Assignment 2: Encrypted File Transfer

Version Date: February 18, 2025

Prerequisites:

Before starting the project, ensure you have done the following.

- Download and install Wireshark. Make sure you can use it to examine the packet transfers. There is plenty of help online.
- Review the basic client-server programming and other Operating System topics.
- Review public-key encryption and RSA.
- Review Block Cipher, including AES.
- Have a Python IDE ready for the coding. The suggestion is PyCharm.

The assignment is designed to make it easier for students by providing the structure of the solution.

Introduction

Modern secure communication relies on public key exchange protocols and stream/block ciphers. These two building blocks ensure that encryption keys are exchanged securely before data encryption. Once two entities exchange keys, they establish a secure "tunnel", allowing encrypted communication.

This homework will explore secure file transfer by implementing an encrypted file transfer system over sockets.

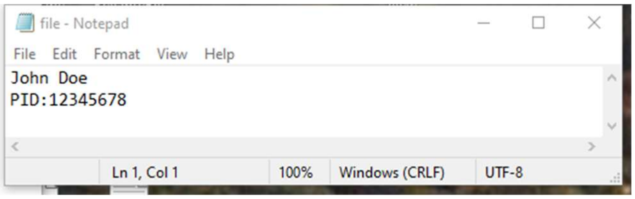
Part 1: Plaintext File Transfer

Before securing our communication, let's examine an insecure version to understand potential vulnerabilities. These programs will help you set up your local environment for simulating client-server programming.

This section provides an insecure version of the script. Please download client1.py and server1 from the website. They are the codes for unencrypted transmission.

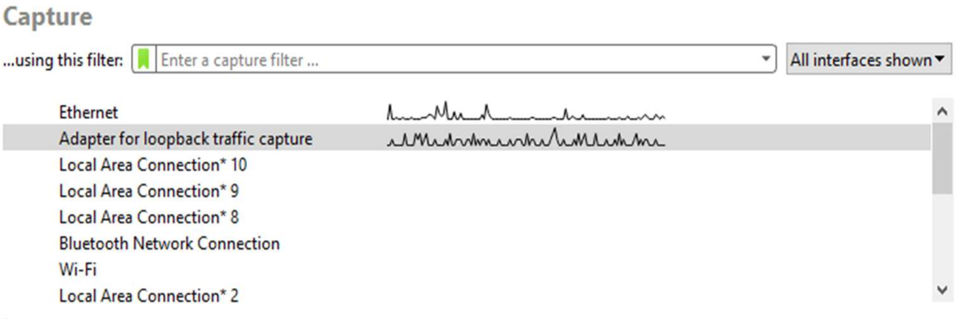
To simulate the traffic on your local machine, open two Python IDE windows (or two terminal sessions), one for the server and one for the client. Additionally, we will inspect the network traffic using Wireshark, a network protocol analyzer. Download (https://www.wireshark.org/download.html) and install Wireshark from its official website if you haven't already. This will allow you to capture and analyze the data packets transmitted during the file transfer. The entire task flow is as follows:

1. Create a MyFile.txt file. Please put some text in it. It does not matter what, but try to make it unique and contain no private information.



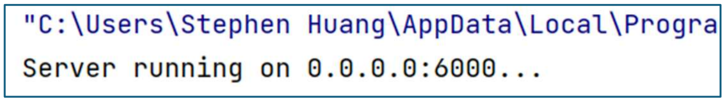
```
John Doe
PID:12345678
```

2. Start capturing traffic using Wireshark.



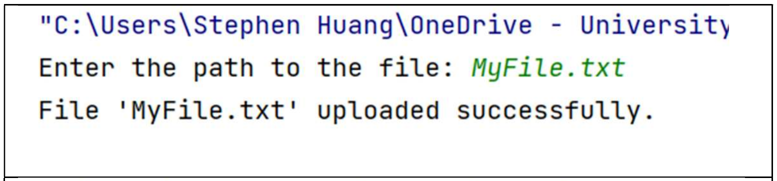
Open the Wireshark software. Select "Loopback" for Windows or "lo0" for macOS/Linux in the capture interface. Double-click the interface, and Wireshark will start capturing traffic.

3. Run the server.

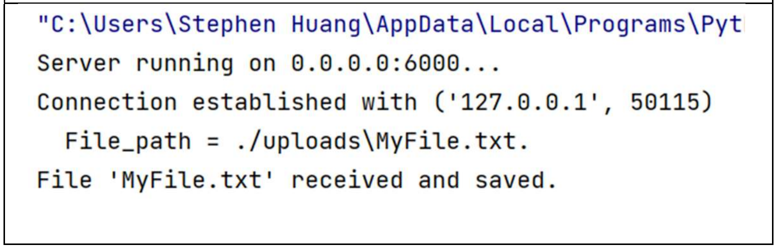


```
"C:\Users\Stephen Huang\AppData\Local\Progra
Server running on 0.0.0.0:6000...
```

4. Run the client, sending the file's metadata and content to the server. If the file is located in the same directory as the code, you may use the file name directly without the path.

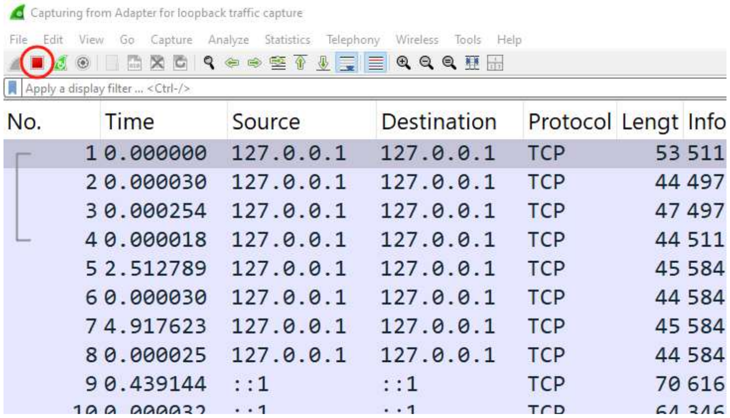


```
"C:\Users\Stephen Huang\OneDrive - University
Enter the path to the file: MyFile.txt
File 'MyFile.txt' uploaded successfully.
```



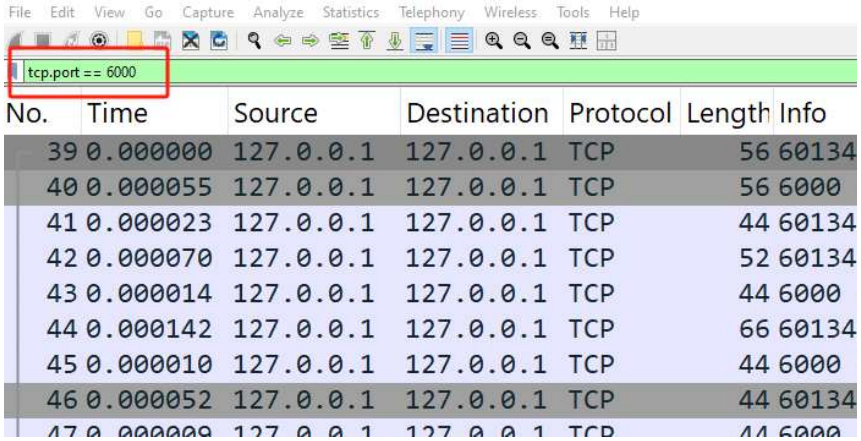
```
"C:\Users\Stephen Huang\AppData\Local\Programs\Pyt
Server running on 0.0.0.0:6000...
Connection established with ('127.0.0.1', 50115)
  File_path = ./uploads\MyFile.txt.
File 'MyFile.txt' received and saved.
```

5. After the file is successfully transferred, stop the traffic capture by clicking the red square in the top-left corner of the Wireshark window.


Capturing from Adapter for loopback traffic capture
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Lengt	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	53	511
2	0.000030	127.0.0.1	127.0.0.1	TCP	44	497
3	0.000254	127.0.0.1	127.0.0.1	TCP	47	497
4	0.000018	127.0.0.1	127.0.0.1	TCP	44	511
5	2.512789	127.0.0.1	127.0.0.1	TCP	45	584
6	0.000030	127.0.0.1	127.0.0.1	TCP	44	584
7	4.917623	127.0.0.1	127.0.0.1	TCP	45	584
8	0.000025	127.0.0.1	127.0.0.1	TCP	44	584
9	0.439144	::1	::1	TCP	70	616
10	0.000032	::1	::1	TCP	64	346

6. Inspect the captured traffic. Type tcp.port == 6000 in the filter box to locate all relevant network packets.

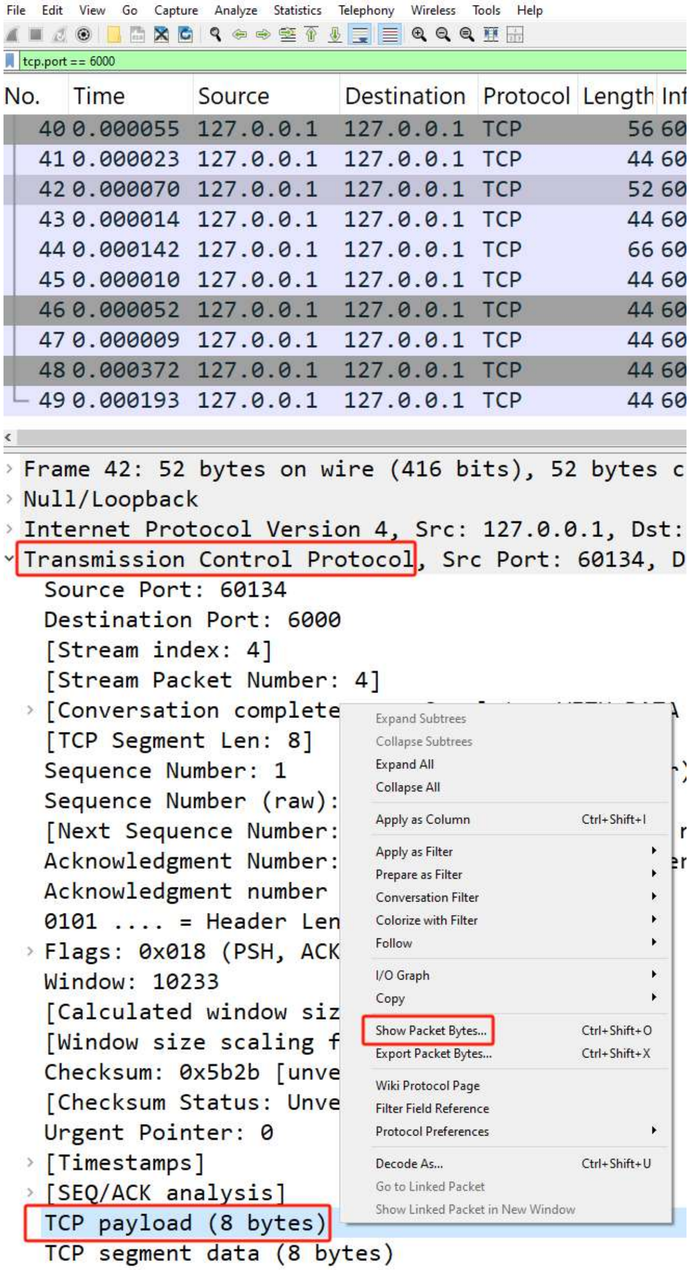

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

tcp.port == 6000

No.	Time	Source	Destination	Protocol	Length	Info
39	0.000000	127.0.0.1	127.0.0.1	TCP	56	60134
40	0.000055	127.0.0.1	127.0.0.1	TCP	56	6000
41	0.000023	127.0.0.1	127.0.0.1	TCP	44	60134
42	0.000070	127.0.0.1	127.0.0.1	TCP	52	60134
43	0.000014	127.0.0.1	127.0.0.1	TCP	44	6000
44	0.000142	127.0.0.1	127.0.0.1	TCP	66	60134
45	0.000010	127.0.0.1	127.0.0.1	TCP	44	6000
46	0.000052	127.0.0.1	127.0.0.1	TCP	44	60134
47	0.000009	127.0.0.1	127.0.0.1	TCP	44	6000

Click through each packet and identify those that contain a **TCP payload** in the **Transmission Control Protocol** section. Right-click on the **TCP payload** and select **"Show Packet Bytes."** As shown in the example on the next page, the payload content of this packet represents the metadata (filename) of the transferred file.

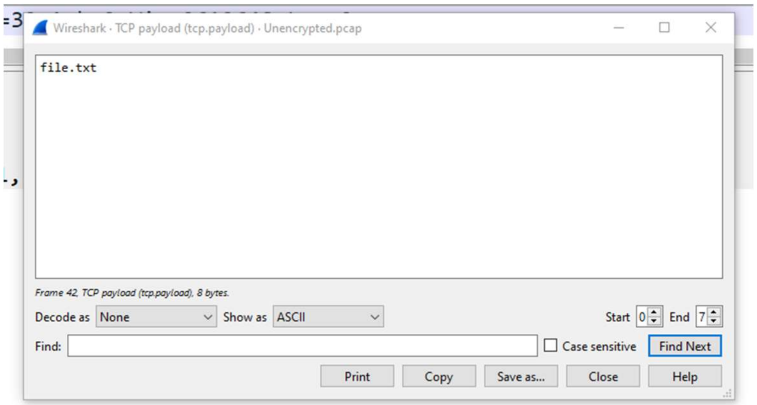
7. Take a screenshot of the **payload content** for all the remaining **payload packets**.



```
tcp.port == 6000
```

```
No.  Time     Source    Destination Protocol Length Inf
  40 0.000055 127.0.0.1 127.0.0.1   TCP          56 60
  41 0.000023 127.0.0.1 127.0.0.1   TCP          44 60
  42 0.000070 127.0.0.1 127.0.0.1   TCP          52 60
  43 0.000014 127.0.0.1 127.0.0.1   TCP          44 60
  44 0.000142 127.0.0.1 127.0.0.1   TCP          66 60
  45 0.000010 127.0.0.1 127.0.0.1   TCP          44 60
  46 0.000052 127.0.0.1 127.0.0.1   TCP          44 60
  47 0.000009 127.0.0.1 127.0.0.1   TCP          44 60
  48 0.000372 127.0.0.1 127.0.0.1   TCP          44 60
  49 0.000193 127.0.0.1 127.0.0.1   TCP          44 60
```

```
> Frame 42: 52 bytes on wire (416 bits), 52 bytes c
> Null/Loopback
> Internet Protocol Version 4, Src: 127.0.0.1, Dst:
v Transmission Control Protocol, Src Port: 60134, D
    Source Port: 60134
    Destination Port: 6000
    [Stream index: 4]
    [Stream Packet Number: 4]
  > [Conversation complete
    [TCP Segment Len: 8]
    Sequence Number: 1
    Sequence Number (raw):
    [Next Sequence Number:
    Acknowledgment Number:
    Acknowledgment number
    0101 .... = Header Len
  > Flags: 0x018 (PSH, ACK
    Window: 10233
    [Calculated window siz
    [Window size scaling f
    Checksum: 0x5b2b [unve
    [Checksum Status: Unve
    Urgent Pointer: 0
  > [Timestamps]
  > [SEQ/ACK analysis]
    TCP payload (8 bytes)
    TCP segment data (8 bytes)
```



Wireshark · TCP payload (tcp.payload) · Unencrypted.pcap

```
file.txt
```

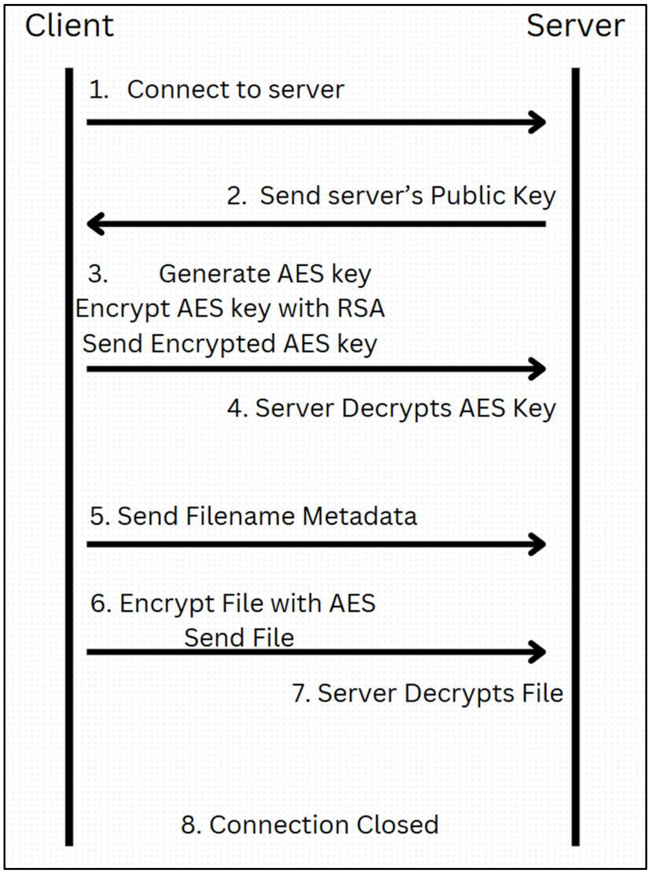
Frame 42, TCP payload (tcp.payload), 8 bytes.

Decode as: None
Show as: ASCII
Start: 0
End: 7

Part 2: Encrypted File Transfer

To prevent eavesdropping, let's secure our system using public key exchange protocols and stream/block ciphers. We will use **RSA** for **public key exchange** and **AES** as the **encryption cipher**.

1. Download the skeleton script for the **encrypted version** from the website and complete all the blanks to ensure the scripts perform their tasks correctly. The protocol between the client and the server is in the figure on the right.
2. Capture the network traffic as done previously for the insecure version and take screenshots of all **payload content**.



For the encrypted version, the files are called client2.py and server2.py. Some of the codes were removed from the programs, and the omitted code is replaced with:

```
## BEGIN
#
## END
```

Your job is to fill in the line between the BEGIN and the END. Keep these two lines so the TA can find them easily. The amount of space is approximately the length of the code I expected. However, there are always multiple ways of doing the same task.

Part 3: Deliverables

1. **Python Codes**: Submit the two Python files (client2.py and server2.py). The TAs will test your code using their test files. Ensure your code contains additional print statements (after each significant step) to help the TAs easily trace your progress.)
2. **Wireshark Logs**: Provide two copies of the Wireshark log in pcap format. You may have several logs, but please submit only one for each scenario: HWK2E.pcap for the encrypted file transfer and HWK2U.pcap for the unencrypted file transfer.
3. **PDF Report**: Please include a PDF report (HWK2.pdf) on your Wireshark log examination. Please include screenshots showing that one could see the plain text when running the unencrypted version and the encoded message for the encrypted experiment. You may include comments on how the assignment can be made more interesting.