test0.txt
[SERVER]: OK: Complete File(test0.txt) Downloaded Successfully
```

Wireshark analysis:

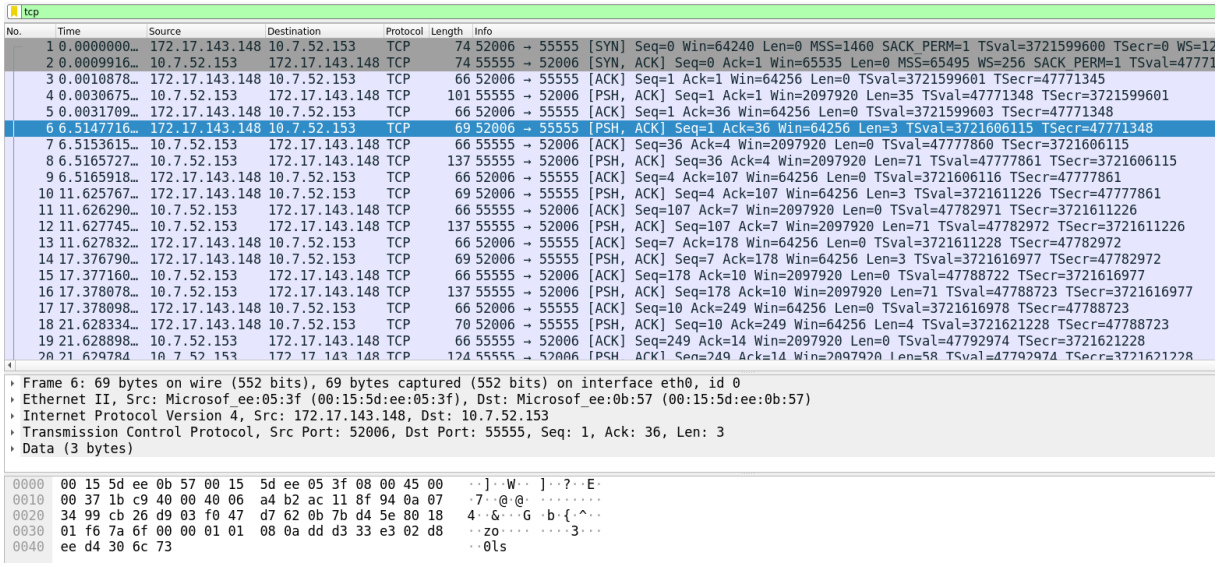
SERVER IP : 10.7.52.153
CLIENT IP: 172.17.143.148

1) LS: list file

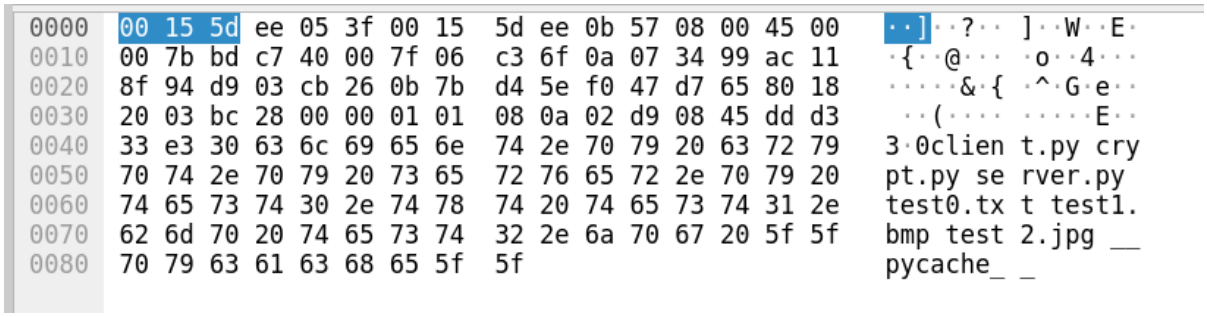
```
❯ python3 client.py
[SERVER]: You are connected to (10.7.52.153)
[CLIENT]: chcrypt 0
Crypt mode changed to 0
[CLIENT]: ls
[SERVER]: client.py crypt.py server.py test0.txt test1.bmp test2.jpg __pycache__
[CLIENT]: chcrypt 1
Crypt mode changed to 1
[CLIENT]: ls
[SERVER]: client.py crypt.py server.py test0.txt test1.bmp test2.jpg __pycache__
[CLIENT]: chcrypt 2
Crypt mode changed to 2
[CLIENT]: ls
[SERVER]: client.py crypt.py server.py test0.txt test1.bmp test2.jpg __pycache__
[CLIENT]: cwd
[SERVER]: Current Directory is D:\coding\python\ComputerNetworks\A1
[CLIENT]: cd ../
[SERVER]: OK: Directory changed to D:\coding\python\ComputerNetworks
[CLIENT]: cd A1
[SERVER]: OK: Directory changed to D:\coding\python\ComputerNetworks\A1
```

Mode of encryption-0 (Plain Text)

-> Request:

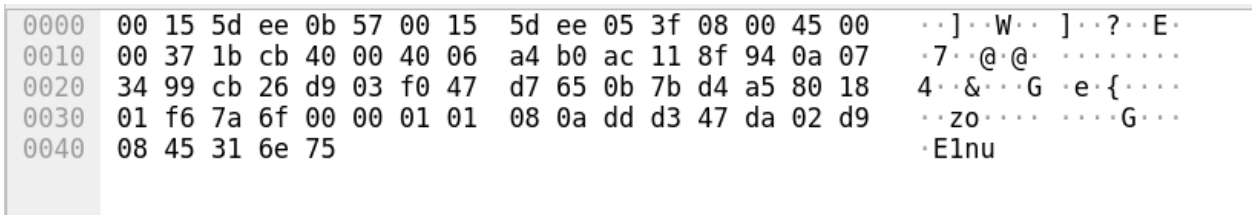


-> Response



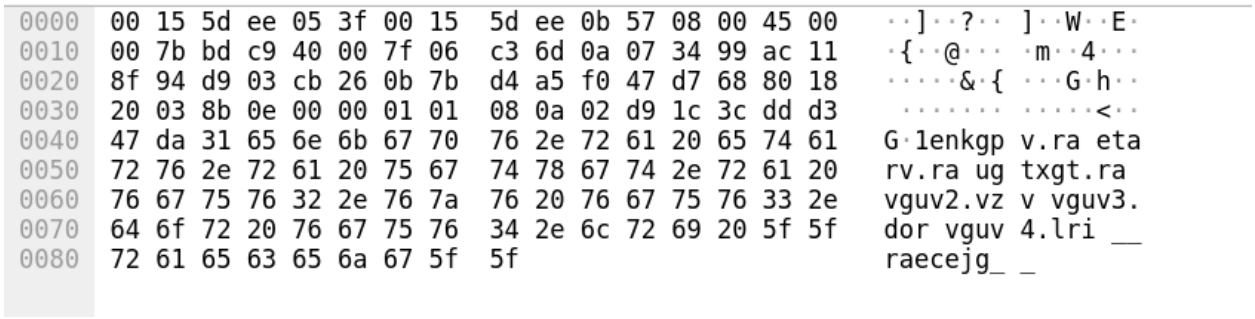
Mode of encryption-0 (Substitution)

-> Request:



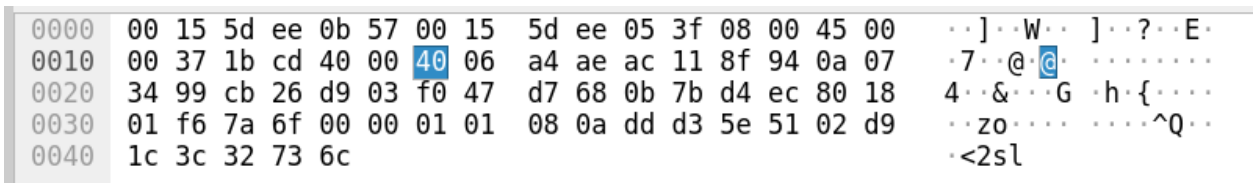
1nu is sent to server. “ls” translates to “nu” when each letter is shifted by 2. 1 is the mode of encryption.

-> Response:



Mode of encryption-2 (Transpose)

-> Request:



‘Ls’ is sent as ‘sl’ with header ‘2’.

-> Response:

```
0000  00 15 5d ee 05 3f 00 15  5d ee 0b 57 08 00 45 00   ··]··?·· ]··W··E·
0010  00 7b bd cb 40 00 7f 06  c3 6b 0a 07 34 99 ac 11   ·{··@··· ·k··4···
0020  8f 94 d9 03 cb 26 0b 7b  d4 ec f0 47 d7 6b 80 18   ·····&·{ ···G·k··
0030  20 03 76 a6 00 00 01 01  08 0a 02 d9 32 b3 dd d3    ·v····· ····2···
0040  5e 51 32 5f 5f 65 68 63  61 63 79 70 5f 5f 20 67   ^Q2__ehc acyp__ g
0050  70 6a 2e 32 74 73 65 74  20 70 6d 62 2e 31 74 73   pj.2tset  pmb.1ts
0060  65 74 20 74 78 74 2e 30  74 73 65 74 20 79 70 2e   et txt.0 tset yp.
0070  72 65 76 72 65 73 20 79  70 2e 74 70 79 72 63 20   revres y p.tpyrc
0080  79 70 2e 74 6e 65 69 6c  63                       yp.tneil c
```

Response is received in backward form.

2) **Download file**

```
[CLIENT]: dwd test0.txt
[SERVER]: OK: Complete File(test0.txt) Downloaded Successfully
[CLIENT]: upd tets0.txt
NOK: The path does not exists
[CLIENT]: dwd test0.txt
[SERVER]: OK: Complete File(test0.txt) Downloaded Successfully
```

Mode of encryption-2 (Transpose)

-> File Download Request from client:

```
0000  00 15 5d ee 0b 57 00 15  5d ee 05 3f 08 00 45 00   ··]··W·· ]··?··E·
0010  00 42 1b d5 40 00 40 06  a4 9b ac 11 8f 94 0a 07   ·B··@·@· ········
0020  34 99 cb 26 d9 03 f0 47  d7 7c 0b 7b d5 e6 80 18   4··&···G ·|·{····
0030  01 f6 7a 7a 00 00 01 01  08 0a dd d3 b8 cd 02 d9   ··zz···· ········
0040  5c f9 32 74 78 74 2e 30  74 73 65 74 20 64 77 64   \·2txt.0 tset dwd
```

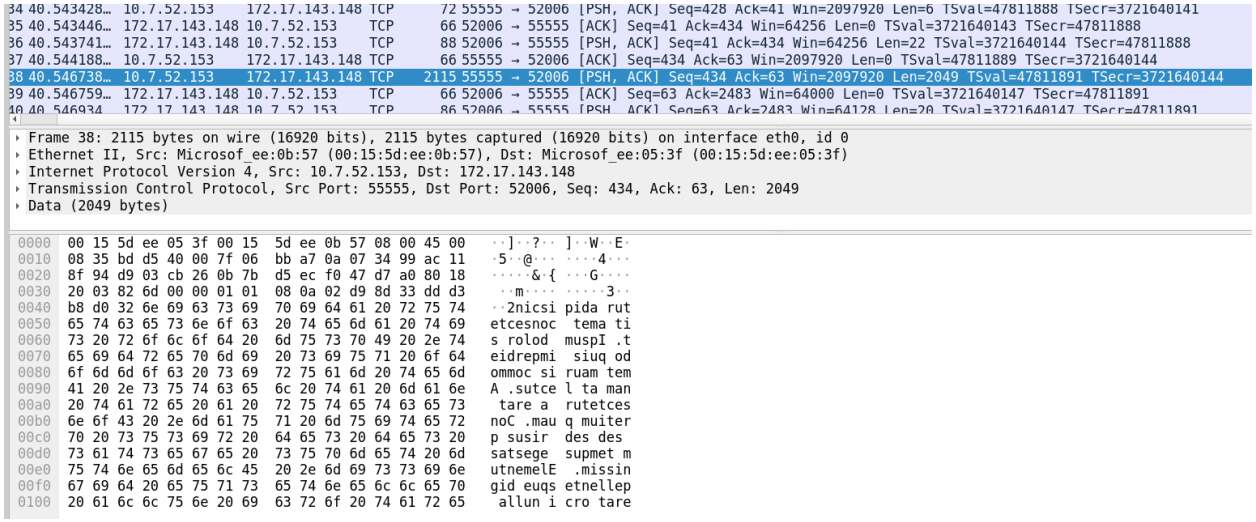
-> File size response from server

```
0000  00 15 5d ee 05 3f 00 15  5d ee 0b 57 08 00 45 00   ··]··?·· ]··W··E·
0010  00 3a bd d3 40 00 7f 06  c3 a4 0a 07 34 99 ac 11   ·:··@··· ····4···
0020  8f 94 d9 03 cb 26 0b 7b  d5 e6 f0 47 d7 8a 80 18   ·····&·{ ···G····
0030  20 03 d0 c0 00 00 01 01  08 0a 02 d9 8d 30 dd d3    ·····0··
0040  b8 cd 32 38 35 32 30 31                           ··285201
```

-> File size received at client

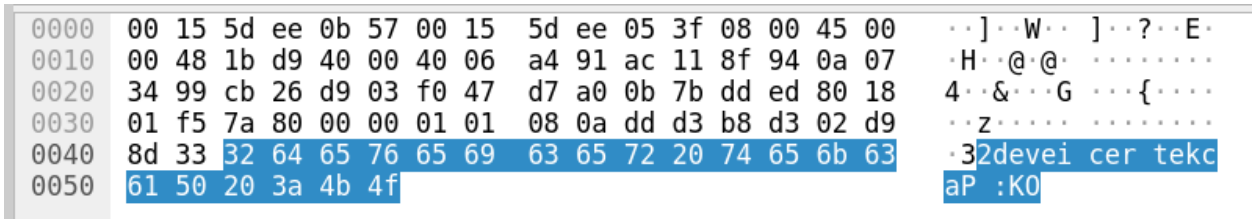
```
0000  00 15 5d ee 0b 57 00 15  5d ee 05 3f 08 00 45 00   ··]··W·· ]··?··E·
0010  00 4a 1b d7 40 00 40 06  a4 91 ac 11 8f 94 0a 07   ·J··@·@· ········
0020  34 99 cb 26 d9 03 f0 47  d7 8a 0b 7b d5 ec 80 18   4··&···G ···{····
0030  01 f6 7a 82 00 00 01 01  08 0a dd d3 b8 d0 02 d9   ··z····· ········
0040  8d 30 32 64 65 76 65 69  63 65 72 20 65 7a 69 73   ·02devei cer ezis
0050  65 6c 69 46 20 3a 4b 4f                           eliF :KO
```

-> File data sent by server(First chunk)



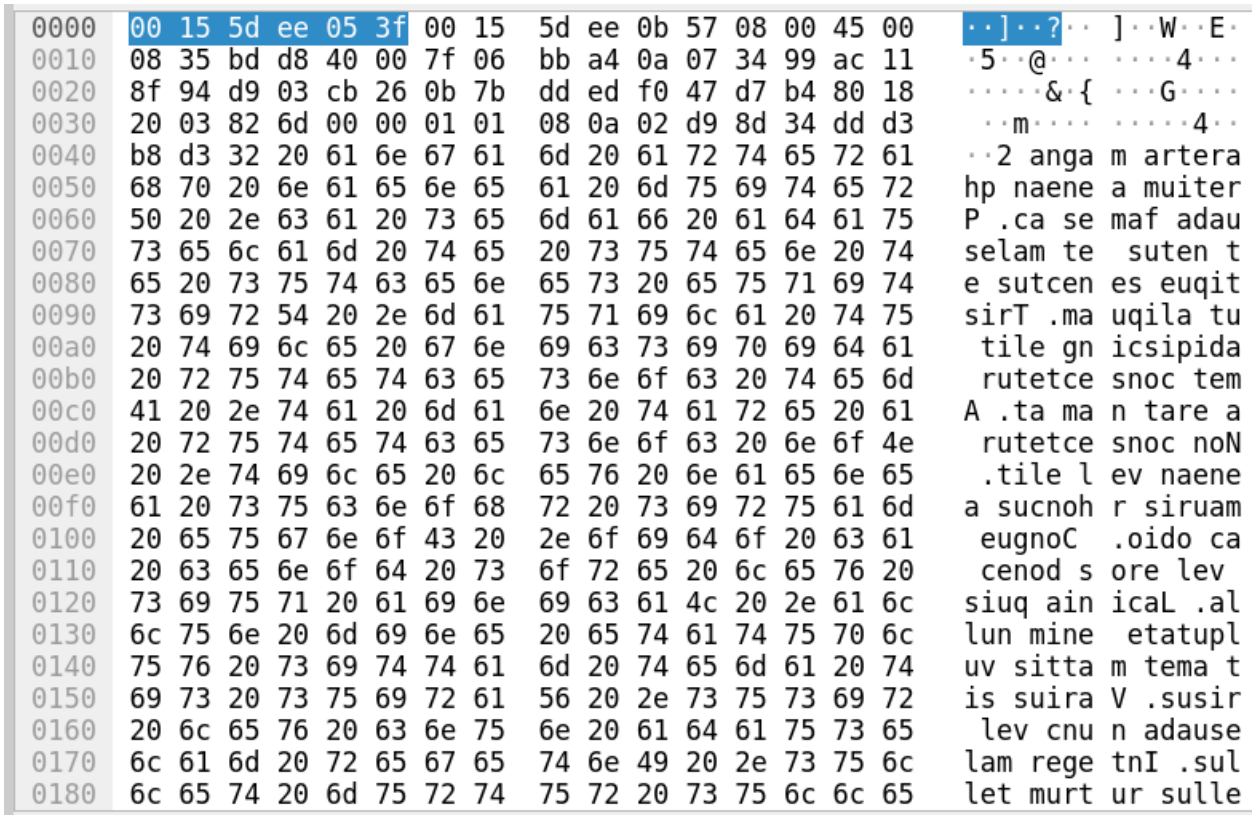
```
34 40.543428… 10.7.52.153    172.17.143.148 TCP      72 55555 → 52006 [PSH, ACK] Seq=428 Ack=41 Win=2097920 Len=6 TSval=47811888 TSecr=3721640141
35 40.543446… 172.17.143.148 10.7.52.153    TCP      66 52006 → 55555 [ACK] Seq=41 Ack=434 Win=64256 Len=0 TSval=3721640143 TSecr=47811888
36 40.543741… 172.17.143.148 10.7.52.153    TCP      88 52006 → 55555 [PSH, ACK] Seq=41 Ack=434 Win=64256 Len=22 TSval=3721640144 TSecr=47811888
37 40.544188… 10.7.52.153    172.17.143.148 TCP      66 55555 → 52006 [ACK] Seq=434 Ack=63 Win=2097920 Len=0 TSval=47811889 TSecr=3721640144
38 40.546738… 10.7.52.153    172.17.143.148 TCP    2115 55555 → 52006 [PSH, ACK] Seq=434 Ack=63 Win=2097920 Len=2049 TSval=47811891 TSecr=3721640144
39 40.546759… 172.17.143.148 10.7.52.153    TCP      66 52006 → 55555 [ACK] Seq=63 Ack=2483 Win=64000 Len=0 TSval=3721640147 TSecr=47811891
40 40.546934… 172.17.143.148 10.7.52.153    TCP      86 52006 → 55555 [PSH, ACK] Seq=63 Ack=2483 Win=64128 Len=20 TSval=3721640147 TSecr=47811891

▸ Frame 38: 2115 bytes on wire (16920 bits), 2115 bytes captured (16920 bits) on interface eth0, id 0
▸ Ethernet II, Src: Microsof_ee:0b:57 (00:15:5d:ee:0b:57), Dst: Microsof_ee:05:3f (00:15:5d:ee:05:3f)
▸ Internet Protocol Version 4, Src: 10.7.52.153, Dst: 172.17.143.148
▸ Transmission Control Protocol, Src Port: 55555, Dst Port: 52006, Seq: 434, Ack: 63, Len: 2049
▸ Data (2049 bytes)

0000  00 15 5d ee 05 3f 00 15  5d ee 0b 57 08 00 45 00   ··]··?·· ]··W··E·
0010  08 35 bd d5 40 00 7f 06  bb a7 0a 07 34 99 ac 11   ·5··@··· ····4···
0020  8f 94 d9 03 cb 26 0b 7b  d5 ec f0 47 d7 a0 80 18   ·····&·{ ···G····
0030  20 03 82 6d 00 00 01 01  08 0a 02 d9 8d 33 dd d3    ··m···· ·····3··
0040  b8 d0 32 6e 69 63 73 69  70 69 64 61 20 72 75 74   ··2nicsi pida rut
0050  65 74 63 65 73 6e 6f 63  20 74 65 6d 61 20 74 69   etcesnoc  tema ti
0060  73 20 72 6f 6c 6f 64 20  6d 75 73 70 49 20 2e 74   s rolod  muspI .t
0070  65 69 64 72 65 70 6d 69  20 73 69 75 71 20 6f 64   eidrepmi  siuq od
0080  6f 6d 6d 6f 63 20 73 69  72 75 61 6d 20 74 65 6d   ommoc si ruam tem
0090  41 20 2e 73 75 74 63 65  6c 20 74 61 20 6d 61 6e   A .sutce l ta man
00a0  20 74 61 72 65 20 61 20  72 75 74 65 74 63 65 73    tare a  rutetces
00b0  6e 6f 43 20 2e 6d 61 75  71 20 6d 75 69 74 65 72   noC .mau q muiter
00c0  70 20 73 75 73 69 72 20  64 65 73 20 64 65 73 20   p susir  des des
00d0  73 61 74 73 65 67 65 20  73 75 70 6d 65 74 20 6d   satsege  supmet m
00e0  75 74 6e 65 6d 65 6c 45  20 2e 6d 69 73 73 69 6e   utnemelE  .missin
00f0  67 69 64 20 65 75 71 73  65 74 6e 65 6c 6c 65 70   gid euqs etnellep
0100  20 61 6c 6c 75 6e 20 69  63 72 6f 20 74 61 72 65    allun i cro tare
```

-> Packet confirmation(First Chunk)



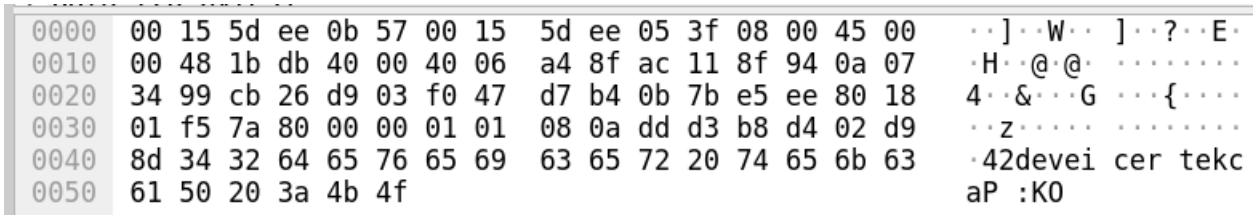
```
0000  00 15 5d ee 0b 57 00 15  5d ee 05 3f 08 00 45 00   ··]··W·· ]··?··E·
0010  00 48 1b d9 40 00 40 06  a4 91 ac 11 8f 94 0a 07   ·H··@·@· ········
0020  34 99 cb 26 d9 03 f0 47  d7 a0 0b 7b dd ed 80 18   4··&···G ···{····
0030  01 f5 7a 80 00 00 01 01  08 0a dd d3 b8 d3 02 d9   ··z····· ········
0040  8d 33 32 64 65 76 65 69  63 65 72 20 74 65 6b 63   ·32devei cer tekc
0050  61 50 20 3a 4b 4f                                 aP :KO
```

-> File data sent by server(Second chunk)



```
0000  00 15 5d ee 05 3f 00 15  5d ee 0b 57 08 00 45 00   ··]··?·· ]··W··E·
0010  08 35 bd d8 40 00 7f 06  bb a4 0a 07 34 99 ac 11   ·5··@··· ····4···
0020  8f 94 d9 03 cb 26 0b 7b  dd ed f0 47 d7 b4 80 18   ·····&·{ ···G····
0030  20 03 82 6d 00 00 01 01  08 0a 02 d9 8d 34 dd d3    ··m···· ·····4··
0040  b8 d3 32 20 61 6e 67 61  6d 20 61 72 74 65 72 61   ··2 anga m artera
0050  68 70 20 6e 61 65 6e 65  61 20 6d 75 69 74 65 72   hp naene a muiter
0060  50 20 2e 63 61 20 73 65  6d 61 66 20 61 64 61 75   P .ca se maf adau
0070  73 65 6c 61 6d 20 74 65  20 73 75 74 65 6e 20 74   selam te  suten t
0080  65 20 73 75 74 63 65 6e  65 73 20 65 75 71 69 74   e sutcen es euqit
0090  73 69 72 54 20 2e 6d 61  75 71 69 6c 61 20 74 75   sirT .ma uqila tu
00a0  20 74 69 6c 65 20 67 6e  69 63 73 69 70 69 64 61    tile gn icsipida
00b0  20 72 75 74 65 74 63 65  73 6e 6f 63 20 74 65 6d    rutetce snoc tem
00c0  41 20 2e 74 61 20 6d 61  6e 20 74 61 72 65 20 61   A .ta ma n tare a
00d0  20 72 75 74 65 74 63 65  73 6e 6f 63 20 6e 6f 4e    rutetce snoc noN
00e0  20 2e 74 69 6c 65 20 6c  65 76 20 6e 61 65 6e 65    .tile l ev naene
00f0  61 20 73 75 63 6e 6f 68  72 20 73 69 72 75 61 6d   a sucnoh r siruam
0100  20 65 75 67 6e 6f 43 20  2e 6f 69 64 6f 20 63 61    eugnoC  .oido ca
0110  20 63 65 6e 6f 64 20 73  6f 72 65 20 6c 65 76 20    cenod s ore lev
0120  73 69 75 71 20 61 69 6e  69 63 61 4c 20 2e 61 6c   siuq ain icaL .al
0130  6c 75 6e 20 6d 69 6e 65  20 65 74 61 74 75 70 6c   lun mine  etatupl
0140  75 76 20 73 69 74 74 61  6d 20 74 65 6d 61 20 74   uv sitta m tema t
0150  69 73 20 73 75 69 72 61  56 20 2e 73 75 73 69 72   is suira V .susir
0160  20 6c 65 76 20 63 6e 75  6e 20 61 64 61 75 73 65    lev cnu n adause
0170  6c 61 6d 20 72 65 67 65  74 6e 49 20 2e 73 75 6c   lam rege tnI .sul
0180  6c 65 74 20 6d 75 72 74  75 72 20 73 75 6c 6c 65   let murt ur sulle
```

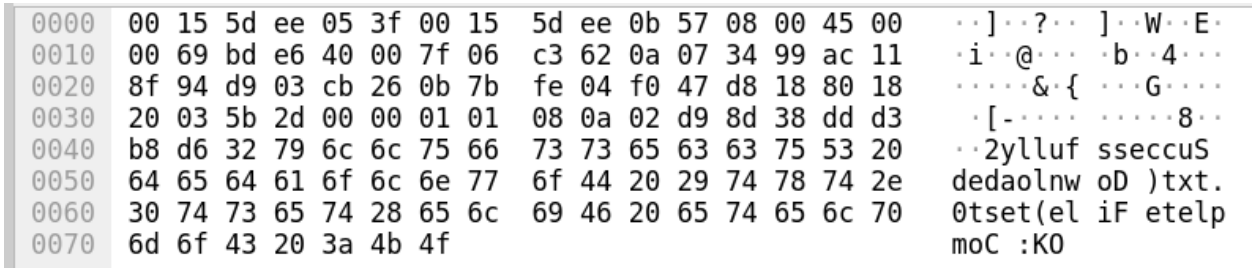
-> Packet Confirmation(Second Chunk)



```
0000  00 15 5d ee 0b 57 00 15  5d ee 05 3f 08 00 45 00   ··]··W·· ]··?··E·
0010  00 48 1b db 40 00 40 06  a4 8f ac 11 8f 94 0a 07   ·H··@·@· ········
0020  34 99 cb 26 d9 03 f0 47  d7 b4 0b 7b e5 ee 80 18   4··&···G ···{····
0030  01 f5 7a 80 00 00 01 01  08 0a dd d3 b8 d4 02 d9   ··z····· ········
0040  8d 34 32 64 65 76 65 69  63 65 72 20 74 65 6b 63   ·42devei cer tekc
0050  61 50 20 3a 4b 4f                                 aP :KO
```

…

-> Complete File downloaded status



```
0000  00 15 5d ee 05 3f 00 15  5d ee 0b 57 08 00 45 00   ··]··?·· ]··W··E·
0010  00 69 bd e6 40 00 7f 06  c3 62 0a 07 34 99 ac 11   ·i··@··· ·b··4···
0020  8f 94 d9 03 cb 26 0b 7b  fe 04 f0 47 d8 18 80 18   ·····&·{ ···G····
0030  20 03 5b 2d 00 00 01 01  08 0a 02 d9 8d 38 dd d3    ·[-····· ·····8··
0040  b8 d6 32 79 6c 6c 75 66  73 73 65 63 63 75 53 20   ··2ylluf sseccuS
0050  64 65 64 61 6f 6c 6e 77  6f 44 20 29 74 78 74 2e   dedaolnw oD )txt.
0060  30 74 73 65 74 28 65 6c  69 46 20 65 74 65 6c 70   0tset(el iF etelp
0070  6d 6f 43 20 3a 4b 4f                              moC :KO
```