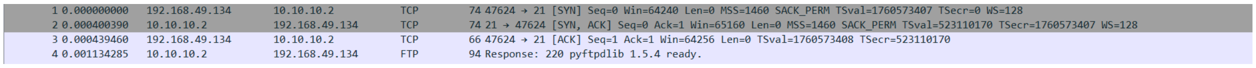
CSI PROBLEM STATEMENT SOLUTIONS

1. WIRESHARK CASE-STUDIES

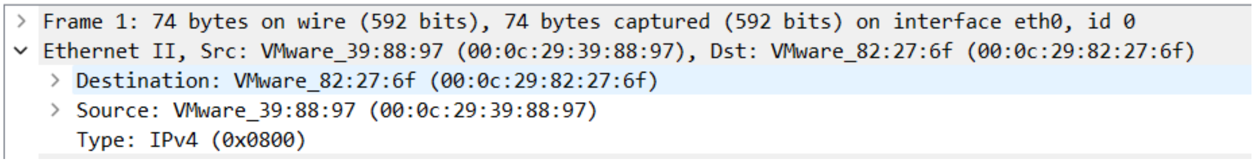
CASE STUDY 1:



```
1 0.000000000   192.168.49.134     10.10.10.2         TCP        74 47624 → 21 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1760573407 TSecr=0 WS=128
2 0.000400390   10.10.10.2         192.168.49.134     TCP        74 21 → 47624 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=523110170 TSecr=1760573407 WS=128
3 0.000439460   192.168.49.134     10.10.10.2         TCP        66 47624 → 21 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1760573408 TSecr=523110170
4 0.001134285   10.10.10.2         192.168.49.134     FTP        94 Response: 220 pyftpdlib 1.5.4 ready.
```

- Sequence nos. 1 and 2 depicts the beginning(initiating phase) of connection between the host and destination by sending and receiving data packets. Also this is a Syn Flood method. Here the destination(target) is **10.10.10.2** and the host(attacker) is **192.168.49.134**. The Protocol used here is TCP(/IP) - Transmission Control Protocol. Sequence no. 2 shows positive reply back from our target's address given as 21 → 47624.
- Sequence number 3 and 4 ensure the firm and positive connection between the host and the destination. Sequence no. 4 shows that Port no. 21 is open as the protocol in this result statement is **FTP**(**FTP** server uses port - 21 by default). Also it shows the FTP server "**pyftpdlib 1.5.4**" ready for use.

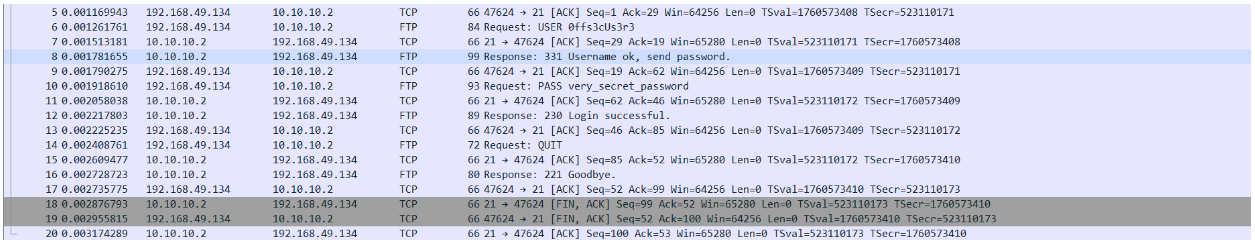
NOTE *- To confirm if a port is actually open, we should use different techniques like* ***Nmap(TOOL)*** *and* ***ZAP Attack Tool(an open source 3rd party app - OWASP ZAP)****.*



```
> Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface eth0, id 0
v Ethernet II, Src: VMware_39:88:97 (00:0c:29:39:88:97), Dst: VMware_82:27:6f (00:0c:29:82:27:6f)
  > Destination: VMware_82:27:6f (00:0c:29:82:27:6f)
  > Source: VMware_39:88:97 (00:0c:29:39:88:97)
    Type: IPv4 (0x0800)
```

- Also to mention both the host's and destination's OS is installed on Virtual Machines. Wireshark helps in extracting MAC addresses as well.
  - Destination's(Target's) MAC Address : (00:0c:29:82:27:6f)

    - Host’s(Attacker’s) MAC Address : (00:0c:29:39:88:97)

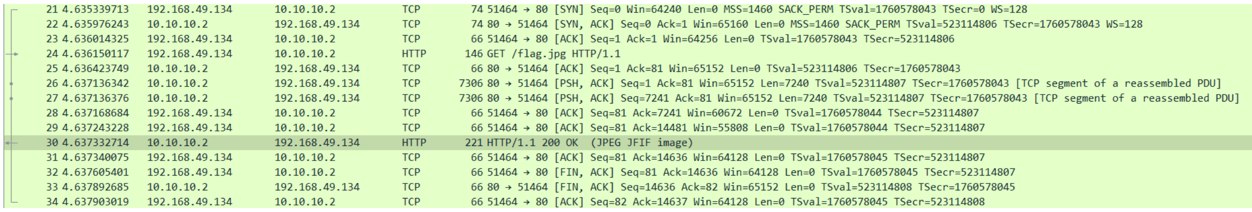


```
 5 0.001169943   192.168.49.134   10.10.10.2       TCP    66 47624 → 21 [ACK] Seq=1 Ack=29 Win=64256 Len=0 TSval=1760573408 TSecr=523110171
 6 0.001261761   192.168.49.134   10.10.10.2       FTP    84 Request: USER 0ffs3cUs3r3
 7 0.001513181   10.10.10.2       192.168.49.134   TCP    66 21 → 47624 [ACK] Seq=29 Ack=19 Win=65280 Len=0 TSval=523110171 TSecr=1760573408
 8 0.001781655   10.10.10.2       192.168.49.134   FTP    99 Response: 331 Username ok, send password.
 9 0.001790275   192.168.49.134   10.10.10.2       TCP    66 47624 → 21 [ACK] Seq=19 Ack=62 Win=64256 Len=0 TSval=1760573409 TSecr=523110171
10 0.001918610   192.168.49.134   10.10.10.2       FTP    93 Request: PASS very_secret_password
11 0.002058038   10.10.10.2       192.168.49.134   TCP    66 21 → 47624 [ACK] Seq=62 Ack=46 Win=65280 Len=0 TSval=523110172 TSecr=1760573409
12 0.002217803   10.10.10.2       192.168.49.134   FTP    89 Response: 230 Login successful.
13 0.002225235   192.168.49.134   10.10.10.2       TCP    66 47624 → 21 [ACK] Seq=46 Ack=85 Win=64256 Len=0 TSval=1760573409 TSecr=523110172
14 0.002408761   192.168.49.134   10.10.10.2       FTP    72 Request: QUIT
15 0.002609477   10.10.10.2       192.168.49.134   TCP    66 21 → 47624 [ACK] Seq=85 Ack=52 Win=65280 Len=0 TSval=523110172 TSecr=1760573410
16 0.002728723   10.10.10.2       192.168.49.134   FTP    80 Response: 221 Goodbye.
17 0.002735775   192.168.49.134   10.10.10.2       TCP    66 47624 → 21 [ACK] Seq=52 Ack=99 Win=64256 Len=0 TSval=1760573410 TSecr=523110173
18 0.002876793   10.10.10.2       192.168.49.134   TCP    66 21 → 47624 [FIN, ACK] Seq=99 Ack=52 Win=65280 Len=0 TSval=523110173 TSecr=1760573410
19 0.002955815   192.168.49.134   10.10.10.2       TCP    66 47624 → 21 [FIN, ACK] Seq=52 Ack=100 Win=64256 Len=0 TSval=1760573410 TSecr=523110173
20 0.003174289   10.10.10.2       192.168.49.134   TCP    66 21 → 47624 [ACK] Seq=100 Ack=53 Win=65280 Len=0 TSval=523110173 TSecr=1760573410
```

- Sequences 5 to 10 are purely based on the information either given by a person of the organization or one might have stole it by some means or are generally gained by hit-n-trial method.
    - Here the attacker sends the username - “**0ffs3cUs3r3**” for validation(if this is the correct username or not) - Sequence 6.
    - Sequence 8 confirms it to be the valid username and further asks for the password for the user id “**0ffs3cUs3r3**”.
    - Again the attacker provides the password - “**very_secret_password**” for validation of the user for getting inside the FTP Server - Sequence 10.
    - Sequence 12 validates and confirms the password.
- Sequences 13 to 16 depict the exiting of the attacker from the information gathering session.
    - Sequence 14 shows the source requesting to close the connection.
    - Sequence 16 affirms the closing of connection by sending “**Response**” as “**Goodbye**”.
- Sequences 18 and 19 show the termination of the connection (**[FIN, ACK]** where **FIN** depicts termination.

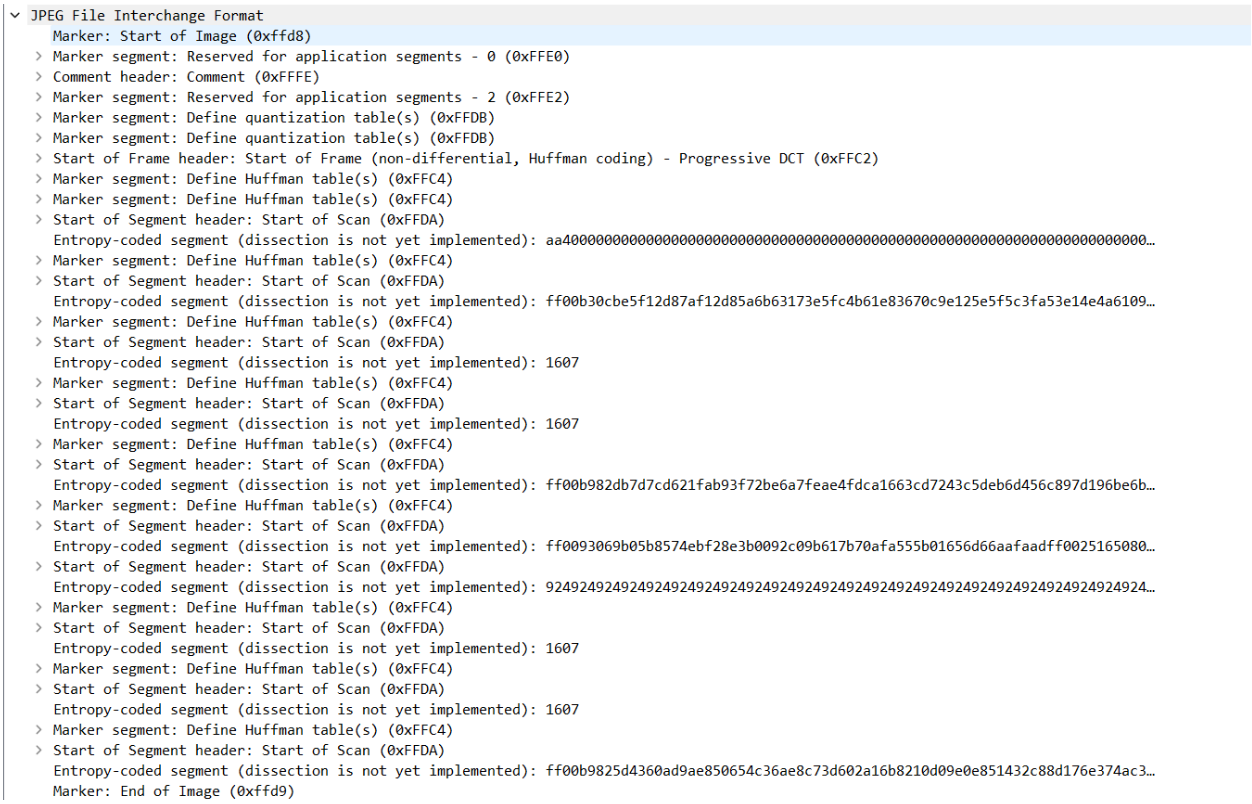
***NOTE* -** *Now after getting the verified credentials, I would use third-party apps like WinSCP or Putty(for Putty SSH[remote access] - port 22 should be enabled on the FTP server) to access the FTP server.*

CASE STUDY 2:



```
21 4.635339713   192.168.49.134   10.10.10.2       TCP     74 51464 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1760578043 TSecr=0 WS=128
22 4.635976243   10.10.10.2       192.168.49.134   TCP     74 80 → 51464 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM TSval=523114806 TSecr=1760578043 WS=128
23 4.636014325   192.168.49.134   10.10.10.2       TCP     66 51464 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1760578043 TSecr=523114806
24 4.636150117   192.168.49.134   10.10.10.2       HTTP   146 GET /flag.jpg HTTP/1.1
25 4.636423749   10.10.10.2       192.168.49.134   TCP     66 80 → 51464 [ACK] Seq=1 Ack=81 Win=65152 Len=0 TSval=523114806 TSecr=1760578043
26 4.637136342   10.10.10.2       192.168.49.134   TCP   7306 80 → 51464 [PSH, ACK] Seq=1 Ack=81 Win=65152 Len=7240 TSval=523114807 TSecr=1760578043 [TCP segment of a reassembled PDU]
27 4.637136376   10.10.10.2       192.168.49.134   TCP   7306 80 → 51464 [PSH, ACK] Seq=7241 Ack=81 Win=65152 Len=7240 TSval=523114807 TSecr=1760578043 [TCP segment of a reassembled PDU]
28 4.637168684   192.168.49.134   10.10.10.2       TCP     66 51464 → 80 [ACK] Seq=81 Ack=7241 Win=60672 Len=0 TSval=1760578044 TSecr=523114807
29 4.637243228   192.168.49.134   10.10.10.2       TCP     66 51464 → 80 [ACK] Seq=81 Ack=14481 Win=55808 Len=0 TSval=1760578044 TSecr=523114807
30 4.637332714   10.10.10.2       192.168.49.134   HTTP   221 HTTP/1.1 200 OK  (JPEG JFIF image)
31 4.637340075   192.168.49.134   10.10.10.2       TCP     66 51464 → 80 [ACK] Seq=81 Ack=14636 Win=64128 Len=0 TSval=1760578045 TSecr=523114807
32 4.637605401   192.168.49.134   10.10.10.2       TCP     66 51464 → 80 [FIN, ACK] Seq=81 Ack=14636 Win=64128 Len=0 TSval=1760578045 TSecr=523114807
33 4.637892685   10.10.10.2       192.168.49.134   TCP     66 80 → 51464 [FIN, ACK] Seq=14636 Ack=82 Win=65152 Len=0 TSval=523114808 TSecr=1760578045
34 4.637903019   192.168.49.134   10.10.10.2       TCP     66 51464 → 80 [ACK] Seq=82 Ack=14637 Win=64128 Len=0 TSval=1760578045 TSecr=523114808
```

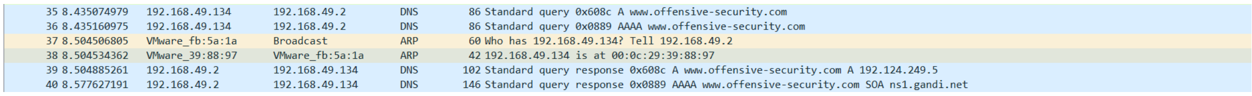
- Sequences 21-34 show a session of getting an image from a destination who's IP is 10.10.10.2 and the host here is 192.168.49.134.
- In this case, we find an open port at 80 which is the port used by **HTTP(here the version used is 1.1),** next the source requests for an image file named "**flag.jpg"** through **GET** method - Sequence 24.
- Sequence 30 shows successful arrival of the image from the destination which was received as "**JPEG/JFIF**"(Refer to the image below for full breakdown of JPEG File Interchange Format(log)).



```
⌄ JPEG File Interchange Format
    Marker: Start of Image (0xffd8)
  › Marker segment: Reserved for application segments - 0 (0xFFE0)
  › Comment header: Comment (0xFFFE)
  › Marker segment: Reserved for application segments - 2 (0xFFE2)
  › Marker segment: Define quantization table(s) (0xFFDB)
  › Marker segment: Define quantization table(s) (0xFFDB)
  › Start of Frame header: Start of Frame (non-differential, Huffman coding) - Progressive DCT (0xFFC2)
  › Marker segment: Define Huffman table(s) (0xFFC4)
  › Marker segment: Define Huffman table(s) (0xFFC4)
  › Start of Segment header: Start of Scan (0xFFDA)
    Entropy-coded segment (dissection is not yet implemented): aa400000000000000000000000000000000000000000000000000000000000000000000000…
  › Marker segment: Define Huffman table(s) (0xFFC4)
  › Start of Segment header: Start of Scan (0xFFDA)
    Entropy-coded segment (dissection is not yet implemented): ff00b30cbe5f12d87af12d85a6b63173e5fc4b61e83670c9e125e5f5c3fa53e14e4a6109…
  › Marker segment: Define Huffman table(s) (0xFFC4)
  › Start of Segment header: Start of Scan (0xFFDA)
    Entropy-coded segment (dissection is not yet implemented): 1607
  › Marker segment: Define Huffman table(s) (0xFFC4)
  › Start of Segment header: Start of Scan (0xFFDA)
    Entropy-coded segment (dissection is not yet implemented): 1607
  › Marker segment: Define Huffman table(s) (0xFFC4)
  › Start of Segment header: Start of Scan (0xFFDA)
    Entropy-coded segment (dissection is not yet implemented): ff00b982db7d7cd621fab93f72be6a7feae4fdca1663cd7243c5deb6d456c897d196be6b…
  › Marker segment: Define Huffman table(s) (0xFFC4)
  › Start of Segment header: Start of Scan (0xFFDA)
    Entropy-coded segment (dissection is not yet implemented): ff0093069b05b8574ebf28e3b0092c09b617b70afa555b01656d66aafaadff0025165080…
  › Start of Segment header: Start of Scan (0xFFDA)
    Entropy-coded segment (dissection is not yet implemented): 924924924924924924924924924924924924924924924924924924924924924924924924…
  › Marker segment: Define Huffman table(s) (0xFFC4)
  › Start of Segment header: Start of Scan (0xFFDA)
    Entropy-coded segment (dissection is not yet implemented): 1607
  › Marker segment: Define Huffman table(s) (0xFFC4)
  › Start of Segment header: Start of Scan (0xFFDA)
    Entropy-coded segment (dissection is not yet implemented): 1607
  › Marker segment: Define Huffman table(s) (0xFFC4)
  › Start of Segment header: Start of Scan (0xFFDA)
    Entropy-coded segment (dissection is not yet implemented): ff00b9825d4360ad9ae850654c36ae8c73d602a16b8210d09e0e851432c88d176e374ac3…
    Marker: End of Image (0xffd9)
```

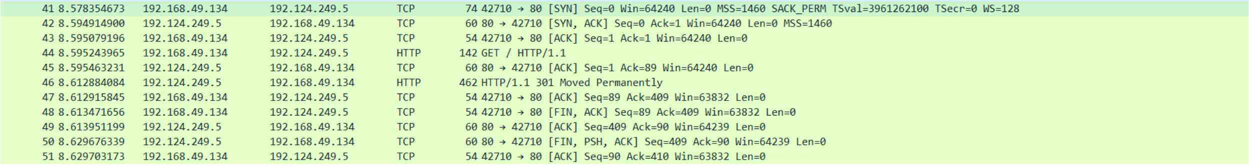
- Sequences 31-34 show the termination of connection from the session.

CASE STUDY 3:



```
35 8.435074979   192.168.49.134    192.168.49.2      DNS        86 Standard query 0x608c A www.offensive-security.com
36 8.435160975   192.168.49.134    192.168.49.2      DNS        86 Standard query 0x0889 AAAA www.offensive-security.com
37 8.504506805   VMware_fb:5a:1a   Broadcast         ARP        60 Who has 192.168.49.134? Tell 192.168.49.2
38 8.504534362   VMware_39:88:97   VMware_fb:5a:1a   ARP        42 192.168.49.134 is at 00:0c:29:39:88:97
39 8.504885261   192.168.49.2      192.168.49.134    DNS       102 Standard query response 0x608c A www.offensive-security.com A 192.124.249.5
40 8.577627191   192.168.49.2      192.168.49.134    DNS       146 Standard query response 0x0889 AAAA www.offensive-security.com SOA ns1.gandi.net
```

- Sequences 35-36 attempt to gather information about a domain "**www.offensive-security.com**" from "**192.168.49.2**"(It seems like "**192.168.49.2**" is the IP of a Database :3).
  - Sequence 39 - Requests for the IPV4 address of the domain (A).
  - Sequence 40 - Requests for the IPV6 address of the domain (AAAA).
- Sequences 37 attempts to find the IP "**192.168.49.134**" and asks it to respond to the source here - "**192.168.49.2**"; Sequence 38 shows successful search of "**192.168.49.134**" and informs the source that it is at MAC address "**00:0c:29:39:88:97**".
- Sequences 39-40 shows the source "**192.168.49.2**" successfully providing the info to the destination "**192.168.49.134**".
  - Sequence 39 - Provides the IPV4 address of the domain "**www.offensive-security.com**" - "**192.124.249.5**".
  - Sequence 40 - Provides the IPV6 address with Start of Authority(SOA) indicating the authoritative DNS server for the domain is "**ns1.gandi.net**".



```
41 8.578354673   192.168.49.134    192.124.249.5     TCP        74 42710 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3961262100 TSecr=0 WS=128
42 8.594914900   192.124.249.5     192.168.49.134    TCP        60 80 → 42710 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
43 8.595079196   192.168.49.134    192.124.249.5     TCP        54 42710 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
44 8.595243965   192.168.49.134    192.124.249.5     HTTP      142 GET / HTTP/1.1
45 8.595463231   192.124.249.5     192.168.49.134    TCP        60 80 → 42710 [ACK] Seq=1 Ack=89 Win=64240 Len=0
46 8.612884084   192.124.249.5     192.168.49.134    HTTP      462 HTTP/1.1 301 Moved Permanently
47 8.612915845   192.168.49.134    192.124.249.5     TCP        54 42710 → 80 [ACK] Seq=89 Ack=409 Win=63832 Len=0
48 8.613471656   192.168.49.134    192.124.249.5     TCP        54 42710 → 80 [FIN, ACK] Seq=89 Ack=409 Win=63832 Len=0
49 8.613951199   192.124.249.5     192.168.49.134    TCP        60 80 → 42710 [ACK] Seq=409 Ack=90 Win=64239 Len=0
50 8.629676339   192.124.249.5     192.168.49.134    TCP        60 80 → 42710 [FIN, PSH, ACK] Seq=409 Ack=90 Win=64239 Len=0
51 8.629703173   192.168.49.134    192.124.249.5     TCP        54 42710 → 80 [ACK] Seq=90 Ack=410 Win=63832 Len=0
```

- Sequences 41-42 shows "**192.168.49.134**" attempting to establish a successful connection with "**192.124.249.5**". Sequence 42 shows it to be a success(with **192.124.249.5**'s open port being **80** which is used by HTTP).
- Sequence 44 shows "**192.168.49.134**" asking the root path - "/" from the web-server "**192.124.249.5**".
- Sequence 46 shows web-server "**192.124.249.5**" responding to the request with **HTTP** status code **301** meaning the resource has been permanently

moved to a different location.

- In sequences 47-51, termination of the session can be seen. In sequence 50, “**192.124.249.5**” asks “**192.168.49.134**” to push the information received to the application ASAP without any further delays.

2. SOUND-FILE CASE STUDY

It is a sound morse code, which when heard on “**Two Tone - 300Hz Frequency & 20 WPM**” mode translates to - ”**HEREISONEMESSAGEAT300HZAND20WPM**”; more clearly - “**HERE IS ONE MESSAGE AT 300 HZ AND 20 WPM**”.

3. ENCRYPTED TEXT

It seems to be an encrypted sentence which can be decoded using the method of “**Whitespace Steganography**”.

P.S. - Cant decode it T_T, CSI seniors pls help ;-;