

2019 oplossingen labo 4 - Lars Lemmens

Met dank aan de [Github van Martijn](#) en natuurlijk Lars Lemmens

Labo 4

What is the IP address of the client and what IP address is used by the server?

Which protocol was used to find the IP address of the FTP server.

Does FTP use UDP or TCP? Why?

Is the session using active or passive FTP?

Who chooses the FTP version, active or passive? Is it the client or the server?

Sketch the three-way handshake used to negotiate the initial sequence numbers.

Do this both for the relative and the real sequence numbers.

What is the benefit in using relative sequence numbers in Wireshark/Tshark?

Which credentials were used to login to the FTP service?

```
'user@:~$' • tshark -r Cnw2_ftp.pcap -Y "tcp.srcport == $(tshark -r Cnw2_ftp.pcap -Y 'ftp.response.code == 230' -T fields -e tcp.dstport) && ftp.request.command == USER" -T -e ftp.request.arg
```

- The tshark command dumps and analyzes network traffic
- The -r argument reads the packet data from infile
- The -Y argument captures the link type
- The -T argument sets the format of the output when viewing decoded packet data.
- The -e argument (in tshark command) adds a field to the list of fields to display if -T fields is selected

"Step 1:

When a user's login attempt is successful, the FTP server answers with FTP response code 230 <- Response code: User logged in, proceed (230).

This message is sent to the client using its chosen TCP port. This very same TCP port is used by the client for every packet sent in the control connection.

So we know the client will use this port as TCP source port when providing the FTP username. Thus the first step is to find this TCP port:

See Cnw2 theory - Chapter 2: Application Layer: Active vs Passive FTP

```
'user@:~$' • tshark -r Cnw2_ftp.pcap -Y 'ftp.response.code == 230' -T fields -e tcp.dstport
```

- The tshark command dumps and analyzes network traffic
- The -r argument reads the packet data from infile
- The -Y argument captures the link type
- The -T argument sets the format of the output when viewing decoded packet data.
- The -e argument (in tshark command) adds a field to the list of fields to display if -T fields is selected

Step 2:

The port we just found is the TCP port the client uses for the FTP control connection.

When using this port as TCP source port in our display filter, we'll only list packets sent from client to FTP server

```
'user@:~$' • tshark -r Cnw2_ftp.pcap -Y "tcp.srcport == $(tshark -r Cnw2_ftp.pcap -Y 'ftp.response.code == 230' -T fields -e tcp.dstport)"
```

- The tshark command dumps and analyzes network traffic
- The -r argument reads the packet data from infile
- The -Y argument captures the link type
- The -T argument sets the format of the output when viewing decoded packet data. The -e argument (in tshark command) adds a field to the list of fields to display if -T fields is selected

Step 3:

We're only interested in the packet containing the username, so let's add 'ftp.request.command == USER' to the display filter:

```
'user@:~$' • tshark -r Cnw2_ftp.pcap -Y "tcp.srcport == $(tshark -r Cnw2_ftp.pcap -Y 'ftp.response.code == 230' -T fields -e tcp.dstport) && ftp.request.command == USER"
```

- The tshark command dumps and analyzes network traffic
- The -r argument reads the packet data from infile
- The -Y argument captures the link type
- The -T argument sets the format of the output when viewing decoded packet data.
- The -e argument (in tshark command) adds a field to the list of fields to display if -T fields is selected
- The server sends a 230 code in response to a command that has provided sufficient credentials to the server to grant the user access to the FTP server.

Step 4:

And finally use the tshark fields option to only display the username used:

```
'user@:~$' • tshark -r Cnw2_ftp.pcap -Y "tcp.srcport == $(tshark -r Cnw2_ftp.pcap -Y 'ftp.response.code == 230' -T fields -e tcp.dstport) && ftp.request.command == USER" -T fields -e ftp.request.arg debbie
```

- The tshark command dumps and analyzes network traffic
- The -r argument reads the packet data from infile
- The -Y argument captures the link type
- The -T argument sets the format of the output when viewing decoded packet data.
- The -e argument (in tshark command) adds a field to the list of fields to display if -T fields is selected
- The server sends a 230 code in response to a command that has provided sufficient credentials to the server to grant the user access to the FTP server.

Which file was downloaded/uploaded from/to the FTP server? (delete as appropriate)

```
'user@~$' • tshark -r Cnw2_ftp.pcap -Y 'ftp.request.command == RETR' | grep -Po 'RETR \K.*'
```

- The RETR argument ==> client issues the RETR command after successfully establishing a data connection when it wishes to download a copy of a file on the server.
- The -e argument (in tshark command) adds a field to the list of fields to display if -T fields is selected
- The -P argument interprets I as Perl-compatible regular expressions (PCREs).
- The -o argument prints only the matched (non-empty) parts of a matching line, with each such part on a separate output line.³²

Can you reconstruct this file from the Wireshark packet dump? Open the file to verify it's contents are intact.

How many packets have their destination port set to 21?

```
'user@~$' • tshark -r Cnw2_ftp_bruteforce.pcap -Y 'tcp.dstport == 21' | wc -l
```

- The -r argument reads the packet data from infile
- The -Y argument captures the link type
- tcp.dstport == 21 => TCP port
- The wc command prints newline, word, and byte counts for each file
- The argument -l prints the newline counts

List all packets which have the PUSH bit set. What is the benefit in setting this bit?

```
'user@~$' • tshark -r Cnw2_ftp_bruteforce.pcap -Y 'tcp.flags.push == 1'
```

- The -r argument reads the packet data from infile
- tcp.flags.push == 1 => Which means "check if the Push flag is set". Filtering for just "tcp.flags.push" would again mean "check if there's a push flag field" (which there is, always)

How can you manually calculate the actual port advertised by the PASV/PORT command?

```
'user@~$' • tshark -r Cnw2_ftp.pcap -Y 'ftp.response.code == 227' | awk -F ',' '{print $5*256+$6}'
```

- The -r argument reads the packet data from infile

- The -Y argument captures the link type
- This is the response given by the server to the PASV command. It indicates that the server is ready for the client to connect to it for the purpose of establishing a data connection.
- The awk command is used for pattern scanning and processing language
- The -F argument
- 5 matches the character 5 literally (case sensitive)
- '*' matches the previous token between zero and unlimited times, as many times as possible, giving back as needed (greedy)
- 25 matches the characters 25 literally (case sensitive)
- 6 matches the character 6 literally (case sensitive) + matches the previous token between one and unlimited times, as many times as possible, giving back as needed (greedy)
- \$ asserts position at the end of a line
- 6 matches the character 6 literally (case sensitive)

How many L4 sessions were created in the FTP session? Note: passive FTP was used in this session.

```
'user@~$' • tshark -r Cnw2_ftp.pcap -Y 'ftp.response.code == 230 || ftp.response.code == 227' | wc -l
```

- The server sends a 230 code in response to a command that has provided sufficient credentials to the server to grant the user access to the FTP server.
- This is the response given by the server to the PASV command. It indicates that the server is ready for the client to connect to it for the purpose of establishing a data connection.
- The wc command prints newline, word, and byte counts for each file
- The argument -l prints the newline counts

Try to answer these questions with
'cnw2_ftp_bruteforce.pcap: (file)'

How many password guesses were made?

```
'user@~$' • tshark -r Cnw2_ftp_bruteforce.pcap -Y 'ftp.request.command == PASS' -T fields -e ftp.request.arg | wc -l
```

- The -r argument reads the packet data from infile
- The -Y argument captures the link type

- The -T argument sets the format of the output when viewing decoded packet data.
- The -e argument (in tshark command) adds a field to the list of fields to display if -T fields is selected
- So, to see all login attempts, try this filter: ftp.request.command==USER || ftp.request.command==PASS
- The wc command prints newline, word, and byte counts for each file
- The argument -l prints the newline counts

Did the attacker finally get in?

```
'user@~$' • tshark -r Cnw2_ftp_bruteforce.pcap -Y 'ftp.response.code == 230' | grep -q 'Response: 230' && echo YES || echo NO
```

- The server sends a 230 code in response to a command that has provided sufficient credentials to the server to grant the user access to the FTP server.
- The -q argument = quiet - do not write anything to standard output
- The -r argument reads the packet data from infile
- The -Y argument captures the link type

Which usernames did he/she try?

```
'user@~$' • tshark -r Cnw2_ftp_bruteforce.pcap -Y 'ftp.request.command == USER' -T fields -e ftp.request.arg | sort -u
```

- The -r argument reads the packet data from infile
- The -Y argument captures the link type
- The -T argument sets the format of the output when viewing decoded packet data.
- The -e argument (in tshark command) adds a field to the list of fields to display if -T fields is selected
- The -u argument => with -c, checks for strict ordering; without -c, output only the first of an equal run

Revision #1

Created 17 June 2021 14:14:22 by Jasper G.

Updated 3 December 2021 22:13:09 by Jasper G.