

بسمه تعالی

سحر توکلی 9920613

فاطمه سلیمانی 9924803

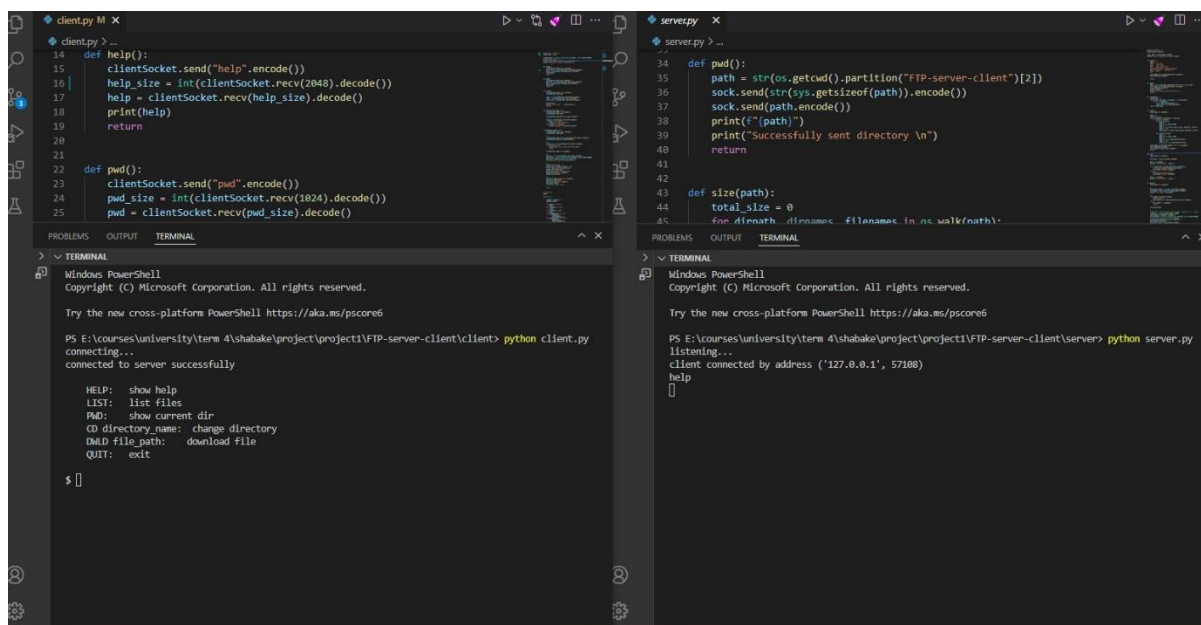
3-ب :

چرا این روند برعکس نیست یعنی کلاینت یک پورت را باز نمی کند و آن را به سرور اطلاع دهد و سرور فایل را روی آن پورت برای کلاینت ارسال کند؟

به علت اینکه در صورتی که کلاینت بتواند پورتی را باز کند مجوز آن پورت میتواند read and write باشد و به همین علت میتواند به هکر دسترسی برای نفوذ و خرابکاری دهد ولی وقتی سرور پورتی را باز کند مدیریت مجوز آن با سرور است و با مجوز read only قابلیت هک و خرابکاری را میگیرد .

اسکرین شات های برنامه:

Help:



```
client.py
def help():
    clientSocket.send("help".encode())
    help_size = int(clientSocket.recv(2048).decode())
    help = clientSocket.recv(help_size).decode()
    print(help)
    return

def pwd():
    clientSocket.send("pwd".encode())
    pwd_size = int(clientSocket.recv(1024).decode())
    pwd = clientSocket.recv(pwd_size).decode()

server.py
def pwd():
    path = str(os.getcwd().partition("FTP-server-client")[2])
    sock.send(str(sys.getsizeof(path)).encode())
    sock.send(path.encode())
    print(f"({path})")
    print("Successfully sent directory \n")
    return

def size(path):
    total_size = 0
    for dirpath, dirnames, filenames in os.walk(path):
```

Terminal 1 (client.py):

```
PS E:\courses\university\term 4\shabake\project\project1\FTP-server-client\client> python client.py
connected to server successfully

HELP: show help
LIST: list files
PWD: show current dir
CD directory_name: change directory
CMD file_path: download file
QUIT: exit

$
```

Terminal 2 (server.py):

```
PS E:\courses\university\term 4\shabake\project\project1\FTP-server-client\server> python server.py
listening...
client connected by address ('127.0.0.1', 57108)
help
[]
```

List:

The screenshot shows two VS Code windows. The left window, titled 'client.py', contains the following Python code:

```
14 def help():
15     clientSocket.send("help".encode())
16     help_size = int(clientSocket.recv(2048).decode())
17     help = clientSocket.recv(help_size).decode()
18     print(help)
19     return
20
21
22 def pwd():
23     clientSocket.send("pwd".encode())
24     pwd_size = int(clientSocket.recv(1024).decode())
25     pwd = clientSocket.recv(pwd_size).decode()
```

The terminal output for 'client.py' shows the following commands and results:

```
PS E:\courses\university\term 4\shabake\project\project1\FTP-server-client\client> python client.py
connecting...
connected to server successfully

HELP:  show help
LIST:   list files
PWD:    show current dir
CD directory_name: change directory
DWD file path:  download file
QUIT:  exit

$ list
dir1      1048601
hi.txt    22
server.py 4158

total size : 1052781

$
```

The right window, titled 'server.py', contains the following Python code:

```
34 def pwd():
35     path = str(os.getcwd().partition("FTP-server-client")[2])
36     sock.send(str(sys.getsizeof(path)).encode())
37     sock.send(path.encode())
38     print(f"{path}")
39     print("Successfully sent directory \n")
40     return
41
42
43 def size(path):
44     total_size = 0
45     for dirpath, dirnames, filenames in os.walk(path):
```

The terminal output for 'server.py' shows the following commands and results:

```
PS E:\courses\university\term 4\shabake\project\project1\FTP-server-client\server> python server.py
listening...
client connected by address ('127.0.0.1', 57108)
help
list
[]
```

The screenshot shows two VS Code windows. The left window, titled 'client.py', contains the following Python code:

```
14 def help():
15     clientSocket.send("help".encode())
16     help_size = int(clientSocket.recv(2048).decode())
17     help = clientSocket.recv(help_size).decode()
18     print(help)
19     return
20
21
22 def pwd():
23     clientSocket.send("pwd".encode())
24     pwd_size = int(clientSocket.recv(1024).decode())
25     pwd = clientSocket.recv(pwd_size).decode()
```

The terminal output for 'client.py' shows the following commands and results:

```
HELP:  show help
LIST:   list files
PWD:    show current dir
CD directory_name: change directory
DWD file path:  download file
QUIT:  exit

$ list
dir1      1048601
hi.txt    22
server.py 4158

total size : 1052781

$ dwd hi.txt
downloading...
downloaded successfully
$ pwd
\server
$ cd dir1
changed to dir1
$ list
bigFile.bin 1048576
inner       25

total size : 1048601

$
```

The right window, titled 'server.py', contains the following Python code:

```
34 def pwd():
35     path = str(os.getcwd().partition("FTP-server-client")[2])
36     sock.send(str(sys.getsizeof(path)).encode())
37     sock.send(path.encode())
38     print(f"{path}")
39     print("Successfully sent directory \n")
40     return
41
42
43 def size(path):
44     total_size = 0
45     for dirpath, dirnames, filenames in os.walk(path):
```

The terminal output for 'server.py' shows the following commands and results:

```
PS E:\courses\university\term 4\shabake\project\project1\FTP-server-client\server> python server.py
listening...
client connected by address ('127.0.0.1', 57108)
help
list
dwd
file name: hi.txt
exists
new socket for downloading on port '14047'
sending file...
completed
socket closed
pwd
\server
Successfully sent directory

cd
current path : E:\courses\university\term 4\shabake\project\project1\FTP-server-client\server
current path : E:\courses\university\term 4\shabake\project\project1\FTP-server-client\server\dir1
list
[]
```

DWLD:

The image displays a development environment with two main windows showing code and terminal output, and a file explorer window in the center.

Left Window (client.py):

```
14 def help():
15     clientSocket.send("help".encode())
16
17     help_size = int(clientSocket.recv(2048).decode())
18     help = clientSocket.recv(help_size).decode()
19     print(help)
20     return
21
22 def pwd():
23     clientSocket.send("pwd".encode())
24     pwd_size = int(clientSocket.recv(1024).decode())
25     pwd = clientSocket.recv(pwd_size).decode()
```

Center Window (File Explorer):

Path: university > term 4 > shabake > project > project1 > FTP-server-client > client

Name	Date modified	Type	Size
client.py	4/21/2022 4:59 PM	Python Source File	3 KB
hi.txt	4/21/2022 5:33 PM	Text Document	1 KB

Right Window (server.py):

```
34 def pwd():
35     path = str(os.getcwd().partition("FTP-server-client")[2])
36     sock.send(str(sys.getsizeof(path)).encode())
37     sock.send(path.encode())
38     print("path")
39     print("Successfully sent directory \n")
40     return
41
42 def size(path):
43     total_size = 0
44     for dirpath, dirnames, filenames in os.walk(path):
45         pass
```

Terminal Output (Left):

```
PS E:\courses\university\term 4\shabake\project\project1\FTP-server-client\client> python client.py
connecting...
connected to server successfully

HELP: show help
LIST: list files
PWD: show current dir
CD directory_name: change directory
DMD file_path: download file
QUIT: exit

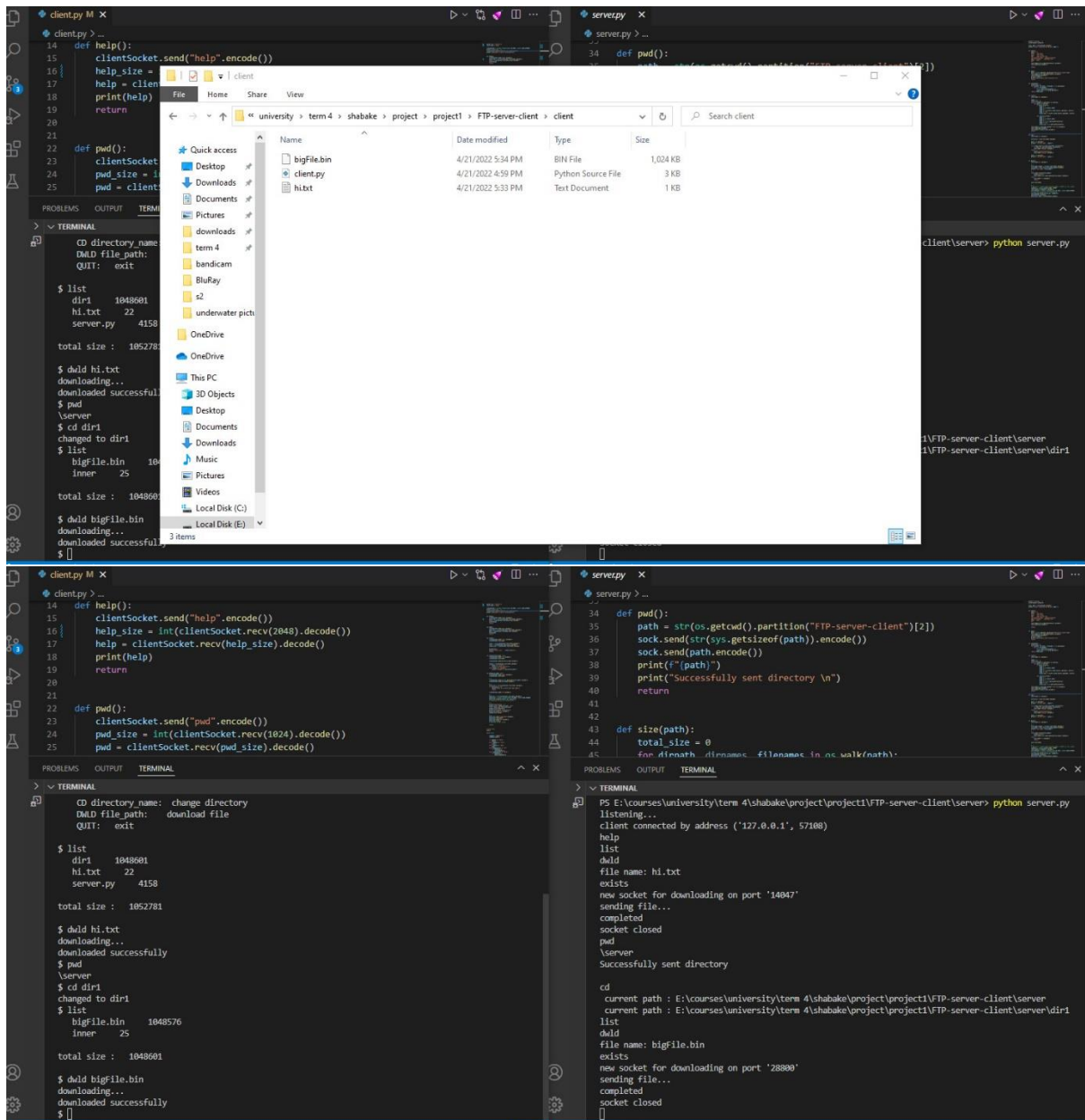
$ list
dir: 1048601
hi.txt: 22
server.py: 4158

total size: 1052781

$ dmd hi.txt
downloading...
downloaded successfully
```

Terminal Output (Right):

```
PS E:\courses\university\term 4\shabake\project\project1\FTP-server-client\server> python server.py
listening...
client connected by address ('127.0.0.1', 57180)
help
list
dmd
file name: hi.txt
exists
new socket for downloading on port '14047'
sending file...
completed
socket closed
```



PWD:

```
client.py 14 def help():
15     clientSocket.send("help".encode())
16     help_size = int(clientSocket.recv(2048).decode())
17     help = clientSocket.recv(help_size).decode()
18     print(help)
19     return
20
21
22 def pwd():
23     clientSocket.send("pwd".encode())
24     pwd_size = int(clientSocket.recv(1024).decode())
25     pwd = clientSocket.recv(pwd_size).decode()

server.py 34 def pwd():
35     path = str(os.getcwd().partition("FTP-server-client")[2])
36     sock.send(str(sys.getsizeof(path)).encode())
37     sock.send(path.encode())
38     print(f"({path})")
39     print("Successfully sent directory \n")
40     return
41
42
43 def size(path):
44     total_size = 0
45     for filename, dirname, filenames in os.walk(path):
```

Top Left Screenshot (client.py): The terminal shows the client connecting to the server successfully. It then sends a 'list' command, and the server responds with a directory listing for '1048601' containing 'hi.txt' (22 bytes) and 'server.py' (4158 bytes). The total size is 1852781. The client then sends a 'pwd' command, and the server responds with the current directory path.

Top Right Screenshot (server.py): The terminal shows the server listening on port 127.0.0.1:57100. It receives a connection from the client and sends back the directory listing for '1048601'.

Bottom Left Screenshot (client.py): The terminal shows the client sending a 'cd' command to navigate to the 'dir1' directory. It then sends a 'list' command, and the server responds with a directory listing for '1048576' containing 'bigfile.bin' (1048576 bytes) and 'inner' (25 bytes). The total size is 1048601. The client then sends a 'pwd' command, and the server responds with the current directory path.

Bottom Right Screenshot (server.py): The terminal shows the server receiving a connection from the client and sending back the directory listing for '1048576'.

Cd:

```
client.py
14 def help():
15     clientSocket.send("help".encode())
16     help_size = int(clientSocket.recv(2048).decode())
17     help = clientSocket.recv(help_size).decode()
18     print(help)
19     return
20
21
22 def pwd():
23     clientSocket.send("pwd".encode())
24     pwd_size = int(clientSocket.recv(1024).decode())
25     pwd = clientSocket.recv(pwd_size).decode()

server.py
24 def pwd():
25     path = str(os.getcwd()).partition("FTP-server-client")[2]
26     sock.send(str(sys.getsizeof(path)).encode())
27     sock.send(path.encode())
28     print("path")
29     print("Successfully sent directory \n")
30     return
31
32 def size(path):
33     total_size = 0
34     for dirpath, dirnames, filenames in os.walk(path):
```

```
PS E:\courses\university\term 4\shabake\project\project1\FTP-server-client\client> python client.py
connected to server successfully

HELP: show help
LIST: list files
PWD: show current dir
CD directory_name: change directory
DWLD file_path: download file
QUIT: exit

$ list
dir1 1048601
hi.txt 22
server.py 4158

total size : 1052781

$ dwld hi.txt
downloading...
downloaded successfully
$ pwd
/server
$ cd dir1
changed to dir1
$
```

```
PS E:\courses\university\term 4\shabake\project\project1\FTP-server-client\server> python server.py
listening...
client connected by address ('127.0.0.1', 57108)
help
list
dwld
file name: hi.txt
exists
new socket for downloading on port '14047'
sending file...
completed
socket closed
pwd
/server
Successfully sent directory

cd
current path : E:\courses\university\term 4\shabake\project\project1\FTP-server-client\server
current path : E:\courses\university\term 4\shabake\project\project1\FTP-server-client\server\dir1
list
dwld
file name: bigFile.bin
exists
new socket for downloading on port '28800'
sending file...
completed
socket closed
pwd
/server\dir1
Successfully sent directory

cd
current path : E:\courses\university\term 4\shabake\project\project1\FTP-server-client\server\dir1
current path : E:\courses\university\term 4\shabake\project\project1\FTP-server-client\server
```

```
$ list
dir1 1048601
hi.txt 22
server.py 4158

total size : 1052781

$ dwld hi.txt
downloading...
downloaded successfully
$ pwd
/server
$ cd dir1
changed to dir1
$ list
bigFile.bin 1048576
inner 25

total size : 1048601

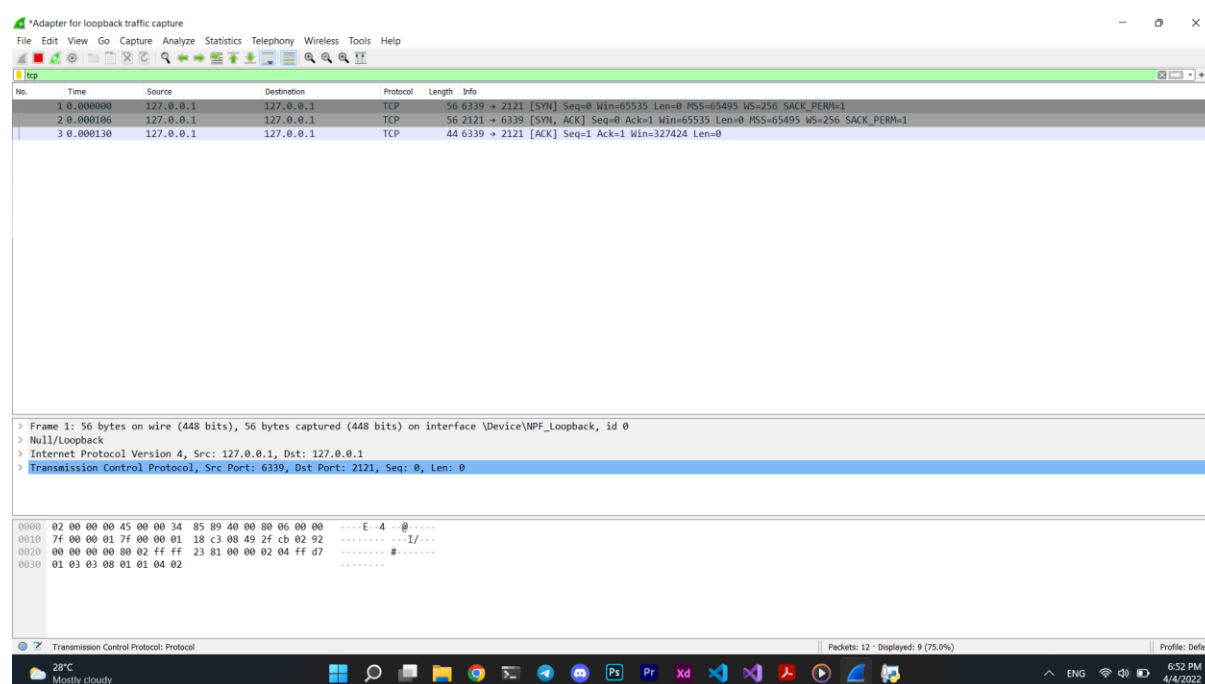
$ dwld bigFile.bin
downloading...
downloaded successfully
$ pwd
/server\dir1
$ cd ..
changed to ..
$
```

سوال : ورودی هایی برای دستورات `cd` و `dwld` که باعث شود فایل یا فولدری باز شود که در زیرشاخه های فولدر اصلی سرور قرار ندارد. (مثل فایل های سیستم عامل) این حمله چه نام دارد؟

حمله Remote file inclusion (rfi) نام دارد. بارز ترین علت این حمله سهل انگاری مدیر وب سرور میباشد که دسترسی های مناسب برای پوشه ریشه در نظر نگرفته است مهاجم با وارد کردن کاراکترهای مشخصی به `parent directory` دسترسی پیدا میکند.

WireShark:

4.1



4.2

ایا این پروتکل محدودیتی برای برای اندازه بسته ها دارد؟ فایل های بزرگ چگونه توسط سوکت ارسال میشوند؟

بله محدودیت دارد و برای ارسال فایل های بزرگ ان هارا به چند قسمت تقسیم میکند و به صورت تکه تکه میفرستد. اندازه استاندارد بسته حداقل 20 بایت و حداکثر 60 بایت است.

برای دانلود این فایل چند بسته فرستاده است؟

حدود 50 بسته (در اسکرین شات از 20 تا 69)

Adapter for loopback traffic capture

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp

No.	Time	Source	Destination	Protocol	Length	Info
19	40.059314	127.0.0.1	127.0.0.1	TCP	51	2121 → 6339 [PSH, ACK] Seq=180 Ack=11 Win=2161152 Len=7
20	40.059416	127.0.0.1	127.0.0.1	TCP	44	6339 → 2121 [ACK] Seq=11 Ack=187 Win=327168 Len=0
21	58.454429	127.0.0.1	127.0.0.1	TCP	48	6339 → 2121 [PSH, ACK] Seq=11 Ack=187 Win=327168 Len=4
22	58.454542	127.0.0.1	127.0.0.1	TCP	44	2121 → 6339 [ACK] Seq=187 Ack=15 Win=2161152 Len=0
23	58.455396	127.0.0.1	127.0.0.1	TCP	46	2121 → 6339 [PSH, ACK] Seq=187 Ack=15 Win=2161152 Len=2
24	58.455400	127.0.0.1	127.0.0.1	TCP	44	6339 → 2121 [ACK] Seq=15 Ack=189 Win=327168 Len=0
25	58.455587	127.0.0.1	127.0.0.1	TCP	46	6339 → 2121 [PSH, ACK] Seq=15 Ack=189 Win=327168 Len=2
26	58.455685	127.0.0.1	127.0.0.1	TCP	44	2121 → 6339 [ACK] Seq=189 Ack=17 Win=2161152 Len=0
27	58.455779	127.0.0.1	127.0.0.1	TCP	55	6339 → 2121 [PSH, ACK] Seq=17 Ack=189 Win=327168 Len=11
28	58.455818	127.0.0.1	127.0.0.1	TCP	44	2121 → 6339 [ACK] Seq=189 Ack=28 Win=2161152 Len=0
29	58.458563	127.0.0.1	127.0.0.1	TCP	51	2121 → 6339 [PSH, ACK] Seq=189 Ack=28 Win=2161152 Len=7
30	58.458660	127.0.0.1	127.0.0.1	TCP	44	6339 → 2121 [ACK] Seq=28 Ack=196 Win=327168 Len=0
31	58.458818	127.0.0.1	127.0.0.1	TCP	46	6339 → 2121 [PSH, ACK] Seq=28 Ack=196 Win=327168 Len=2
32	58.458913	127.0.0.1	127.0.0.1	TCP	44	2121 → 6339 [ACK] Seq=196 Ack=30 Win=2161152 Len=0
33	58.459099	127.0.0.1	127.0.0.1	TCP	48	2121 → 6339 [PSH, ACK] Seq=196 Ack=30 Win=2161152 Len=4
34	58.459194	127.0.0.1	127.0.0.1	TCP	44	6339 → 2121 [ACK] Seq=30 Ack=200 Win=327168 Len=0
35	58.459720	127.0.0.1	127.0.0.1	TCP	56	6341 → 9973 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
36	58.459892	127.0.0.1	127.0.0.1	TCP	56	9973 → 6341 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM=1
37	58.459954	127.0.0.1	127.0.0.1	TCP	44	6341 → 9973 [ACK] Seq=1 Ack=1 Win=327424 Len=0
38	58.463955	127.0.0.1	127.0.0.1	TCP	65539	9973 → 6341 [ACK] Seq=1 Ack=1 Win=2161152 Len=65495
39	58.463224	127.0.0.1	127.0.0.1	TCP	65539	9973 → 6341 [ACK] Seq=65496 Ack=1 Win=2161152 Len=65495
40	58.463346	127.0.0.1	127.0.0.1	TCP	65539	9973 → 6341 [ACK] Seq=130991 Ack=1 Win=2161152 Len=65495

> Frame 20: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface \Device\NPF_{Loopback}, id 0

> Null/Loopback

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

> Transmission Control Protocol, Src Port: 6339, Dst Port: 2121, Seq: 11, Ack: 187, Len: 0

0000 02 00 00 00 45 00 00 28 85 99 40 00 80 06 00 00E-@.....
0010 7f 00 00 01 7f 00 00 01 18 c3 08 49 2f cb 02 9dI/-...
0020 8e c8 ec 8b 50 10 04 fe de 00 00 00P.....

wireShark_NPF_LoopbackOAV1.pcapng Packets: 69 · Displayed: 66 (95.7%) · Dropped: 0 (0.0%) Profile: Default

28°C Mostly cloudy 6:53 PM 4/4/2022

Adapter for loopback traffic capture

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp

No.	Time	Source	Destination	Protocol	Length	Info
40	58.463346	127.0.0.1	127.0.0.1	TCP	65539	9973 → 6341 [ACK] Seq=130991 Ack=1 Win=2161152 Len=65495
41	58.463495	127.0.0.1	127.0.0.1	TCP	65539	9973 → 6341 [ACK] Seq=196486 Ack=1 Win=2161152 Len=65495
42	58.463945	127.0.0.1	127.0.0.1	TCP	44	6341 → 9973 [ACK] Seq=1 Ack=261981 Win=327424 Len=0
43	58.464089	127.0.0.1	127.0.0.1	TCP	65539	9973 → 6341 [ACK] Seq=261981 Ack=1 Win=2161152 Len=65495
44	58.464172	127.0.0.1	127.0.0.1	TCP	65539	9973 → 6341 [ACK] Seq=327476 Ack=1 Win=2161152 Len=65495
45	58.464280	127.0.0.1	127.0.0.1	TCP	65539	9973 → 6341 [ACK] Seq=392971 Ack=1 Win=2161152 Len=65495
46	58.464399	127.0.0.1	127.0.0.1	TCP	65539	9973 → 6341 [ACK] Seq=458466 Ack=1 Win=2161152 Len=65495
47	58.464756	127.0.0.1	127.0.0.1	TCP	44	6341 → 9973 [ACK] Seq=1 Ack=523961 Win=327424 Len=0
48	58.464873	127.0.0.1	127.0.0.1	TCP	65539	9973 → 6341 [ACK] Seq=523961 Ack=1 Win=2161152 Len=65495
49	58.464990	127.0.0.1	127.0.0.1	TCP	65539	9973 → 6341 [ACK] Seq=589456 Ack=1 Win=2161152 Len=65495
50	58.465122	127.0.0.1	127.0.0.1	TCP	65539	9973 → 6341 [ACK] Seq=654951 Ack=1 Win=2161152 Len=65495
51	58.465247	127.0.0.1	127.0.0.1	TCP	65539	9973 → 6341 [ACK] Seq=720446 Ack=1 Win=2161152 Len=65495
52	58.465609	127.0.0.1	127.0.0.1	TCP	44	6341 → 9973 [ACK] Seq=1 Ack=785941 Win=327424 Len=0
53	58.465708	127.0.0.1	127.0.0.1	TCP	65539	9973 → 6341 [ACK] Seq=785941 Ack=1 Win=2161152 Len=65495
54	58.465814	127.0.0.1	127.0.0.1	TCP	65539	9973 → 6341 [ACK] Seq=851436 Ack=1 Win=2161152 Len=65495
55	58.466196	127.0.0.1	127.0.0.1	TCP	65539	9973 → 6341 [ACK] Seq=916931 Ack=1 Win=2161152 Len=65495
56	58.466304	127.0.0.1	127.0.0.1	TCP	65539	9973 → 6341 [ACK] Seq=982426 Ack=1 Win=2161152 Len=65495
57	58.466399	127.0.0.1	127.0.0.1	TCP	700	9973 → 6341 [PSH, ACK] Seq=1047921 Ack=1 Win=2161152 Len=656
58	58.466718	127.0.0.1	127.0.0.1	TCP	44	6341 → 9973 [ACK] Seq=1 Ack=1048577 Win=327424 Len=0
59	58.468277	127.0.0.1	127.0.0.1	TCP	46	6341 → 9973 [PSH, ACK] Seq=1 Ack=1048577 Win=327424 Len=2
60	58.468363	127.0.0.1	127.0.0.1	TCP	44	9973 → 6341 [ACK] Seq=1048577 Ack=3 Win=2161152 Len=0
61	58.470320	127.0.0.1	127.0.0.1	TCP	53	6341 → 9973 [PSH, ACK] Seq=3 Ack=1048577 Win=327424 Len=9

> Frame 20: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface \Device\NPF_{Loopback}, id 0

> Null/Loopback

> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

> Transmission Control Protocol, Src Port: 6339, Dst Port: 2121, Seq: 11, Ack: 187, Len: 0

0000 02 00 00 00 45 00 00 28 85 99 40 00 80 06 00 00E-@.....
0010 7f 00 00 01 7f 00 00 01 18 c3 08 49 2f cb 02 9dI/-...
0020 8e c8 ec 8b 50 10 04 fe de 00 00 00P.....

wireShark_NPF_LoopbackOAV1.pcapng Packets: 69 · Displayed: 66 (95.7%) · Dropped: 0 (0.0%) Profile: Default

28°C Mostly cloudy 6:54 PM 4/4/2022

Wireshark interface showing a packet capture on the loopback interface \Device\NPF_{...}. The packet list shows a series of TCP connections from 127.0.0.1 to 127.0.0.1. The packet details pane shows the selected packet (No. 69) as an Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1, and a Transmission Control Protocol, Src Port: 6339, Dst Port: 2121, Seq: 11, Ack: 187, Len: 0. The packet bytes pane shows the raw data in hexadecimal and ASCII.

Ngrok:

Wireshark interface showing a packet capture on the loopback interface \Device\NPF_{...}. The packet list shows a series of TCP connections from 127.0.0.1 to 127.0.0.1. The packet details pane shows the selected packet (No. 81) as an Internet Protocol Version 6, Src: fe80::c5ad:ea46:388:815c, Dst: ff02::fb. The packet bytes pane shows the raw data in hexadecimal and ASCII.

Wireshark packet capture analysis showing a list of network packets. The interface includes a packet list, packet details, and packet bytes panes. The packet list shows various TCP and UDP packets, including a large packet (No. 129) with a length of 36 bytes. The packet details pane shows the structure of the selected packet, including the Ethernet II header, Internet Protocol Version 4 header, and Transmission Control Protocol header. The packet bytes pane shows the raw data of the selected packet.

No.	Time	Source	Destination	Protocol	Length	Info
105	180.951939	127.0.0.1	127.0.0.1	TCP	51	12345 → 62142 [PSH, ACK] Seq=180 Ack=8 Win=2619648 Len=7
106	180.952000	127.0.0.1	127.0.0.1	TCP	44	62142 → 12345 [ACK] Seq=8 Ack=187 Win=2619392 Len=0
107	115.959002	127.0.0.1	127.0.0.1	TCP	45	[TCP Keep-Alive] 62142 → 12345 [ACK] Seq=7 Ack=187 Win=2619392 Len=1
108	115.959148	127.0.0.1	127.0.0.1	TCP	56	[TCP Keep-Alive ACK] 12345 → 62142 [ACK] Seq=187 Ack=8 Win=2619648 Len=0 SLE=7 SRE=8
109	116.983072	127.0.0.1	127.0.0.1	TCP	46	62142 → 12345 [PSH, ACK] Seq=8 Ack=187 Win=2619392 Len=2
110	116.983166	127.0.0.1	127.0.0.1	TCP	44	12345 → 62142 [ACK] Seq=187 Ack=10 Win=2619648 Len=0
111	116.983792	127.0.0.1	127.0.0.1	TCP	46	12345 → 62142 [PSH, ACK] Seq=187 Ack=10 Win=2619648 Len=2
112	116.983775	127.0.0.1	127.0.0.1	TCP	44	62142 → 12345 [ACK] Seq=10 Ack=189 Win=2619392 Len=0
113	117.231484	127.0.0.1	127.0.0.1	TCP	48	62142 → 12345 [PSH, ACK] Seq=10 Ack=189 Win=2619392 Len=4
114	117.231584	127.0.0.1	127.0.0.1	TCP	44	12345 → 62142 [ACK] Seq=189 Ack=14 Win=2619648 Len=0
115	117.233132	127.0.0.1	127.0.0.1	TCP	51	12345 → 62142 [PSH, ACK] Seq=189 Ack=14 Win=2619648 Len=7
116	117.233227	127.0.0.1	127.0.0.1	TCP	44	62142 → 12345 [ACK] Seq=14 Ack=196 Win=2619392 Len=0
117	130.758082	127.0.0.1	127.0.0.1	TCP	48	62142 → 12345 [PSH, ACK] Seq=14 Ack=196 Win=2619392 Len=4
118	130.758982	127.0.0.1	127.0.0.1	TCP	44	12345 → 62142 [ACK] Seq=196 Ack=18 Win=2619648 Len=0
119	130.759661	127.0.0.1	127.0.0.1	TCP	46	12345 → 62142 [PSH, ACK] Seq=196 Ack=18 Win=2619648 Len=2
120	130.759743	127.0.0.1	127.0.0.1	TCP	44	62142 → 12345 [ACK] Seq=18 Ack=198 Win=2619392 Len=0
121	130.842788	127.0.0.1	127.0.0.1	TCP	46	12345 → 62142 [PSH, ACK] Seq=198 Ack=18 Win=2619648 Len=2
122	130.842889	127.0.0.1	127.0.0.1	TCP	44	62142 → 12345 [ACK] Seq=18 Ack=200 Win=2619392 Len=0
123	130.843015	127.0.0.1	127.0.0.1	TCP	87	12345 → 62142 [PSH, ACK] Seq=200 Ack=18 Win=2619648 Len=43
124	130.843094	127.0.0.1	127.0.0.1	TCP	44	62142 → 12345 [ACK] Seq=18 Ack=243 Win=2619392 Len=0
125	130.843221	127.0.0.1	127.0.0.1	TCP	51	12345 → 62142 [PSH, ACK] Seq=243 Ack=18 Win=2619648 Len=7
126	130.843306	127.0.0.1	127.0.0.1	TCP	44	62142 → 12345 [ACK] Seq=18 Ack=250 Win=2619392 Len=0
127	130.419514	192.168.1.103	239.255.255.250	SSDP	207	M-SEARCH * HTTP/1.1
128	130.419743	192.168.1.103	239.255.255.250	SSDP	207	M-SEARCH * HTTP/1.1
129	139.499624	192.168.1.103	224.0.0.252	IGMPv2	36	Membership Report group 224.0.0.252
130	140.433849	192.168.1.103	239.255.255.250	SSDP	207	M-SEARCH * HTTP/1.1
131	140.433116	192.168.1.103	239.255.255.250	SSDP	207	M-SEARCH * HTTP/1.1
132	141.443333	192.168.1.103	239.255.255.250	SSDP	207	M-SEARCH * HTTP/1.1
133	141.443463	192.168.1.103	239.255.255.250	SSDP	207	M-SEARCH * HTTP/1.1

Checksum: 0x651b [Unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
[Timestamps]
[Time since first frame in this TCP stream: 19.638331000 seconds]
[Time since previous frame in this TCP stream: 0.000091000 seconds]
> [Seq/ACK analysis]

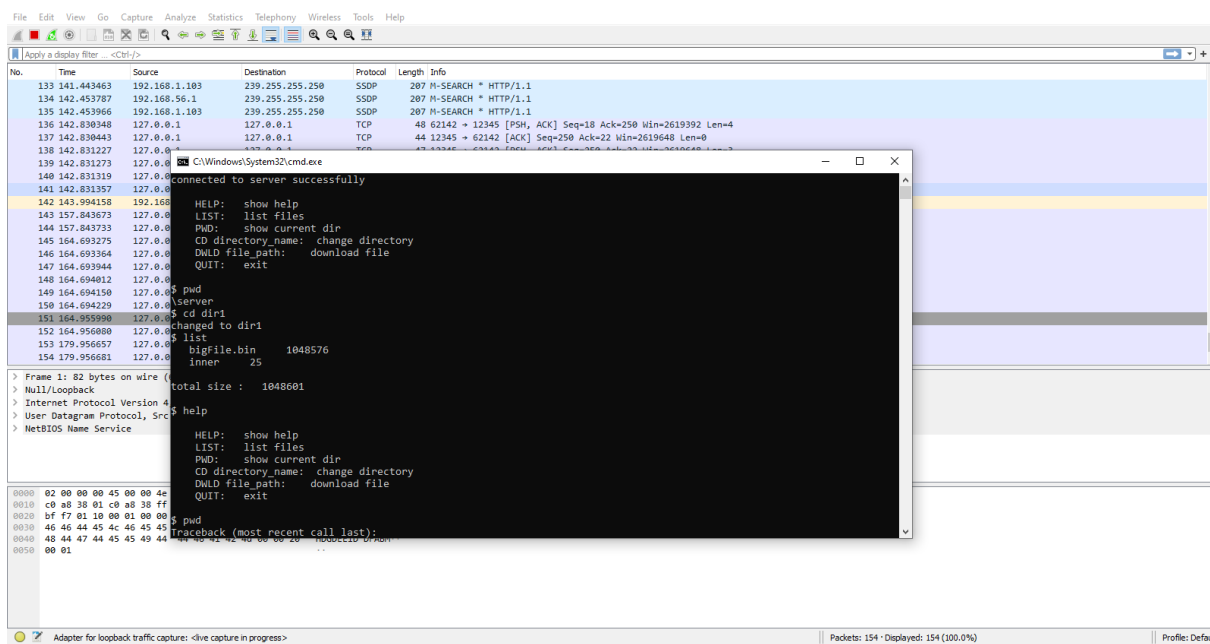
0000 02 00 00 00 45 00 00 28 73 c2 40 00 00 06 00 00E..(s @.....
0010 7f 00 00 01 7f 00 00 01 f2 be 30 39 e2 6c 36 e0-09.16..
0020 11 fa d6 7f 50 10 27 f8 65 1b 00 00P..@.....

Wireshark packet capture analysis showing a list of network packets. The interface includes a packet list, packet details, and packet bytes panes. The packet list shows various DNS and HTTP packets, including a large packet (No. 89) with a length of 12345 bytes. The packet details pane shows the structure of the selected packet, including the Ethernet II header, Internet Protocol Version 4 header, and Transmission Control Protocol header. The packet bytes pane shows the raw data of the selected packet.

No.	Time	Source	Destination	Protocol	Length	Info
82	70.837472	192.168.56.1	192.168.56.255	NBNS	82	Name query NB DESKTOP-7764H3P<ic>
83	70.977796	192.168.56.1	192.168.56.255	NBNS	82	Name query NB CLIENT<00>
84	70.978128	192.168.1.103	192.168.1.255	NBNS	82	Name query NB CLIENT<00>
85	71.597163	192.168.1.103	192.168.1.255	NBNS	82	Name query NB DESKTOP-7764H3P<ic>
86	72.362289	192.168.1.103	192.168.1.255	NBNS	82	Name query NB DESKTOP-7764H3P<ic>
87	73.124459	192.168.1.103	192.168.1.255	NBNS	82	Name query NB DESKTOP-7764H3P<ic>
88	81.004743	192.168.1.103	192.168.1.255	NBNS	82	Name query NB DESKTOP-7764H3P<ic>
89	81.004812	192.168.1.103	192.168.1.255	NBNS	82	Name query NB DESKTOP-7764H3P<ic>
90	81.313503	192.168.1.103	192.168.1.255	NBNS	82	Name query NB DESKTOP-7764H3P<ic>
91	81.313808	192.168.1.103	192.168.1.255	NBNS	82	Name query NB DESKTOP-7764H3P<ic>
92	81.313944	192.168.1.103	192.168.1.255	NBNS	82	Name query NB DESKTOP-7764H3P<ic>
93	81.314542	192.168.1.103	192.168.1.255	NBNS	82	Name query NB DESKTOP-7764H3P<ic>
94	81.314617	192.168.1.103	192.168.1.255	NBNS	82	Name query NB DESKTOP-7764H3P<ic>
95	81.316026	192.168.1.103	192.168.1.255	NBNS	82	Name query NB DESKTOP-7764H3P<ic>
96	81.316194	192.168.1.103	192.168.1.255	NBNS	82	Name query NB DESKTOP-7764H3P<ic>
97	81.316173	192.168.1.103	192.168.1.255	NBNS	82	Name query NB DESKTOP-7764H3P<ic>
98	81.316226	192.168.1.103	192.168.1.255	NBNS	82	Name query NB DESKTOP-7764H3P<ic>
99	96.320808	192.168.1.103	192.168.1.255	NBNS	82	Name query NB DESKTOP-7764H3P<ic>
100	96.320937	192.168.1.103	192.168.1.255	NBNS	82	Name query NB DESKTOP-7764H3P<ic>
101	100.950967	192.168.1.103	192.168.1.255	NBNS	82	Name query NB DESKTOP-7764H3P<ic>
102	100.951050	192.168.1.103	192.168.1.255	NBNS	82	Name query NB DESKTOP-7764H3P<ic>
103	100.951743	192.168.1.103	192.168.1.255	NBNS	82	Name query NB DESKTOP-7764H3P<ic>

Flags: 0x002 [SYN]
Window: 65535
[Calculated window size: 65535]
Checksum: 0x0210 [Unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0
Options: (12 bytes)
[Timestamps]

0000 18 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030 18 b9 37 3f 00 00 00 00 00 02 ff ff 02 10 00 007.....
0040 02 04 ff c3 01 03 03 00 01 01 04 02



**شرایط دانلود فایل ها از طریق ایجاد کانال داده با ngrok را ایجاد کنید آیا این کار ممکن است ؟ اگر بله
چطور و اگر نه چه مشکلی وجود دارد؟**

راهکار خود را برای حل این مشکل توضیح دهید

خیر ممکن نیست چون در حالت لوکال برای دانلود نیاز به ایجاد یک کانال با پورت رندوم داریم که سرور آن را انتخاب میکند و برای کلاینت میفرستد . حال در شرایطی که از ngrok استفاده میکنیم برای اتصال کلاینت به سرور ما پورت سرور را به ngrok میدادیم و ngrok متناسب با آن به ما یک پورتنی میداد که آن را به صورت دستی در کلاینت وارد میکردیم.

در شرایط لوکال برای دانلود ، یک کانال با پورت رندوم در سمت سرور ایجاد میشد و سرور آن را به کلاینت میفرستاد و با این شرایط اگر بخواهیم دانلود را به وسیله ی ngrok ایجاد کنیم باید به نحوی آن پورت رندوم را که در سمت سرور ایجاد میشود به ngrok بدهیم و ngrok هم به تناسب آن یک پورت به ما بدهد و ما آن را به کلاینت بدهیم تا این ارتباط برقرار شود و فرایند دانلود انجام شود .