Networking Secure Protocols

Transport Layer Security (TLS) is added to existing protocols to protect communication confidentiality, integrity, and authenticity

it is deemed insecure to remotely access a system using the TELNET protocol; Secure Shell (SSH) was created to provide a secure way to access remote systems.

TLS - Transport Layer Security

TLS ensures that no one can read or modify the exchanged data ensuring confidentiality and integrity.

overview of how TLS is set up and used:

The first step for every server (or client) that needs to identify itself is to get a signed TLS certificate. Generally, the server administrator creates a Certificate Signing Request (CSR) and submits it to a Certificate Authority (CA); the CA verifies the CSR and issues a digital certificate. Once the (signed) certificate is received, it can be used to identify the server (or the client) to others, who can confirm the validity of the signature. For a host to confirm the validity of a signed certificate, the certificates of the signing authorities need to be installed on the host. In the non-digital world, this is similar to recognising the stamps of various authorities.

Generally speaking, getting a certificate signed requires paying an annual fee. However, Let's Encrypt allows you to get your certificate signed for free.

Finally, we should mention that some users opt to create a self-signed certificate. A self-signed certificate cannot prove the server's authenticity as no third party has confirmed it.

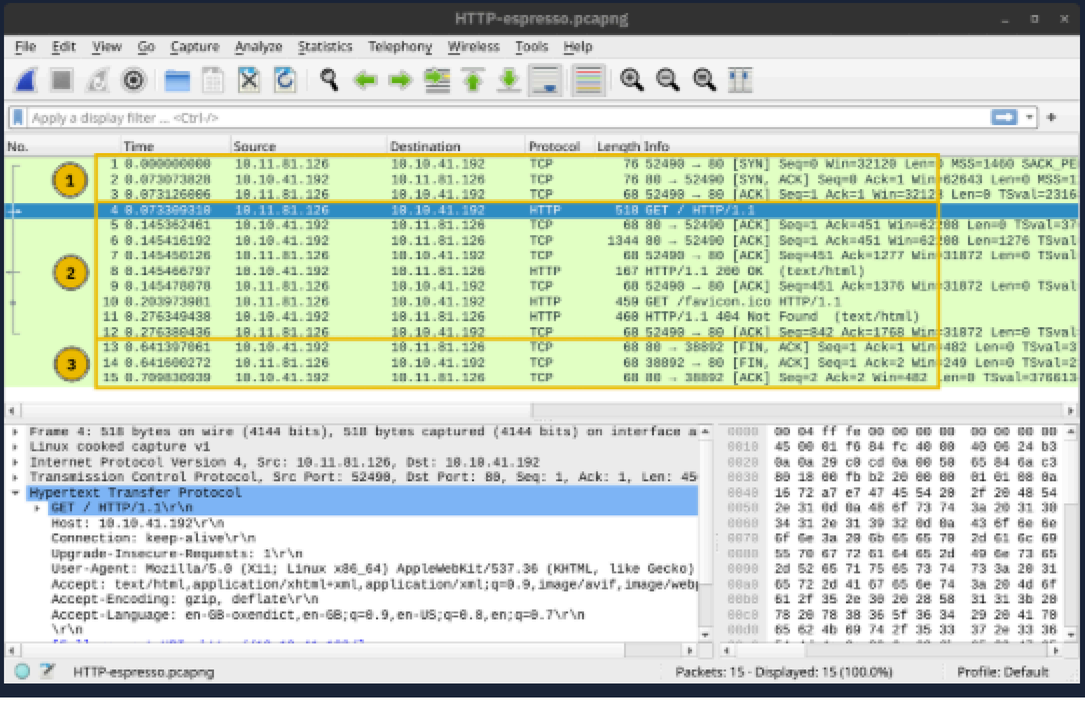
HTTPS

HTTP

most common steps before a web browser can request a page over HTTP. After resolving the domain name to an IP address, the client will carry out the following two steps:

1. Establish a TCP three-way handshake with the target server
2. Communicate using the HTTP protocol; for example, issue HTTP requests, such as `GET / HTTP/1.1`

The two steps described above are shown in the window below. The three packets for the TCP handshake (marked with 1) precede the first HTTP packet with `GET` in it. The HTTP communication is marked with 2. The last three displayed packets are for TCP connection termination and are marked with 3.


HTTP-espresso.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
1
2
3

```
No.   Time          Source          Destination     Protocol  Length Info
  1 0.000000000   10.11.81.126    10.10.41.192    TCP        76 52490 → 80 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PE
  2 0.073073828   10.10.41.192    10.11.81.126    TCP        76 80 → 52490 [SYN, ACK] Seq=0 Ack=1 Win=62643 Len=0 MSS=1
  3 0.073126806   10.11.81.126    10.10.41.192    TCP        68 52490 → 80 [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSval=2316
  4 0.073309318   10.11.81.126    10.10.41.192    HTTP      518 GET / HTTP/1.1
  5 0.145362461   10.10.41.192    10.11.81.126    TCP        68 80 → 52490 [ACK] Seq=1 Ack=451 Win=62208 Len=0 TSval=37
  6 0.145416192   10.10.41.192    10.11.81.126    TCP      1344 80 → 52490 [ACK] Seq=1 Ack=451 Win=62208 Len=1276 TSval
  7 0.145450126   10.11.81.126    10.10.41.192    TCP        68 52490 → 80 [ACK] Seq=451 Ack=1277 Win=31872 Len=0 TSval
  8 0.145466797   10.10.41.192    10.11.81.126    HTTP      167 HTTP/1.1 200 OK  (text/html)
  9 0.145478078   10.11.81.126    10.10.41.192    TCP        68 52490 → 80 [ACK] Seq=451 Ack=1376 Win=31872 Len=0 TSval
 10 0.203973981   10.11.81.126    10.10.41.192    HTTP      459 GET /favicon.ico HTTP/1.1
 11 0.276349438   10.10.41.192    10.11.81.126    HTTP      468 HTTP/1.1 404 Not Found  (text/html)
 12 0.276380436   10.11.81.126    10.10.41.192    TCP        68 52490 → 80 [ACK] Seq=842 Ack=1768 Win=31872 Len=0 TSval
 13 0.641397061   10.10.41.192    10.11.81.126    TCP        68 80 → 38892 [FIN, ACK] Seq=1 Ack=1 Win=482 Len=0 TSval=3
 14 0.641600272   10.11.81.126    10.10.41.192    TCP        68 38892 → 80 [FIN, ACK] Seq=1 Ack=2 Win=249 Len=0 TSval=2
 15 0.709830939   10.10.41.192    10.11.81.126    TCP        68 80 → 38892 [ACK] Seq=2 Ack=2 Win=482 Len=0 TSval=378613
```

```
▸ Frame 4: 518 bytes on wire (4144 bits), 518 bytes captured (4144 bits) on interface a
▸ Linux cooked capture v1
▸ Internet Protocol Version 4, Src: 10.11.81.126, Dst: 10.10.41.192
▸ Transmission Control Protocol, Src Port: 52490, Dst Port: 80, Seq: 1, Ack: 1, Len: 45
▾ Hypertext Transfer Protocol
  ▸ GET / HTTP/1.1\r\n
    Host: 10.10.41.192\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/web
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: en-GB-oxendict,en-GB;q=0.9,en-US;q=0.8,en;q=0.7\r\n
    \r\n
```

```
0000  00 04 ff fe 00 00 00 00   00 00 00 00
0010  45 00 01 f6 84 fc 40 00   40 06 24 b3
0020  0a 0a 29 c8 cd 0a 00 50   65 84 6a c3
0030  80 18 00 fb b2 20 00 00   01 01 08 0a
0040  16 72 a7 e7 47 45 54 20   2f 20 48 54
0050  2e 31 0d 0a 48 6f 73 74   3a 20 31 30
0060  34 31 2e 31 39 32 0d 0a   43 6f 6e 6e
0070  6f 6e 3a 20 6b 65 65 70   2d 61 6c 69
0080  55 70 67 72 61 64 65 2d   49 6e 73 65
0090  2d 52 65 71 75 65 73 74   73 3a 20 31
00a0  65 72 2d 41 67 65 6e 74   3a 20 4d 6f
00b0  61 2f 35 2e 30 20 28 58   31 31 3b 20
00c0  78 20 78 38 36 5f 36 34   29 20 41 70
00d0  65 62 4b 69 74 2f 35 33   37 2e 33 36
```

```
HTTP-espresso.pcapng          Packets: 15 · Displayed: 15 (100.0%)          Profile: Default
```

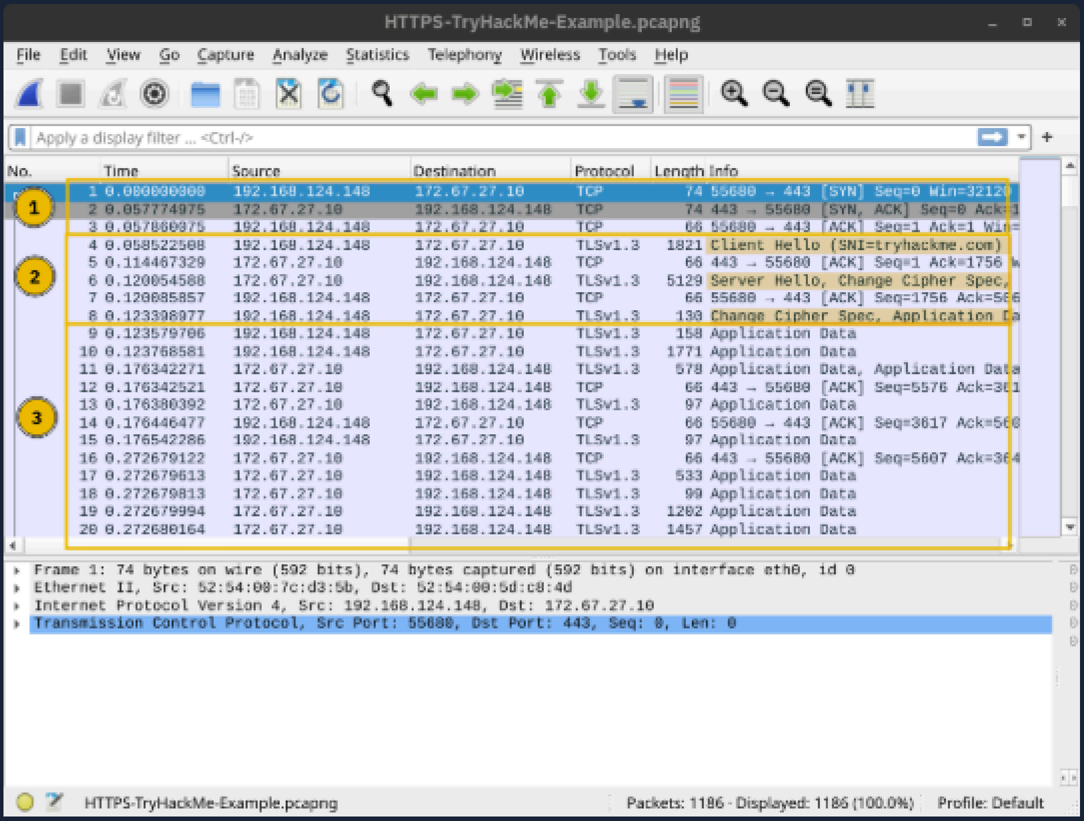
HTTP Over TLS

HTTPS stands for Hypertext Transfer Protocol Secure. It is basically HTTP over TLS. Consequently, requesting a page over HTTPS will require the following three steps (after resolving the domain name):

1. Establish a TCP three-way handshake with the target server
2. Establish a TLS session
3. Communicate using the HTTP protocol; for example, issue HTTP requests, such as `GET / HTTP/1.1`

The screenshot below shows that a TCP session is established in the first three packets, marked with `1`. Then, several packets are exchanged to negotiate the TLS protocol, marked with `2`. `1` and `2` are where the **TLS negotiation and establishment** take place.

Finally, HTTP application data is exchanged, marked with `3`. Looking at the Wireshark screenshot, we see that it says "Application Data" because there is no way to know if it is indeed HTTP or some other protocol sent over port 443.


HTTPS-TryHackMe-Example.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
1
2
3
HTTPS-TryHackMe-Example.pcapng
Packets: 1186 · Displayed: 1186 (100.0%)
Profile: Default

```
No.  Time         Source           Destination      Protocol Length Info
   1 0.000000000  192.168.124.148  172.67.27.10     TCP        74 55680 → 443 [SYN] Seq=0 Win=32120
   2 0.057774975  172.67.27.10     192.168.124.148  TCP        74 443 → 55680 [SYN, ACK] Seq=0 Ack=1
   3 0.057860975  192.168.124.148  172.67.27.10     TCP        66 55680 → 443 [ACK] Seq=1 Ack=1 Win=
   4 0.058522508  192.168.124.148  172.67.27.10     TLSv1.3  1821 Client Hello (SNI=tryhackme.com)
   5 0.114467329  172.67.27.10     192.168.124.148  TCP        66 443 → 55680 [ACK] Seq=1 Ack=1756 W
   6 0.120054588  172.67.27.10     192.168.124.148  TLSv1.3  5129 Server Hello, Change Cipher Spec,
   7 0.120085857  192.168.124.148  172.67.27.10     TCP        66 55680 → 443 [ACK] Seq=1756 Ack=556
   8 0.123398977  192.168.124.148  172.67.27.10     TLSv1.3   130 Change Cipher Spec, Application Da
   9 0.123579706  192.168.124.148  172.67.27.10     TLSv1.3   158 Application Data
  10 0.123768581  192.168.124.148  172.67.27.10     TLSv1.3  1771 Application Data
  11 0.176342271  172.67.27.10     192.168.124.148  TLSv1.3   578 Application Data, Application Data
  12 0.176342521  172.67.27.10     192.168.124.148  TCP        66 443 → 55680 [ACK] Seq=5576 Ack=361
  13 0.176380392  172.67.27.10     192.168.124.148  TLSv1.3    97 Application Data
  14 0.176446477  192.168.124.148  172.67.27.10     TCP        66 55680 → 443 [ACK] Seq=3617 Ack=560
  15 0.176542286  192.168.124.148  172.67.27.10     TLSv1.3    97 Application Data
  16 0.272679122  172.67.27.10     192.168.124.148  TCP        66 443 → 55680 [ACK] Seq=5607 Ack=364
  17 0.272679613  172.67.27.10     192.168.124.148  TLSv1.3   533 Application Data
  18 0.272679813  172.67.27.10     192.168.124.148  TLSv1.3    99 Application Data
  19 0.272679994  172.67.27.10     192.168.124.148  TLSv1.3  1202 Application Data
  20 0.272680164  172.67.27.10     192.168.124.148  TLSv1.3  1457 Application Data

▸ Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface eth0, id 0
▸ Ethernet II, Src: 52:54:00:7c:d3:5b, Dst: 52:54:00:5d:c8:4d
▸ Internet Protocol Version 4, Src: 192.168.124.148, Dst: 172.67.27.10
▸ Transmission Control Protocol, Src Port: 55680, Dst Port: 443, Seq: 0, Len: 0
```

In wireshark if someone tries capturing the stream of packets it shows the traffic as encrypted so cannot be read in plain text like HTTP and is no way of reading it without an encryption key

Getting the Encryption Key

TLS offered security for HTTP without requiring any changes in the lower or higher layer protocols. In other words, TCP and IP were not modified, while HTTP was sent over TLS the way it would be sent over TCP.

SMTPS, POP3S, IMAPS

Adding TLS to SMTP, POP3, and IMAP is no different than adding TLS to HTTP.

The insecure versions use the default TCP port numbers shown in the table below:

Protocol	Default Port Number
HTTP	80
SMTP	25
POP3	110
IMAP	143

The secure versions, i.e., over TLS, use the following TCP port numbers by default:

Protocol	Default Port Number
HTTPS	443
SMTPS	465 and 587
POP3S	995
IMAPS	993

SSH

It is easy for anyone monitoring the network traffic to get hold of your login credentials once you use `telnet`

OpenSSH offers several benefits. We will list a few key points:

- **Secure authentication**: Besides password-based authentication, SSH supports public key and two-factor authentication.
- **Confidentiality**: OpenSSH provides end-to-end encryption, protecting against eavesdropping. Furthermore, it notifies you of new server keys to protect against man-in-the-middle attacks.
- **Integrity**: In addition to protecting the confidentiality of the exchanged data, cryptography also protects the integrity of the traffic.
- **Tunneling**: SSH can create a secure “tunnel” to route other protocols through SSH. This setup leads to a VPN-like connection.
- **X11 Forwarding**: If you connect to a Unix-like system with a graphical user interface, SSH allows you to use the graphical application over the network.

For ssh the command is: `ssh username@hostname` If the username is the same as your logged-in username, you only need `ssh hostname`

While the TELNET server listens on **port 23**, the SSH server listens on **port 22**.

SFTP and FTPS

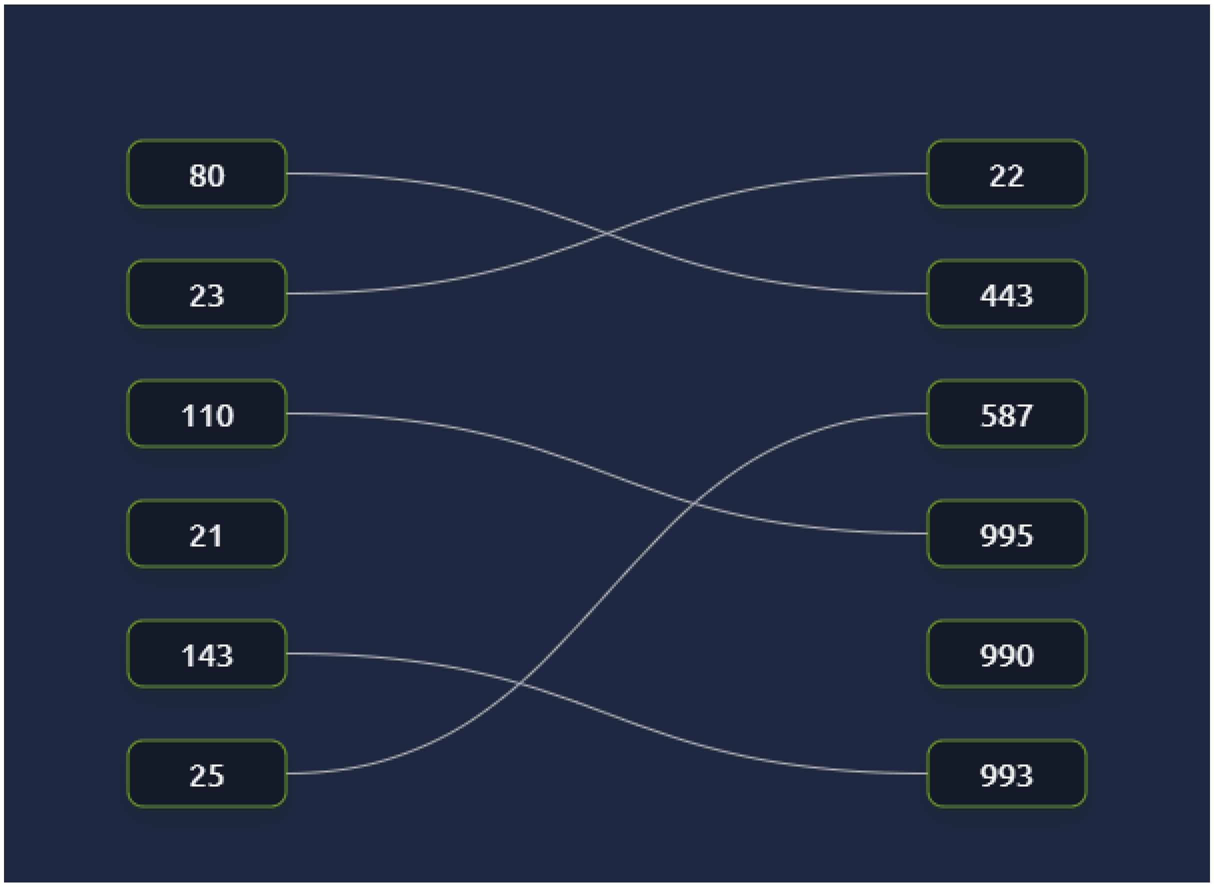
SFTP stands for SSH File Transfer Protocol and allows secure file transfer. It is part of the SSH protocol suite and shares the same port number, **22**. If enabled in the OpenSSH server configuration, you can connect using a command such as `sftp username@hostname`

Once logged in, you can issue commands such as `get filename` and `put filename` to download and upload files, respectively

While FTP uses port **21**, FTPS usually uses port **990**

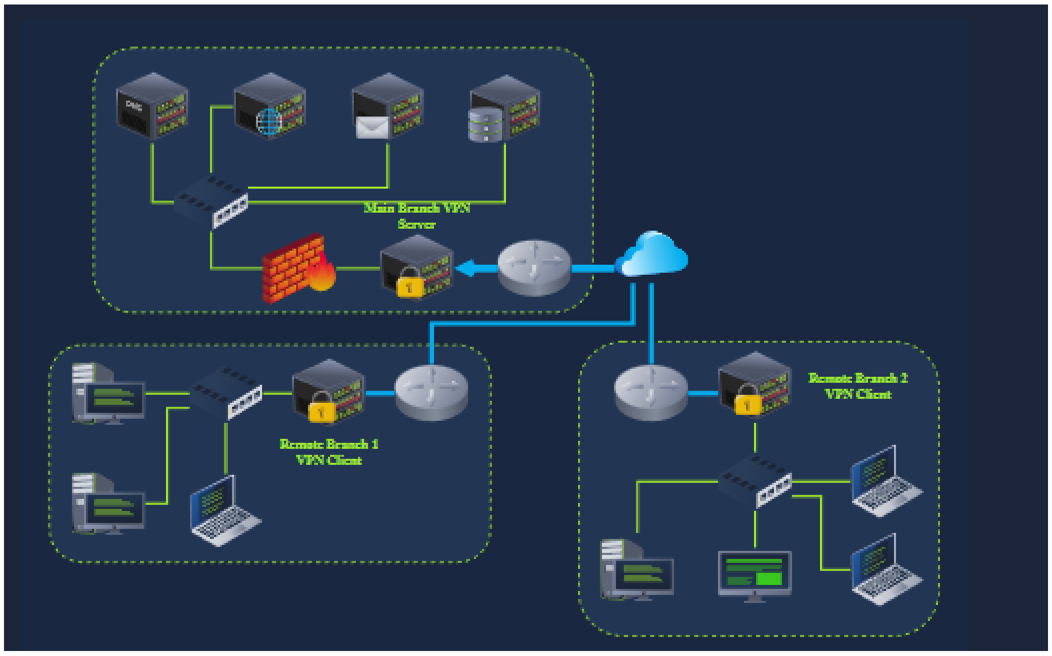
Here are the default port numbers of protocols for their insecure and secure versions

insecure being left side secure being right side

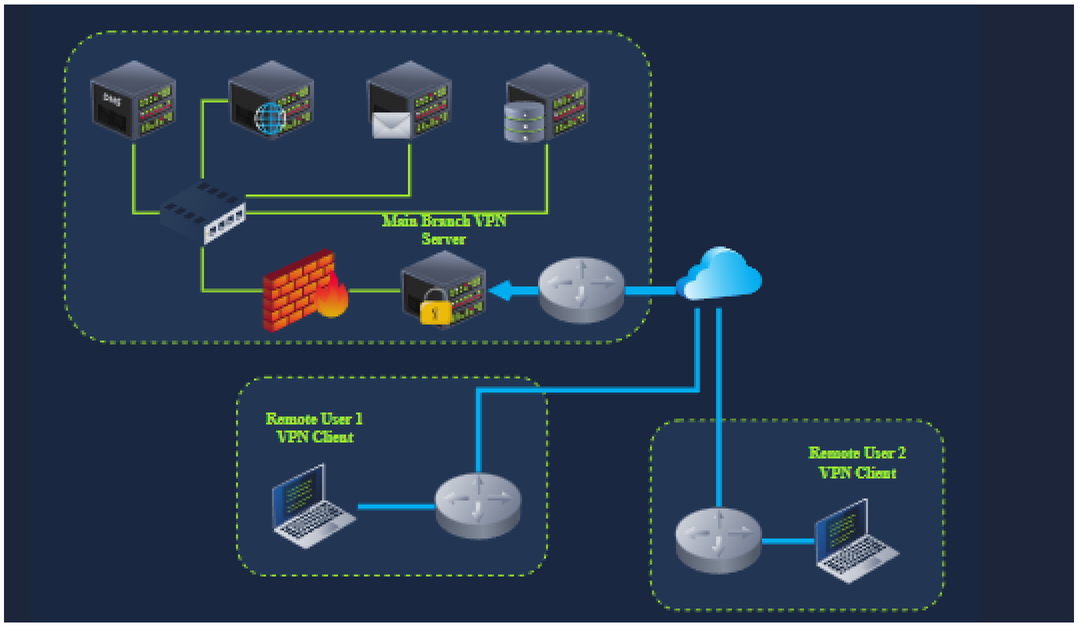


VPN

A VPN client in the remote branches is expected to connect to the VPN server in the main branch. In this case, the VPN client will encrypt the traffic and pass it to the main branch via the established VPN tunnel (shown in blue). The VPN traffic is limited to the blue lines; the green lines would carry the decrypted VPN traffic.

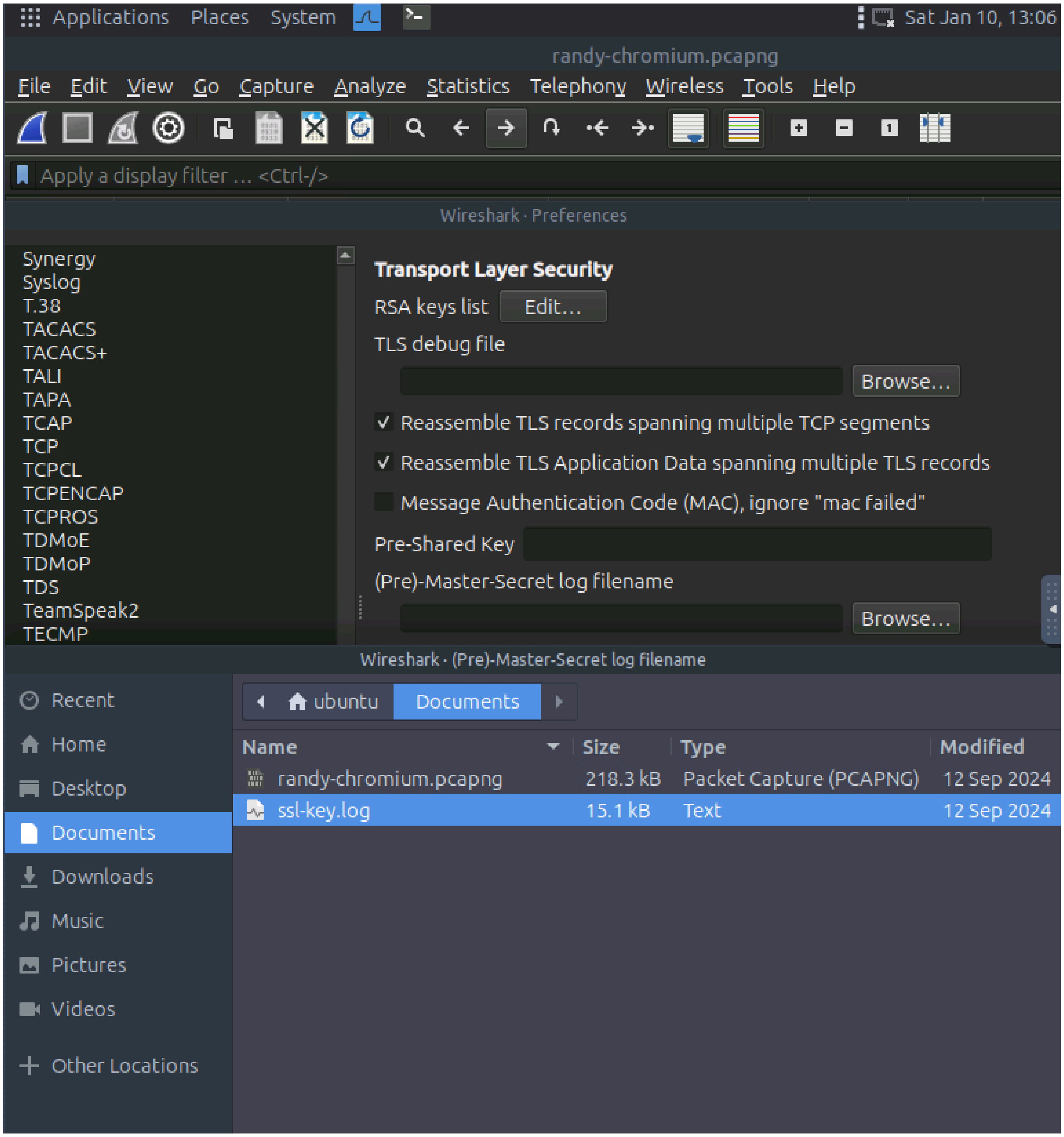


In the network diagram below, we see two remote users using VPN clients to connect to the VPN server in the main branch. In this case, the VPN client connects a single device.



Practical task

on wireshark i opened up the file we will be analysing then had to right click anywhere and have to select the ssl.key log file to decrypt the tls traffic. this can be done in protocol preferences > transport layer security > open transport layer security and will open this page :



Transport Layer Security

RSA keys list Edit…

TLS debug file Browse…

✓ Reassemble TLS records spanning multiple TCP segments

✓ Reassemble TLS Application Data spanning multiple TLS records

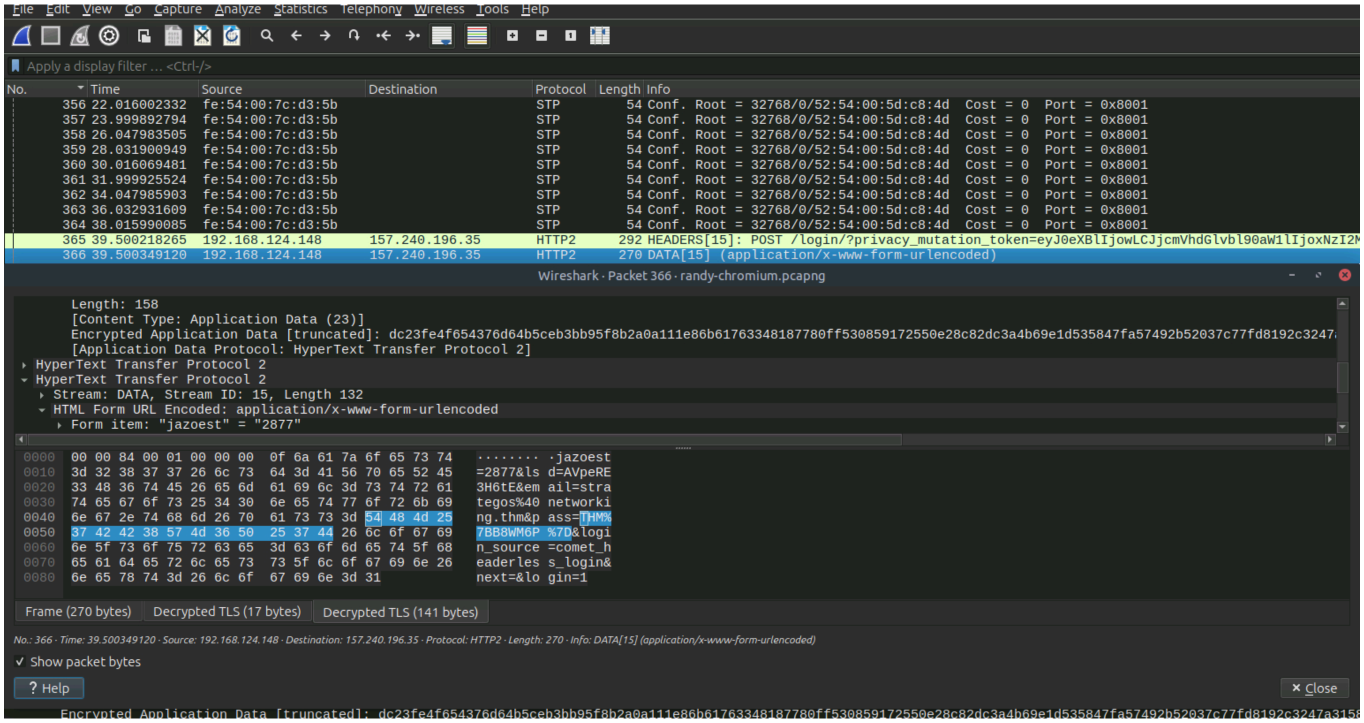
Message Authentication Code (MAC), ignore "mac failed"

Pre-Shared Key

(Pre)-Master-Secret log filename Browse…

Name	Size	Type	Modified
randy-chromium.pcapng	218.3 kB	Packet Capture (PCAPNG)	12 Sep 2024
ssl-key.log	15.1 kB	Text	12 Sep 2024

where i can then select the ssl key log file under the pre-master-secret log filename


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>

```
No.    Time         Source             Destination      Protocol Length Info
   356 22.016002332 fe:54:00:7c:d3:5b                   STP        54 Conf. Root = 32768/0/52:54:00:5d:c8:4d  Cost = 0  Port = 0x8001
   357 23.999892794 fe:54:00:7c:d3:5b                   STP        54 Conf. Root = 32768/0/52:54:00:5d:c8:4d  Cost = 0  Port = 0x8001
   358 26.047983505 fe:54:00:7c:d3:5b                   STP        54 Conf. Root = 32768/0/52:54:00:5d:c8:4d  Cost = 0  Port = 0x8001
   359 28.031900949 fe:54:00:7c:d3:5b                   STP        54 Conf. Root = 32768/0/52:54:00:5d:c8:4d  Cost = 0  Port = 0x8001
   360 30.016069481 fe:54:00:7c:d3:5b                   STP        54 Conf. Root = 32768/0/52:54:00:5d:c8:4d  Cost = 0  Port = 0x8001
   361 31.999925524 fe:54:00:7c:d3:5b                   STP        54 Conf. Root = 32768/0/52:54:00:5d:c8:4d  Cost = 0  Port = 0x8001
   362 34.047985903 fe:54:00:7c:d3:5b                   STP        54 Conf. Root = 32768/0/52:54:00:5d:c8:4d  Cost = 0  Port = 0x8001
   363 36.032931609 fe:54:00:7c:d3:5b                   STP        54 Conf. Root = 32768/0/52:54:00:5d:c8:4d  Cost = 0  Port = 0x8001
   364 38.015990085 fe:54:00:7c:d3:5b                   STP        54 Conf. Root = 32768/0/52:54:00:5d:c8:4d  Cost = 0  Port = 0x8001
   365 39.500218265 192.168.124.148    157.240.196.35   HTTP2     292 HEADERS[15]: POST /login/?privacy_mutation_token=eyJ0eXBlIjowLCJjcmVhdGlvbl90aW1lIjoxNzI2M
   366 39.500349120 192.168.124.148    157.240.196.35   HTTP2     270 DATA[15] (application/x-www-form-urlencoded)
```

Wireshark · Packet 366 · randy-chromium.pcapng

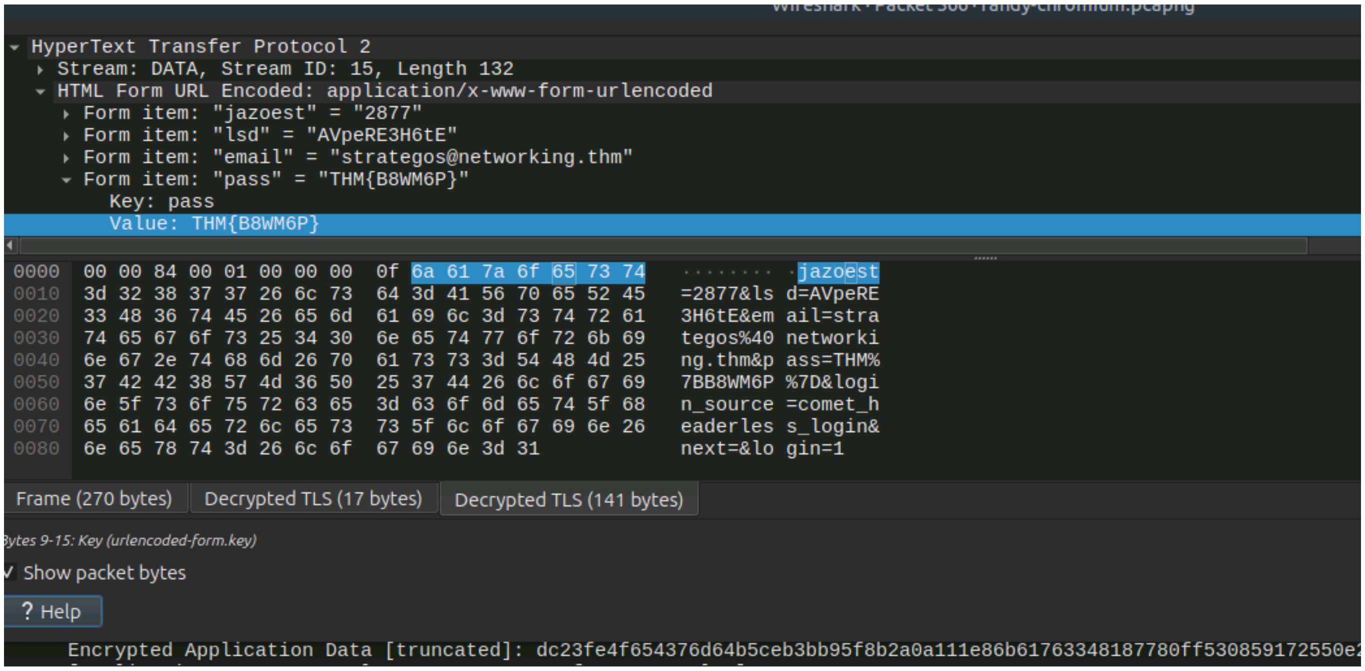
```
        Length: 158
        [Content Type: Application Data (23)]
        Encrypted Application Data [truncated]: dc23fe4f654376d64b5ceb3bb95f8b2a0a111e86b61763348187780ff530859172550e28c82dc3a4b69e1d535847fa57492b52037c77fd8192c3247
        [Application Data Protocol: HyperText Transfer Protocol 2]
  › HyperText Transfer Protocol 2
  ˅ HyperText Transfer Protocol 2
    › Stream: DATA, Stream ID: 15, Length 132
    ˅ HTML Form URL Encoded: application/x-www-form-urlencoded
      › Form item: "jazoest" = "2877"
```

```
0000  00 00 84 00 01 00 00 00  0f 6a 61 7a 6f 65 73 74   ········ ·jazoest
0010  3d 32 38 37 37 26 6c 73  64 3d 41 56 70 65 52 45   =2877&ls d=AVpeRE
0020  33 48 36 74 45 26 65 6d  61 69 6c 3d 73 74 72 61   3H6tE&em ail=stra
0030  74 65 67 6f 73 25 34 30  6e 65 74 77 6f 72 6b 69   tegos%40 networki
0040  6e 67 2e 74 68 6d 26 70  61 73 73 3d 54 48 4d 25   ng.thm&p ass=THM%
0050  37 42 42 38 57 4d 36 50  25 37 44 26 6c 6f 67 69   7BB8WM6P %7D&logi
0060  6e 5f 73 6f 75 72 63 65  3d 63 6f 6d 65 74 5f 68   n_source =comet_h
0070  65 61 64 65 72 6c 65 73  73 5f 6c 6f 67 69 6e 26   eaderles s_login&
0080  6e 65 78 74 3d 26 6c 6f  67 69 6e 3d 31            next=&lo gin=1
```

No.: 366 · Time: 39.500349120 · Source: 192.168.124.148 · Destination: 157.240.196.35 · Protocol: HTTP2 · Length: 270 · Info: DATA[15] (application/x-www-form-urlencoded)

```
Encrypted Application Data [truncated]: dc23fe4f654376d64b5ceb3bb95f8b2a0a111e86b61763348187780ff530859172550e28c82dc3a4b69e1d535847fa57492b52037c77fd8192c3247a3158
```

I then found the packet 336 which had login credentials stored, i had to go into the encrypted application layer further and then go to decrypt tls and it showed the password as highlighted in the screenshot tthen clicking it further revealed the flag



```
˅ HyperText Transfer Protocol 2
  › Stream: DATA, Stream ID: 15, Length 132
  ˅ HTML Form URL Encoded: application/x-www-form-urlencoded
    › Form item: "jazoest" = "2877"
    › Form item: "lsd" = "AVpeRE3H6tE"
    › Form item: "email" = "strategos@networking.thm"
    ˅ Form item: "pass" = "THM{B8WM6P}"
        Key: pass
        Value: THM{B8WM6P}
```

```
0000  00 00 84 00 01 00 00 00  0f 6a 61 7a 6f 65 73 74   ········ ·jazoest
0010  3d 32 38 37 37 26 6c 73  64 3d 41 56 70 65 52 45   =2877&ls d=AVpeRE
0020  33 48 36 74 45 26 65 6d  61 69 6c 3d 73 74 72 61   3H6tE&em ail=stra
0030  74 65 67 6f 73 25 34 30  6e 65 74 77 6f 72 6b 69   tegos%40 networki
0040  6e 67 2e 74 68 6d 26 70  61 73 73 3d 54 48 4d 25   ng.thm&p ass=THM%
0050  37 42 42 38 57 4d 36 50  25 37 44 26 6c 6f 67 69   7BB8WM6P %7D&logi
0060  6e 5f 73 6f 75 72 63 65  3d 63 6f 6d 65 74 5f 68   n_source =comet_h
0070  65 61 64 65 72 6c 65 73  73 5f 6c 6f 67 69 6e 26   eaderles s_login&
0080  6e 65 78 74 3d 26 6c 6f  67 69 6e 3d 31            next=&lo gin=1
```

ytes 9-15: Key (urlencoded-form.key)

```
Encrypted Application Data [truncated]: dc23fe4f654376d64b5ceb3bb95f8b2a0a111e86b61763348187780ff530859172550e
```

****