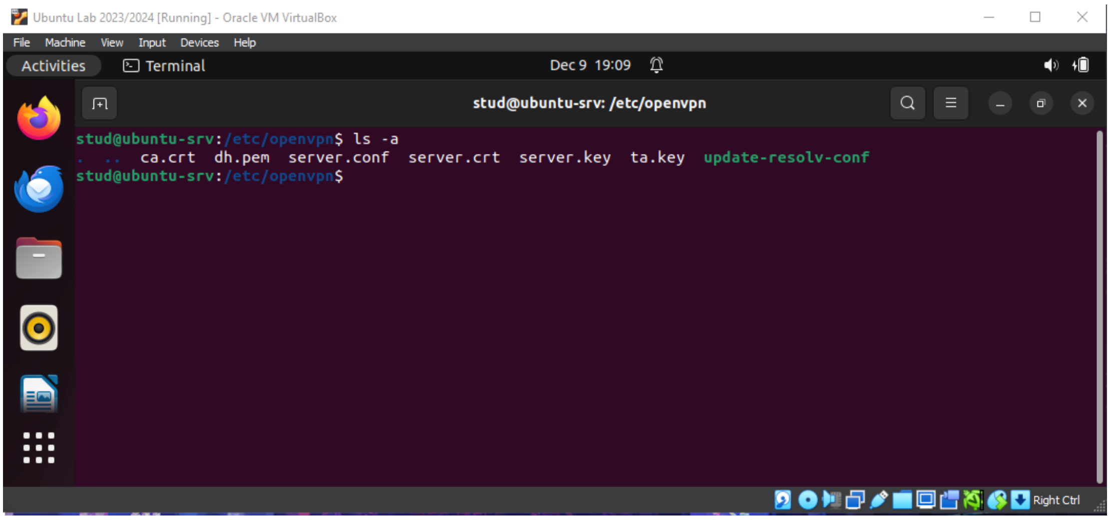
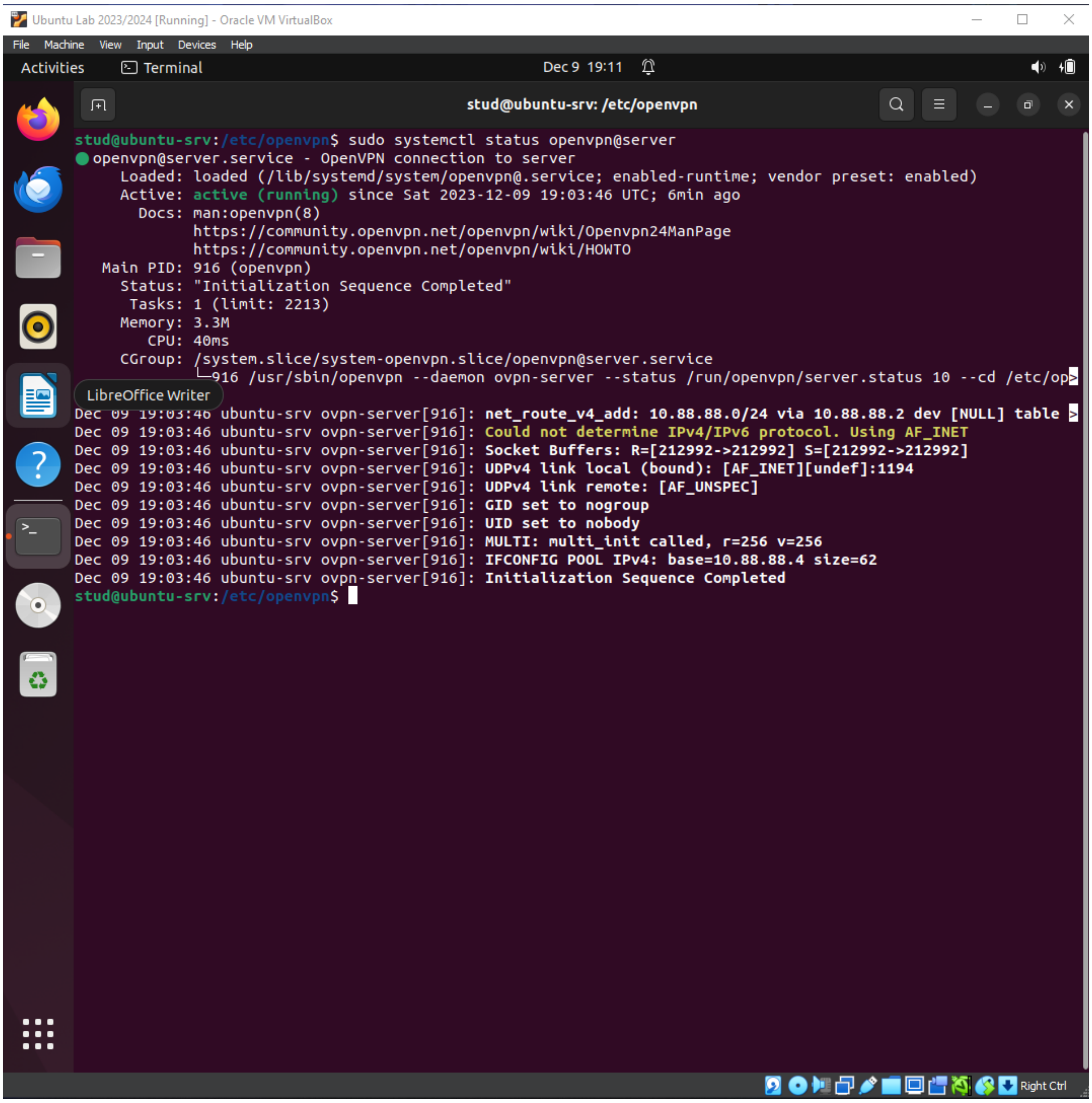
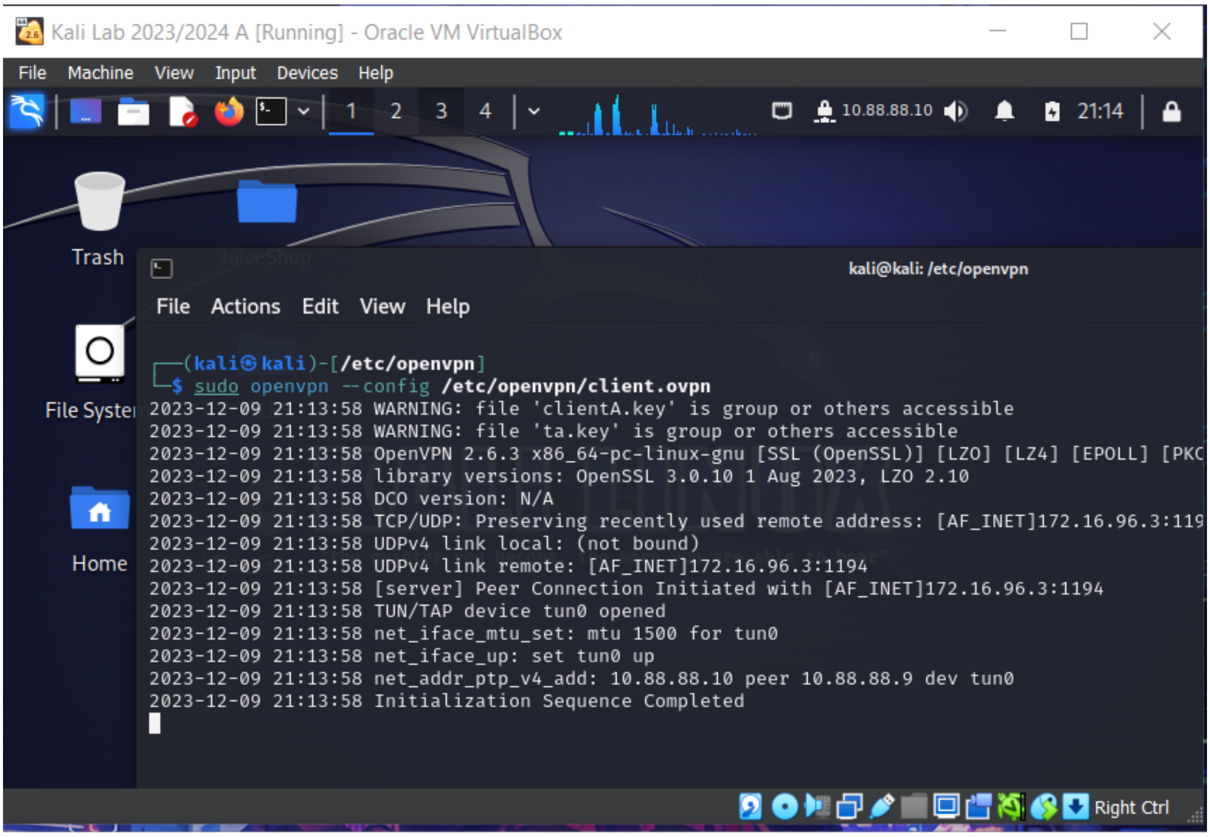
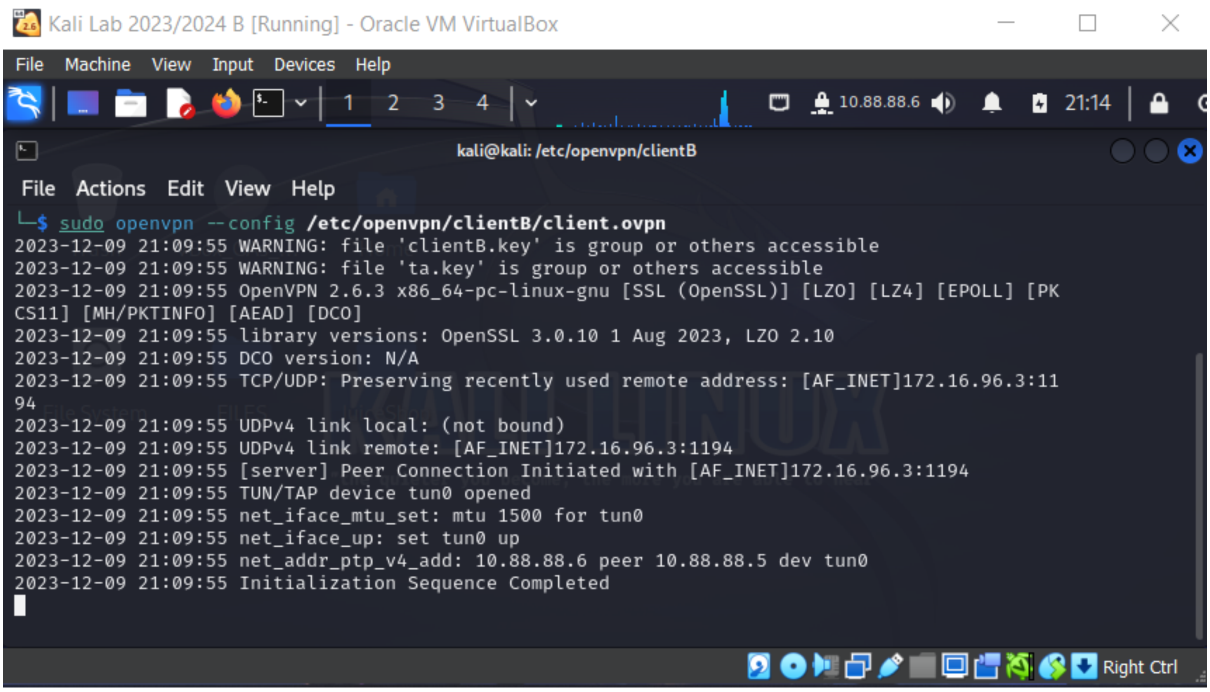
# TASKS 1


Ubuntu Lab 2023/2024 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Dec 9 19:09
stud@ubuntu-srv: /etc/openvpn
stud@ubuntu-srv:/etc/openvpn$ ls -a
. .. ca.crt dh.pem server.conf server.crt server.key ta.key update-resolv-conf
stud@ubuntu-srv:/etc/openvpn$
Right Ctrl

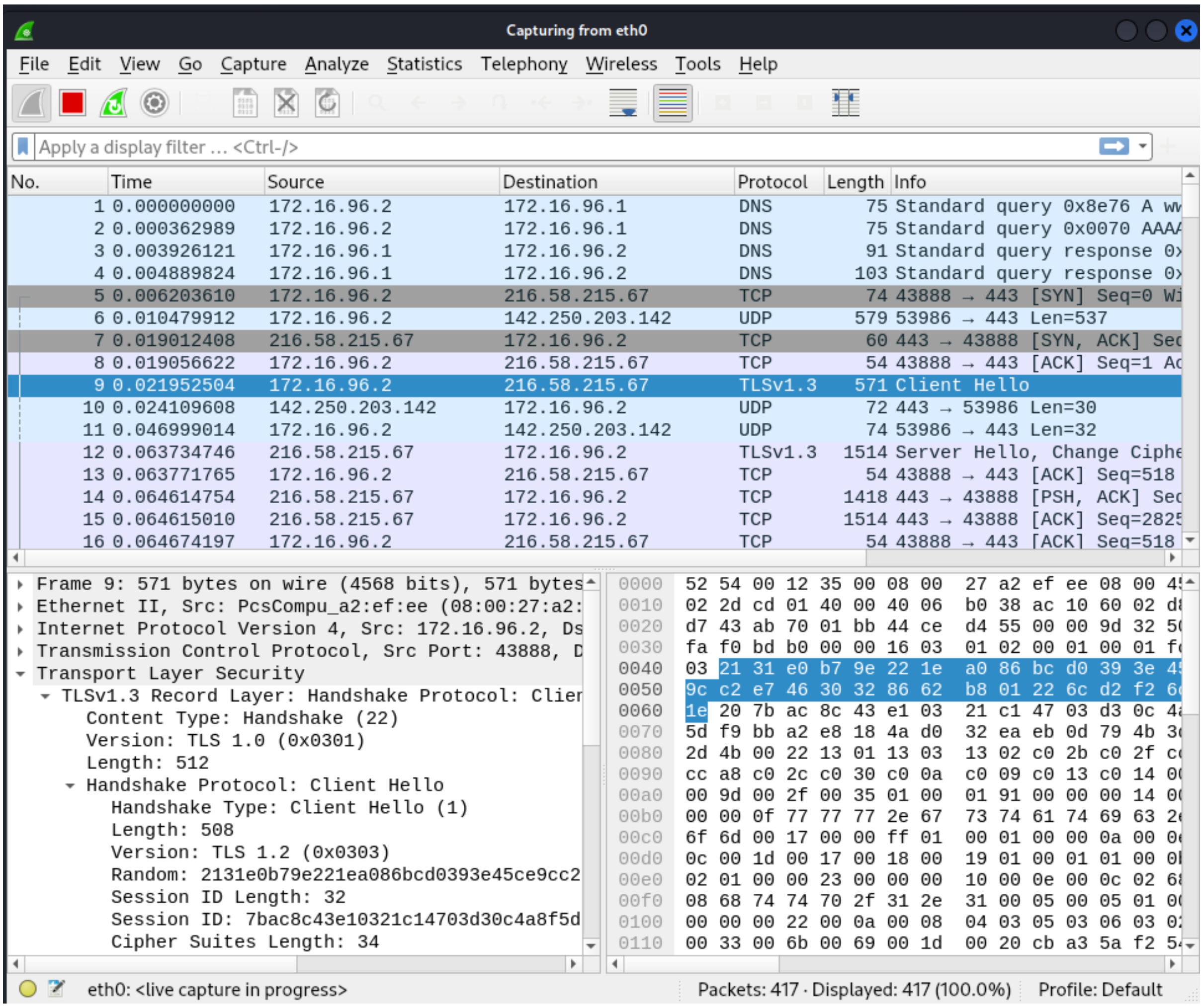

Ubuntu Lab 2023/2024 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities
Terminal
Dec 9 19:11
stud@ubuntu-srv: /etc/openvpn
stud@ubuntu-srv:/etc/openvpn$ sudo systemctl status openvpn@server
openvpn@server.service - OpenVPN connection to server
Loaded: loaded (/lib/systemd/system/openvpn@.service; enabled-runtime; vendor preset: enabled)
Active: active (running) since Sat 2023-12-09 19:03:46 UTC; 6min ago
Docs: man:openvpn(8)
https://community.openvpn.net/openvpn/wiki/Openvpn24ManPage
https://community.openvpn.net/openvpn/wiki/HOWTO
Main PID: 916 (openvpn)
Status: "Initialization Sequence Completed"
Tasks: 1 (limit: 2213)
Memory: 3.3M
CPU: 40ms
CGroup: /system.slice/system-openvpn.slice/openvpn@server.service
└─916 /usr/sbin/openvpn --daemon ovpn-server --status /run/openvpn/server.status 10 --cd /etc/op
LibreOffice Writer
Dec 09 19:03:46 ubuntu-srv ovpn-server[916]: net_route_v4_add: 10.88.88.0/24 via 10.88.88.2 dev [NULL] table
Dec 09 19:03:46 ubuntu-srv ovpn-server[916]: Could not determine IPv4/IPv6 protocol. Using AF_INET
Dec 09 19:03:46 ubuntu-srv ovpn-server[916]: Socket Buffers: R=[212992->212992] S=[212992->212992]
Dec 09 19:03:46 ubuntu-srv ovpn-server[916]: UDPv4 link local (bound): [AF_INET][undef]:1194
Dec 09 19:03:46 ubuntu-srv ovpn-server[916]: UDPv4 link remote: [AF_UNSPEC]
Dec 09 19:03:46 ubuntu-srv ovpn-server[916]: GID set to nogroup
Dec 09 19:03:46 ubuntu-srv ovpn-server[916]: UID set to nobody
Dec 09 19:03:46 ubuntu-srv ovpn-server[916]: MULTI: multi_init called, r=256 v=256
Dec 09 19:03:46 ubuntu-srv ovpn-server[916]: IFCONFIG POOL IPv4: base=10.88.88.4 size=62
Dec 09 19:03:46 ubuntu-srv ovpn-server[916]: Initialization Sequence Completed
stud@ubuntu-srv:/etc/openvpn$
Right Ctrl

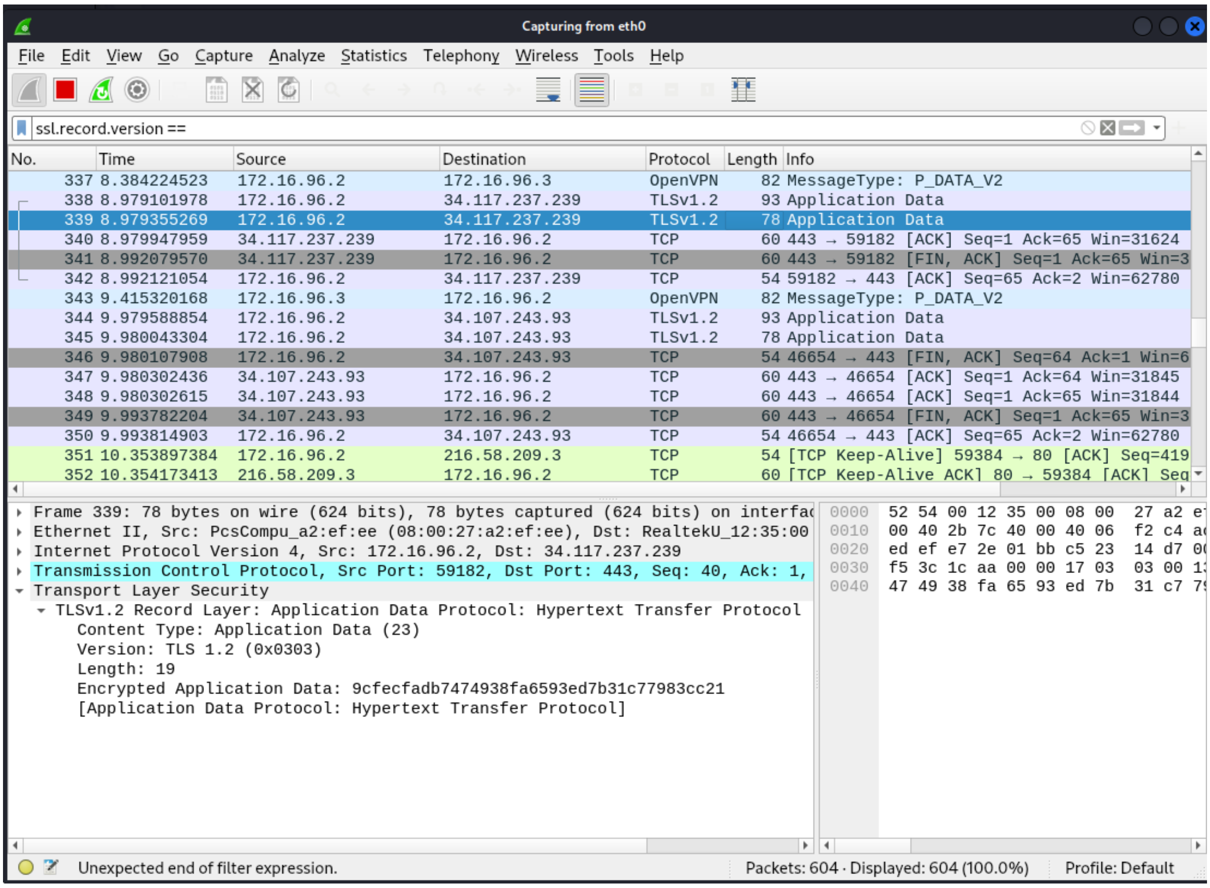
**KALI A : 172.16.96.2/24**
**KALI B : 172.16.96.5/24**
**UBUNTU : 172.16.96.3/24**

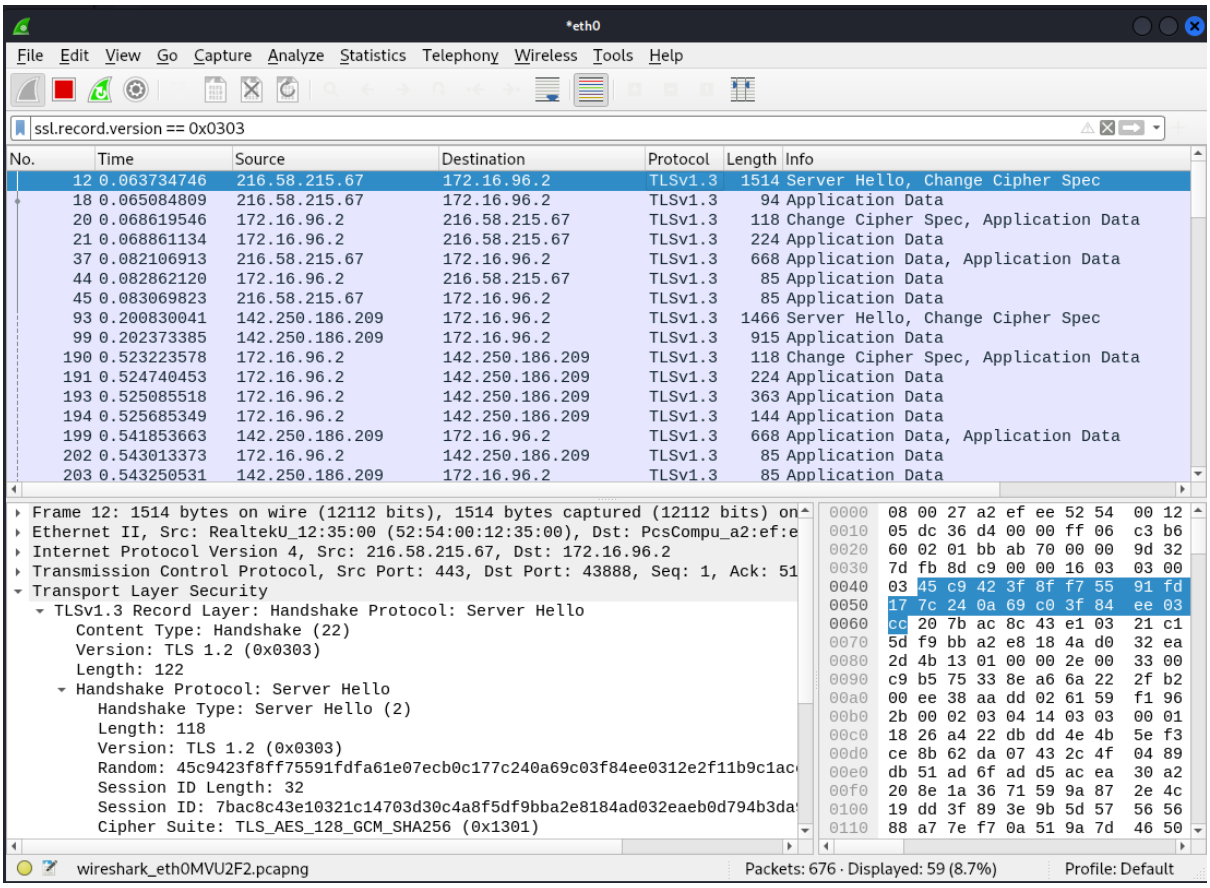


# TASKS 1.1- 1.3




Capturing from eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
ssl.record.version ==
No. Time Source Destination Protocol Length Info
337 8.384224523 172.16.96.2 172.16.96.3 OpenVPN 82 MessageType: P_DATA_V2
338 8.979101978 172.16.96.2 34.117.237.239 TLSv1.2 93 Application Data
339 8.979355269 172.16.96.2 34.117.237.239 TLSv1.2 78 Application Data
340 8.979947959 34.117.237.239 172.16.96.2 TCP 60 443 → 59182 [ACK] Seq=1 Ack=65 Win=31624
341 8.992079570 34.117.237.239 172.16.96.2 TCP 60 443 → 59182 [FIN, ACK] Seq=1 Ack=65 Win=3
342 8.992121054 172.16.96.2 34.117.237.239 TCP 54 59182 → 443 [ACK] Seq=65 Ack=2 Win=62780
343 9.415320168 172.16.96.3 172.16.96.2 OpenVPN 82 MessageType: P_DATA_V2
344 9.979588854 172.16.96.2 34.107.243.93 TLSv1.2 93 Application Data
345 9.980043304 172.16.96.2 34.107.243.93 TLSv1.2 78 Application Data
346 9.980107908 172.16.96.2 34.107.243.93 TCP 54 46654 → 443 [FIN, ACK] Seq=64 Ack=1 Win=6
347 9.980302436 34.107.243.93 172.16.96.2 TCP 60 443 → 46654 [ACK] Seq=1 Ack=64 Win=31845
348 9.980302615 34.107.243.93 172.16.96.2 TCP 60 443 → 46654 [ACK] Seq=1 Ack=65 Win=31844
349 9.993782204 34.107.243.93 172.16.96.2 TCP 60 443 → 46654 [FIN, ACK] Seq=1 Ack=65 Win=3
350 9.993814903 172.16.96.2 34.107.243.93 TCP 54 46654 → 443 [ACK] Seq=65 Ack=2 Win=62780
351 10.353897384 172.16.96.2 216.58.209.3 TCP 54 [TCP Keep-Alive] 59384 → 80 [ACK] Seq=419
352 10.354173413 216.58.209.3 172.16.96.2 TCP 60 [TCP Keep-Alive ACK] 80 → 59384 [ACK] Seq
Frame 339: 78 bytes on wire (624 bits), 78 bytes captured (624 bits) on interfac
Ethernet II, Src: PcsCompu_a2:ef:ee (08:00:27:a2:ef:ee), Dst: RealtekU_12:35:00
Internet Protocol Version 4, Src: 172.16.96.2, Dst: 34.117.237.239
Transmission Control Protocol, Src Port: 59182, Dst Port: 443, Seq: 40, Ack: 1,
Transport Layer Security
TLSv1.2 Record Layer: Application Data Protocol: Hypertext Transfer Protocol
Content Type: Application Data (23)
Version: TLS 1.2 (0x0303)
Length: 19
Encrypted Application Data: 9cfecfadb7474938fa6593ed7b31c77983cc21
[Application Data Protocol: Hypertext Transfer Protocol]
0000 52 54 00 12 35 00 08 00 27 a2 ef
0010 00 40 2b 7c 40 00 40 06 f2 c4 a
0020 ed ef e7 2e 01 bb c5 23 14 d7 0
0030 f5 3c 1c aa 00 00 17 03 03 00 1
0040 47 49 38 fa 65 93 ed 7b 31 c7 7
Unexpected end of filter expression.
Packets: 604 · Displayed: 604 (100.0%)
Profile: Default


*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
ssl.record.version == 0x0303
No. Time Source Destination Protocol Length Info
12 0.063734746 216.58.215.67 172.16.96.2 TLSv1.3 1514 Server Hello, Change Cipher Spec
18 0.065084809 216.58.215.67 172.16.96.2 TLSv1.3 94 Application Data
20 0.068619546 172.16.96.2 216.58.215.67 TLSv1.3 118 Change Cipher Spec, Application Data
21 0.068861134 172.16.96.2 216.58.215.67 TLSv1.3 224 Application Data
37 0.082106913 216.58.215.67 172.16.96.2 TLSv1.3 668 Application Data, Application Data
44 0.082862120 172.16.96.2 216.58.215.67 TLSv1.3 85 Application Data
45 0.083069823 216.58.215.67 172.16.96.2 TLSv1.3 85 Application Data
93 0.200830041 142.250.186.209 172.16.96.2 TLSv1.3 1466 Server Hello, Change Cipher Spec
99 0.202373385 142.250.186.209 172.16.96.2 TLSv1.3 915 Application Data
190 0.523223578 172.16.96.2 142.250.186.209 TLSv1.3 118 Change Cipher Spec, Application Data
191 0.524740453 172.16.96.2 142.250.186.209 TLSv1.3 224 Application Data
193 0.525085518 172.16.96.2 142.250.186.209 TLSv1.3 363 Application Data
194 0.525685349 172.16.96.2 142.250.186.209 TLSv1.3 144 Application Data
199 0.541853663 142.250.186.209 172.16.96.2 TLSv1.3 668 Application Data, Application Data
202 0.543013373 172.16.96.2 142.250.186.209 TLSv1.3 85 Application Data
203 0.543250531 142.250.186.209 172.16.96.2 TLSv1.3 85 Application Data
Frame 12: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on
Ethernet II, Src: RealtekU_12:35:00 (52:54:00:12:35:00), Dst: PcsCompu_a2:ef:e
Internet Protocol Version 4, Src: 216.58.215.67, Dst: 172.16.96.2
Transmission Control Protocol, Src Port: 443, Dst Port: 43888, Seq: 1, Ack: 51
Transport Layer Security
TLSv1.3 Record Layer: Handshake Protocol: Server Hello
Content Type: Handshake (22)
Version: TLS 1.2 (0x0303)
Length: 122
Handshake Protocol: Server Hello
Handshake Type: Server Hello (2)
Length: 118
Version: TLS 1.2 (0x0303)
Random: 45c9423f8ff75591fdfa61e07ecb0c177c240a69c03f84ee0312e2f11b9c1ac
Session ID Length: 32
Session ID: 7bac8c43e10321c14703d30c4a8f5df9bba2e8184ad032eaeb0d794b3da
Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
0000 08 00 27 a2 ef ee 52 54 00 12
0010 05 dc 36 d4 00 00 ff 06 c3 b6
0020 60 02 01 bb ab 70 00 00 9d 32
0030 7d fb 8d c9 00 00 16 03 03 00
0040 03 45 c9 42 3f 8f f7 55 91 fd
0050 17 7c 24 0a 69 c0 3f 84 ee 03
0060 cc 20 7b ac 8c 43 e1 03 21 c1
0070 5d f9 bb a2 e8 18 4a d0 32 ea
0080 2d 4b 13 01 00 00 2e 00 33 00
0090 c9 b5 75 33 8e a6 6a 22 2f b2
00a0 00 ee 38 aa dd 02 61 59 f1 96
00b0 2b 00 02 03 04 14 03 03 00 01
00c0 18 26 a4 22 db dd 4e 4b 5e f3
00d0 ce 8b 62 da 07 43 2c 4f 04 89
00e0 db 51 ad 6f ad d5 ac ea 30 a2
00f0 20 8e 1a 36 71 59 9a 87 2e 4c
0100 19 dd 3f 89 3e 9b 5d 57 56 56
0110 88 a7 7e f7 0a 51 9a 7d 46 50
wireshark_eth0MVU2F2.pcapng
Packets: 676 · Displayed: 59 (8.7%)
Profile: Default

## TASK 1.4

While **TLS** and **SSL** are related cryptographic protocols designed for secure communication over a network, they are not exactly the same. SSL was the original protocol developed by Netscape in the 1990s for internet communication security. However, due to discovered vulnerabilities, SSL underwent revisions.

TLS, on the other hand, is an updated and more secure version of SSL, with versions like TLS 1.0, TLS 1.1, TLS 1.2, and TLS 1.3. Essentially, TLS serves the same purpose as SSL but is considered more secure, and it is the protocol currently used for securing web traffic and various applications.

## TASK 1.5

The TLS/SSL handshake is a crucial process at the start of a secure communication session between a client (e.g., a web browser) and a server. This process establishes the parameters for secure communication, including encryption algorithms and keys, ensuring both parties agree on the terms of the secure connection. The handshake lays the foundation for a secure and encrypted data exchange.

Here's a breakdown of the TLS/SSL handshake:

- **Initiation** (ClientHello): The client begins by sending a message called ClientHello to the server, conveying information about its supported cryptographic algorithms, TLS/SSL versions, and other parameters.

- **Response** (ServerHello): The server responds with ServerHello, selecting the strongest cryptographic algorithms and the highest supported version of TLS/SSL.

- **Key Exchange**: The server sends its public key to the client. This step may involve the server's certificate, used to verify the authenticity of the server's public key. The client generates a pre-master secret, encrypts it with the server's public key, and sends it back.

- **Shared Secret** (Pre-Master Secret): Both the client and server independently create the shared secret known as the pre-master secret. This secret is never transmitted over the network in its raw form but is used to derive symmetric keys for encrypting and decrypting data.

- **Confirmation** (Finished): Both parties confirm the completion of the handshake and readiness for encrypted communication. They exchange a "Finished" message, signaling that subsequent data will be encrypted using the negotiated parameters.

Once the TLS/SSL handshake concludes, the client and server can securely exchange data over the encrypted connection. The derived encryption keys safeguard the confidentiality and integrity of the transmitted information.

## TASK 1.6

The TLS/SSL protocol provides secure communication over a computer network by ensuring the confidentiality and integrity of the transmitted data. It achieves this through encryption and authentication mechanisms. Here are two examples of applications where the TLS/SSL protocol is commonly used:

- Secure Web Browsing (HTTPS):

  **Description**: One of the most well-known applications of TLS/SSL is securing web browsing through HTTPS (Hypertext Transfer Protocol Secure). When you connect to a website using HTTPS, the TLS/SSL protocol encrypts the data exchanged between your web browser and the server, preventing unauthorized access or tampering.

  **Example**: When you access your online banking account, make an e-commerce transaction, or log in to a secure email service, the TLS/SSL protocol ensures that the sensitive information you send and receive, such as login credentials, financial details, or personal messages, is encrypted and protected.

- Email Communication (SMTPS, IMAPS):

  **Description**: TLS/SSL is often employed to secure email communication. Protocols like SMTPS (Secure SMTP) for sending emails and IMAPS (Internet Message Access Protocol Secure) for retrieving emails use TLS/SSL to encrypt the data exchanged between email clients and servers. This ensures the confidentiality of email content and protects against eavesdropping.

  **Example**: When you configure your email client (e.g., Microsoft Outlook or Mozilla Thunderbird) to use a secure connection for sending and receiving emails, the TLS/SSL protocol is employed. This is especially crucial when dealing with sensitive or confidential information via email.

## TASK 1.7

TLS 1.3 is acknowledged as the most secure and broadly embraced iteration of the TLS/SSL protocol. It introduces advancements in security, performance, and privacy when contrasted with its forerunners. Numerous websites and services are actively making the switch to TLS 1.3 to leverage its improved features.

In addition, TLS 1.2 enjoys widespread usage and support. Although not as contemporary as TLS 1.3, it maintains a commendable standard of security and compatibility with a diverse array of clients and servers. TLS 1.2 has held its position as the predominant version for several years.

## TASK 1.8

TLS 1.3 is generally recognized as providing heightened security compared to TLS 1.2, and several factors contribute to this assessment:

- **Enhanced Cipher Suites**: TLS 1.3 has done away with older, less secure cipher suites found in TLS 1.2. It employs modern cryptographic algorithms, bolstering the overall security of the protocol.
- **Mandatory Perfect Forward Secrecy (PFS):** TLS 1.3 mandates the use of Perfect Forward Secrecy as an integral design feature. This ensures that even if a long-term secret key is compromised, past communications remain secure. In TLS 1.2, PFS support was optional and contingent on the configuration.
- **Simplified Handshake Complexity:** TLS 1.3 has streamlined and optimized the handshake process, diminishing the attack surface and enhancing security. This streamlined process also reduces the likelihood of vulnerabilities associated with the handshake.
- **Elimination of Legacy Features:** TLS 1.3 removes several legacy features and insecure options present in TLS 1.2, thereby decreasing potential attack vectors.
- **Defence Against Known Attacks**: TLS 1.3 addresses vulnerabilities and known attacks applicable to earlier versions, offering a more resilient defence against potential threats.

## TASK 1.9

TLS 1.3 generally exhibits superior performance compared to TLS 1.2, with enhancements attributed to several key factors:

- **Reduced Handshake Latency:** TLS 1.3 has significantly minimized the latency associated with the handshake process. The design of the TLS 1.3 handshake prioritizes speed and efficiency, leading to a swifter establishment of secure connections.
- **0-RTT (Zero Round-Trip Time) Handshake Mode**: Introducing a 0-RTT handshake mode, TLS 1.3 allows clients with a prior connection to a server to resume communication without a complete handshake. This minimizes the round-trip time for subsequent connections, particularly benefiting frequently visited websites.
- **Optimized Cipher Suites**: TLS 1.3 has eliminated older, less effective cipher suites found in TLS 1.2. The utilization of contemporary and more efficient cryptographic algorithms contributes to an overall improvement in performance.
- **Parallelism in Key Exchange**: TLS 1.3 facilitates the execution of the key exchange process in parallel with other handshake steps, promoting concurrency and reducing the time needed to establish a secure connection.
- **Smaller and More Efficient Protocol Design**: With a focus on simplicity and efficiency, TLS 1.3 has been crafted to eliminate unnecessary complexities present in TLS 1.2. This streamlined design significantly contributes to improved performance.

## TASK 1.10

Wireshark has recorded two versions of the protocol: TLS 1.2 and TLS 1.3

## TASK 1.11

**ClientHello**:

The client initiates the handshake, sending parameters like supported protocol versions, encryption algorithms, and data compression methods.

**ServerHello**:

The server responds by selecting connection parameters and sending them back to the client.

**Certificate**:

The server provides its certificate to allow the client to verify its identity.

**ServerKeyExchange**:

The server shares information about its public key, with the type and length determined by the algorithm sent in the ServerKeyExchange message.

**ServerHelloDone**:

The server signals its readiness for the client's response.

**ClientKeyExchange**:

The client sends the initial session key encrypted with the server's public key. Both parties use established parameters to generate the session key for data exchange.

**ChangeCipherSpec** (Client):

The client notifies the server that communication should proceed with the parameters set in the previous messages.

**Finished** (Client):

The client signals readiness to receive encrypted data.

**ChangeCipherSpec** (Server):

The server notifies that it will send only encrypted data from this point forward.

**Finished** (Server):

A message confirming the successful handshake process, sent securely through the established encrypted channel.

The TLS/SSL protocol version employed in a connection relies on the negotiation process between the client and server. This determination is influenced by various factors:

- **Client and Server Capabilities**:

  Both the client and server have a spectrum of supported TLS/SSL versions. In the initial handshake phase (ClientHello and ServerHello messages), they communicate their respective supported versions.

- **Highest Common Version**:

  The TLS/SSL protocol version selected for the connection is the highest common version supported by both the client and server. This information is typically included in the ServerHello message.

- **Fallback Mechanism**:

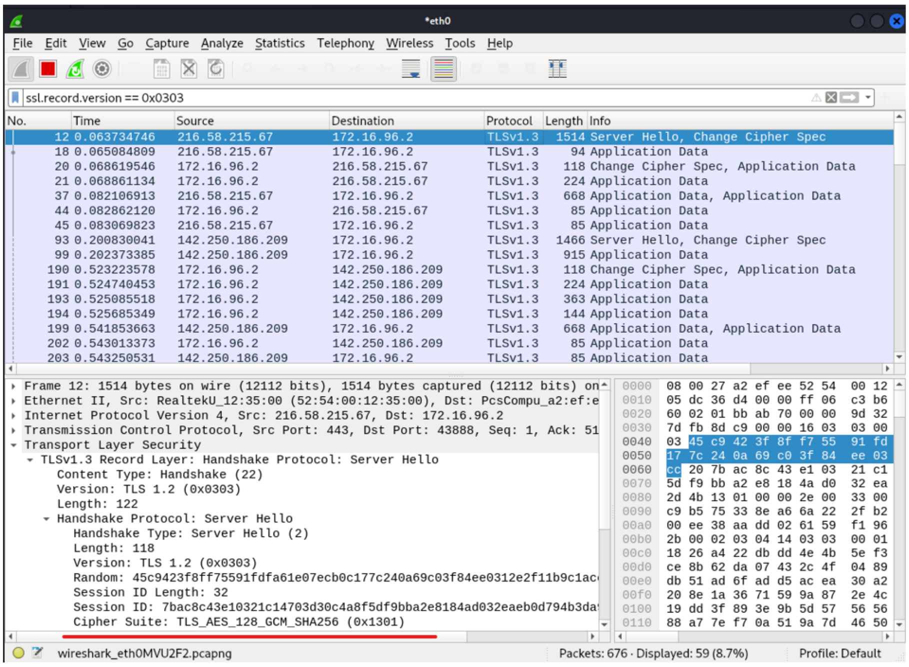
  In instances where a consensus on a common TLS/SSL version is not reached, a fallback mechanism may be employed. The client may attempt negotiation with a lower version if the higher version is not supported.

- **Configuration Settings**:

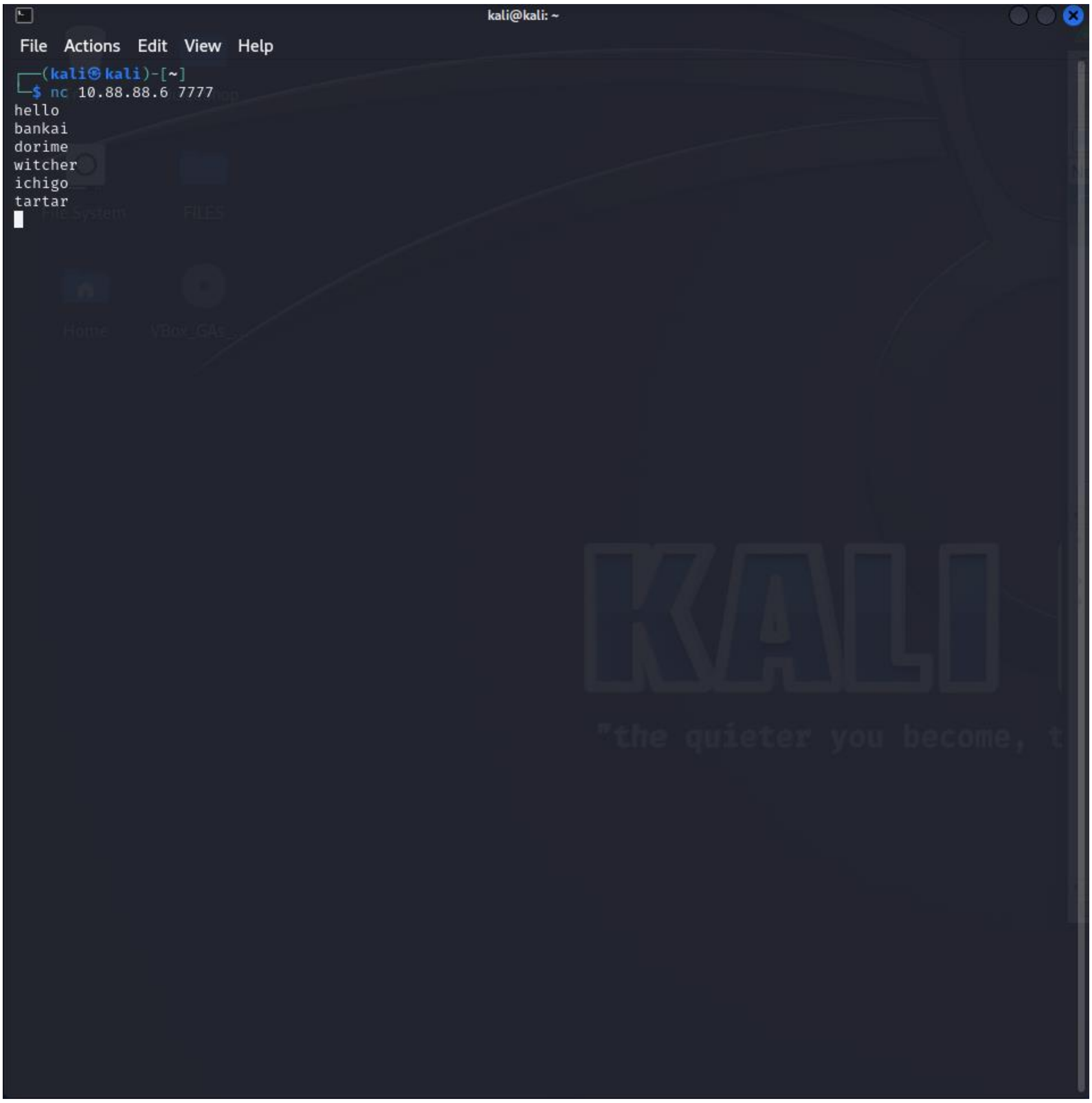
  The configuration settings on both the client and server play a role. System administrators can configure preferences for specific TLS/SSL versions based on compatibility or security considerations.

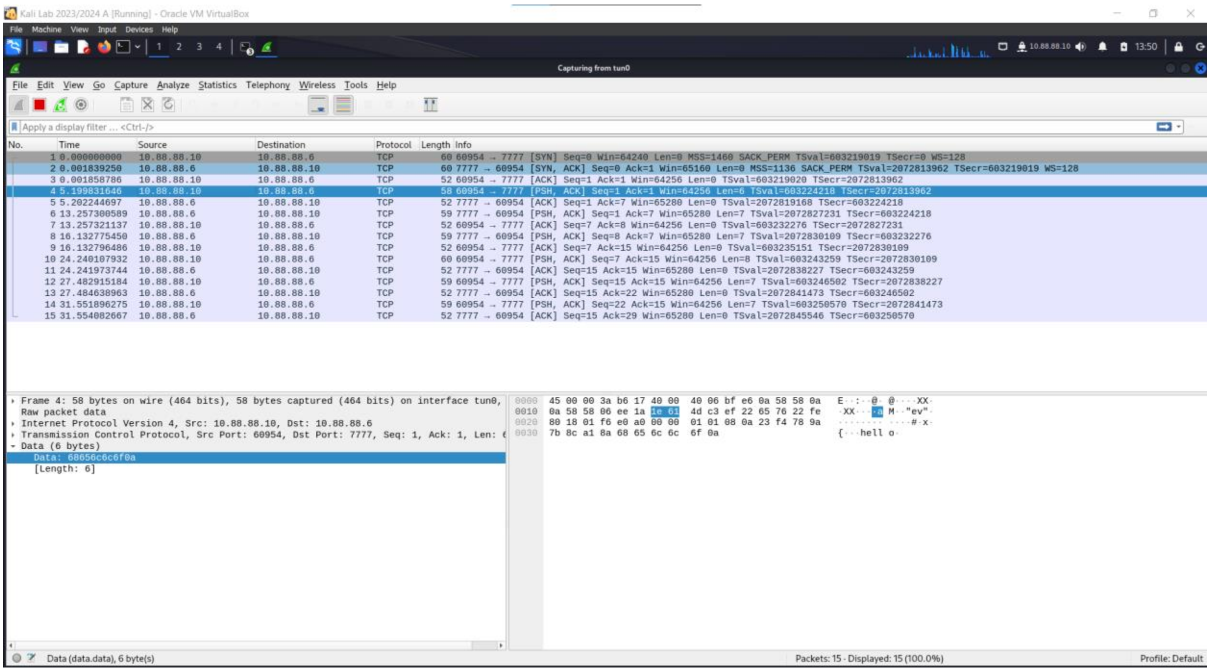
- **Protocol Negotiation Extensions**:

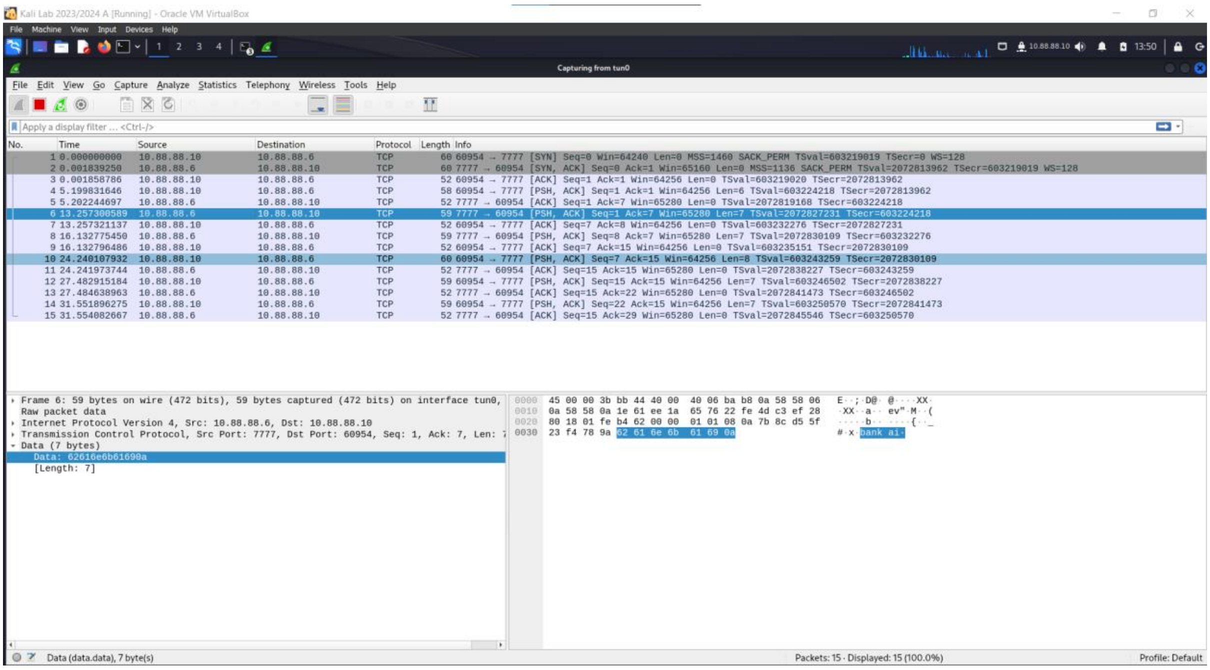
  Extensions within the TLS/SSL protocol can influence version negotiation. For instance, the "Supported Versions" extension enables the client to explicitly indicate its supported versions.

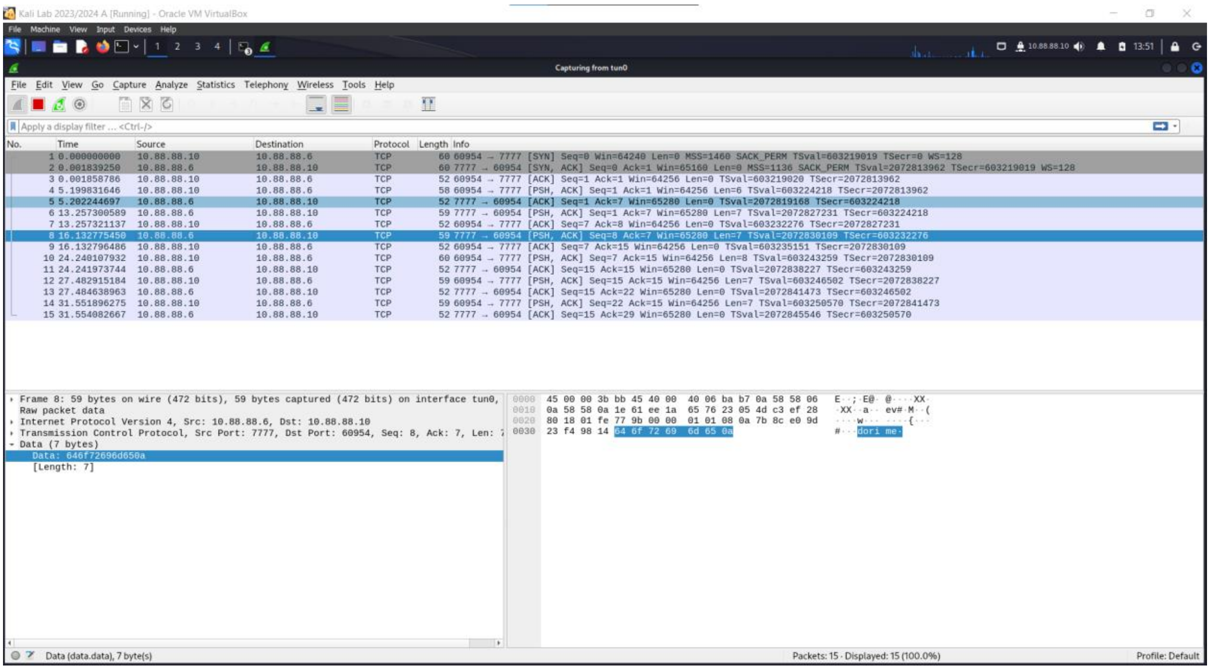


# TASKS 2


kali@kali: ~
File Actions Edit View Help
(kali㉿kali)-[~]
$ nc 10.88.88.6 7777
hello
bankai
dorime
witcher
ichigo
tartar
KALI


Kali Lab 2023/2024 A [Running] - Oracle VM VirtualBox
Capturing from tun0
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000000 10.88.88.10 10.88.88.6 TCP 60 60954 → 7777 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=603219019 TSecr=0 WS=128
2 0.001839250 10.88.88.6 10.88.88.10 TCP 60 7777 → 60954 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1136 SACK_PERM TSval=2072813962 TSecr=603219019 WS=128
3 0.001858786 10.88.88.10 10.88.88.6 TCP 52 60954 → 7777 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=603219020 TSecr=2072813962
4 5.199831646 10.88.88.10 10.88.88.6 TCP 58 60954 → 7777 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=6 TSval=603224218 TSecr=2072813962
5 5.202244697 10.88.88.6 10.88.88.10 TCP 52 7777 → 60954 [ACK] Seq=1 Ack=7 Win=65280 Len=0 TSval=2072819168 TSecr=603224218
6 13.257300589 10.88.88.6 10.88.88.10 TCP 59 7777 → 60954 [PSH, ACK] Seq=1 Ack=7 Win=65280 Len=7 TSval=2072827231 TSecr=603224218
7 13.257321137 10.88.88.10 10.88.88.6 TCP 52 60954 → 7777 [ACK] Seq=7 Ack=8 Win=64256 Len=0 TSval=603232276 TSecr=2072827231
8 16.132775450 10.88.88.6 10.88.88.10 TCP 59 7777 → 60954 [PSH, ACK] Seq=8 Ack=7 Win=65280 Len=7 TSval=2072830109 TSecr=603232276
9 16.132796486 10.88.88.10 10.88.88.6 TCP 52 60954 → 7777 [ACK] Seq=7 Ack=15 Win=64256 Len=0 TSval=603235151 TSecr=2072830109
10 24.240107932 10.88.88.10 10.88.88.6 TCP 60 60954 → 7777 [PSH, ACK] Seq=7 Ack=15 Win=64256 Len=8 TSval=603243259 TSecr=2072830109
11 24.241973744 10.88.88.6 10.88.88.10 TCP 52 7777 → 60954 [ACK] Seq=15 Ack=15 Win=65280 Len=0 TSval=2072838227 TSecr=603243259
12 27.482915184 10.88.88.10 10.88.88.6 TCP 59 60954 → 7777 [PSH, ACK] Seq=15 Ack=15 Win=64256 Len=7 TSval=603246502 TSecr=2072838227
13 27.484638963 10.88.88.6 10.88.88.10 TCP 52 7777 → 60954 [ACK] Seq=15 Ack=22 Win=65280 Len=0 TSval=2072841473 TSecr=603246502
14 31.551896275 10.88.88.10 10.88.88.6 TCP 59 60954 → 7777 [PSH, ACK] Seq=22 Ack=15 Win=64256 Len=7 TSval=603250570 TSecr=2072841473
15 31.554082667 10.88.88.6 10.88.88.10 TCP 52 7777 → 60954 [ACK] Seq=15 Ack=29 Win=65280 Len=0 TSval=2072845546 TSecr=603250570
Frame 4: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface tun0,
Raw packet data
Internet Protocol Version 4, Src: 10.88.88.10, Dst: 10.88.88.6
Transmission Control Protocol, Src Port: 60954, Dst Port: 7777, Seq: 1, Ack: 1, Len:
Data (6 bytes)
Data: 68656c6c6f0a
[Length: 6]
Data (data.data), 6 byte(s)
Packets: 15 · Displayed: 15 (100.0%)
Profile: Default


Kali Lab 2023/2024 A [Running] - Oracle VM VirtualBox
Capturing from tun0
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
1 0.000000000 10.88.88.10 10.88.88.6 TCP 60 60954 → 7777 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=603219019 TSecr=0 WS=128
2 0.001839250 10.88.88.6 10.88.88.10 TCP 60 7777 → 60954 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1136 SACK_PERM TSval=2072813962 TSecr=603219019 WS=128
3 0.001858786 10.88.88.10 10.88.88.6 TCP 52 60954 → 7777 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=603219020 TSecr=2072813962
4 5.199831646 10.88.88.10 10.88.88.6 TCP 58 60954 → 7777 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=6 TSval=603224218 TSecr=2072813962
5 5.202244697 10.88.88.6 10.88.88.10 TCP 52 7777 → 60954 [ACK] Seq=1 Ack=7 Win=65280 Len=0 TSval=2072819168 TSecr=603224218
6 13.257300589 10.88.88.6 10.88.88.10 TCP 59 7777 → 60954 [PSH, ACK] Seq=1 Ack=7 Win=65280 Len=7 TSval=2072827231 TSecr=603224218
7 13.257321137 10.88.88.10 10.88.88.6 TCP 52 60954 → 7777 [ACK] Seq=7 Ack=8 Win=64256 Len=0 TSval=603232276 TSecr=2072827231
8 16.132775450 10.88.88.6 10.88.88.10 TCP 59 7777 → 60954 [PSH, ACK] Seq=8 Ack=7 Win=65280 Len=7 TSval=2072830109 TSecr=603232276
9 16.132796486 10.88.88.10 10.88.88.6 TCP 52 60954 → 7777 [ACK] Seq=7 Ack=15 Win=64256 Len=0 TSval=603235151 TSecr=2072830109
10 24.240107932 10.88.88.10 10.88.88.6 TCP 60 60954 → 7777 [PSH, ACK] Seq=7 Ack=15 Win=64256 Len=8 TSval=603243259 TSecr=2072830109
11 24.241973744 10.88.88.6 10.88.88.10 TCP 52 7777 → 60954 [ACK] Seq=15 Ack=15 Win=65280 Len=0 TSval=2072838227 TSecr=603243259
12 27.482915184 10.88.88.10 10.88.88.6 TCP 59 60954 → 7777 [PSH, ACK] Seq=15 Ack=15 Win=64256 Len=7 TSval=603246502 TSecr=2072838227
13 27.484638963 10.88.88.6 10.88.88.10 TCP 52 7777 → 60954 [ACK] Seq=15 Ack=22 Win=65280 Len=0 TSval=2072841473 TSecr=603246502
14 31.551896275 10.88.88.10 10.88.88.6 TCP 59 60954 → 7777 [PSH, ACK] Seq=22 Ack=15 Win=64256 Len=7 TSval=603250570 TSecr=2072841473
15 31.554082667 10.88.88.6 10.88.88.10 TCP 52 7777 → 60954 [ACK] Seq=15 Ack=29 Win=65280 Len=0 TSval=2072845546 TSecr=603250570
Frame 6: 59 bytes on wire (472 bits), 59 bytes captured (472 bits) on interface tun0,
Raw packet data
Internet Protocol Version 4, Src: 10.88.88.6, Dst: 10.88.88.10
Transmission Control Protocol, Src Port: 7777, Dst Port: 60954, Seq: 1, Ack: 7, Len:
Data (7 bytes)
Data: 62616e6b61690a
[Length: 7]
Data (data.data), 7 byte(s)
Packets: 15 · Displayed: 15 (100.0%)
Profile: Default

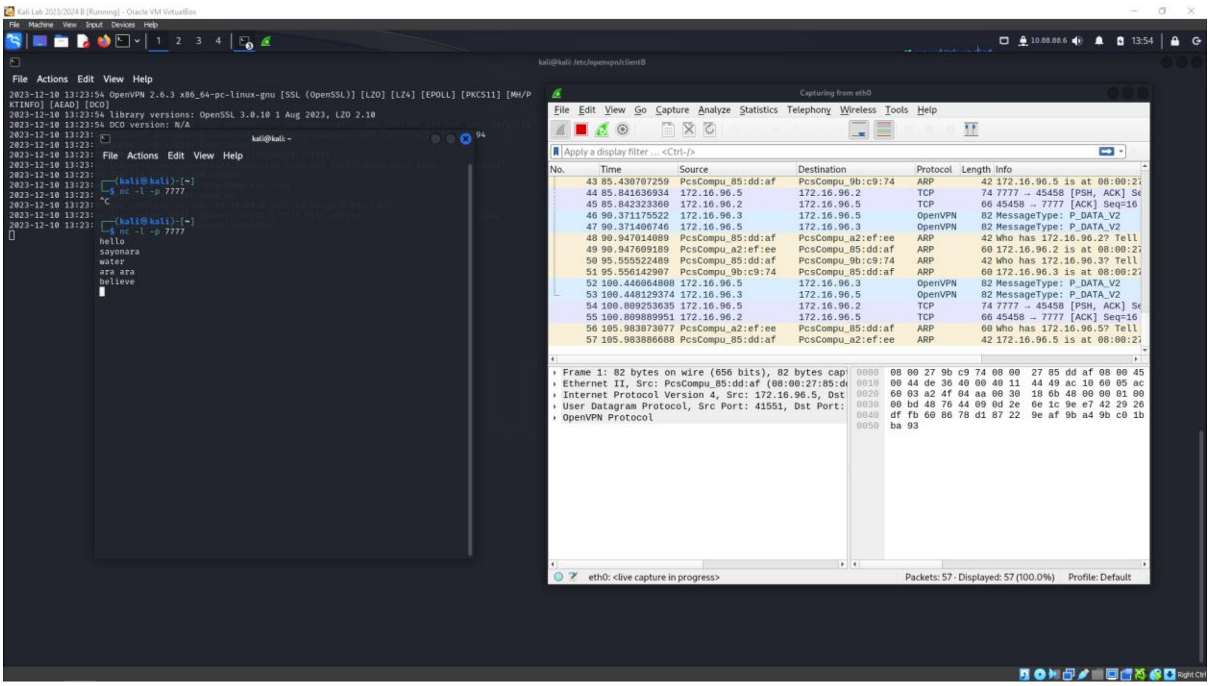


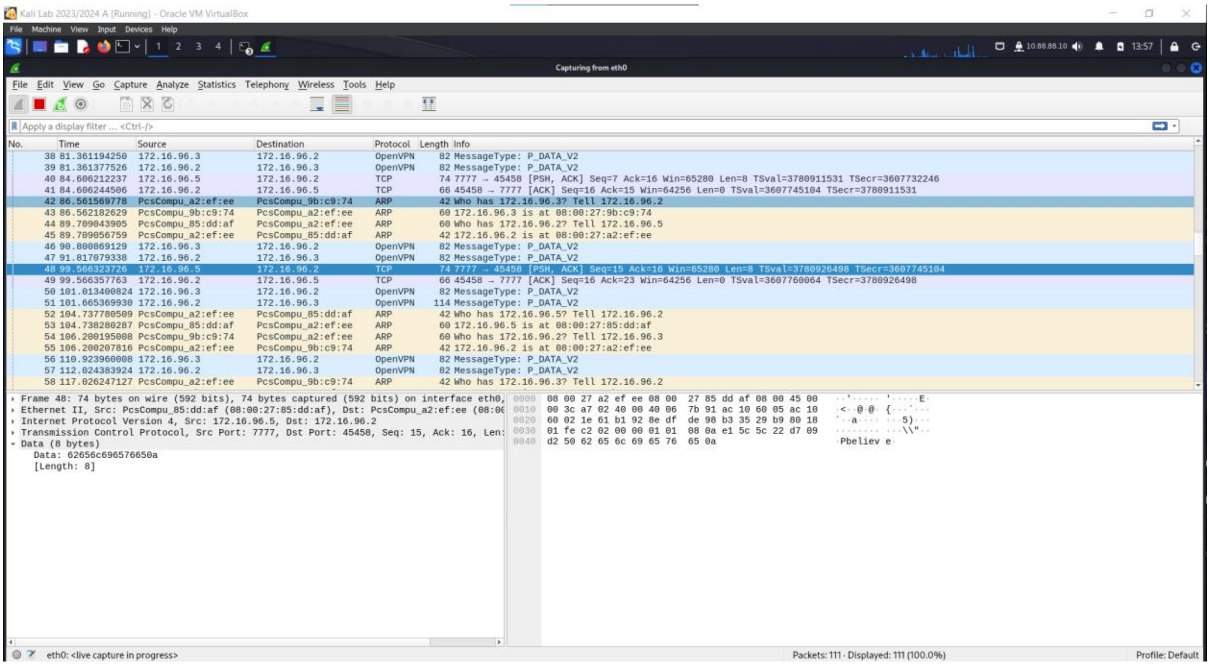
TCP packets sent between the machines on the VPN interface (tun0) allowed the contents of the transmitted message to be read from the packet.

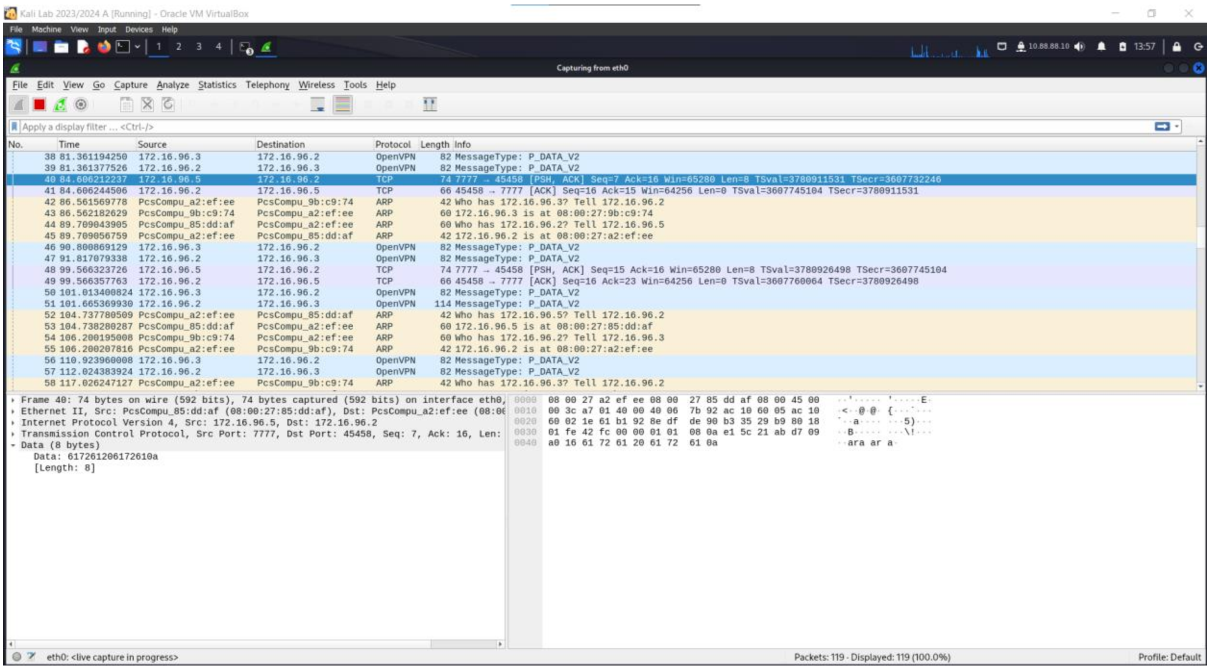
## TASKS 3

## TASK 3.1 – 3.4

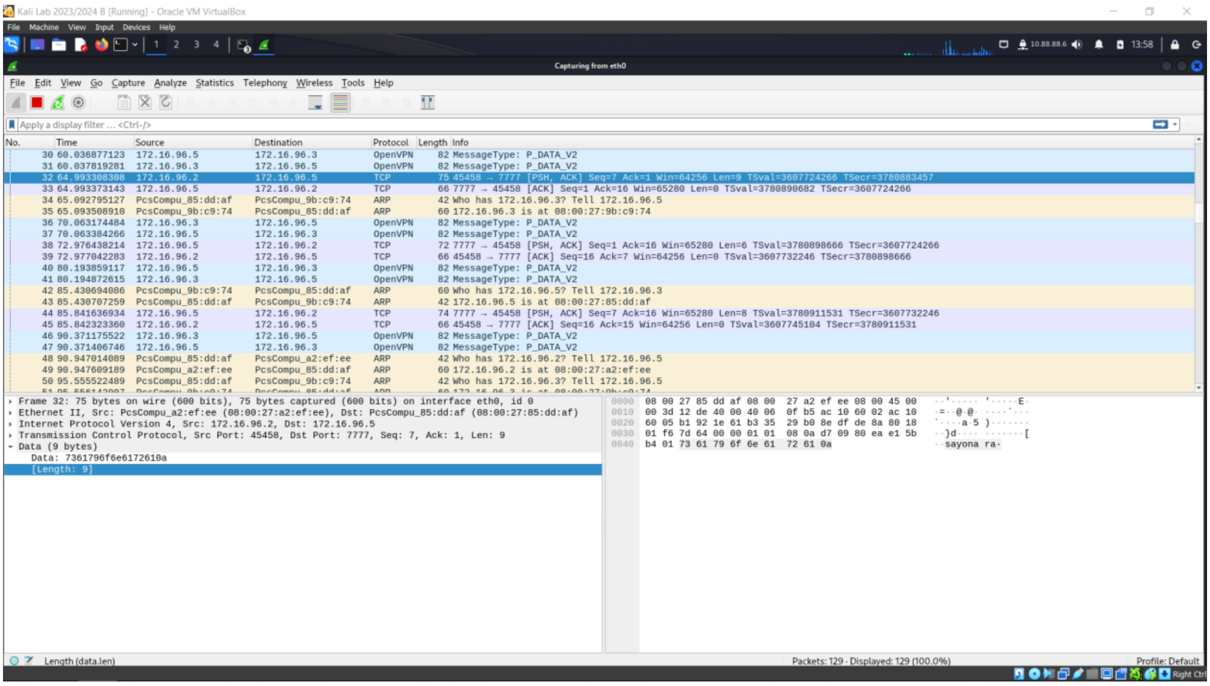
KALI A:

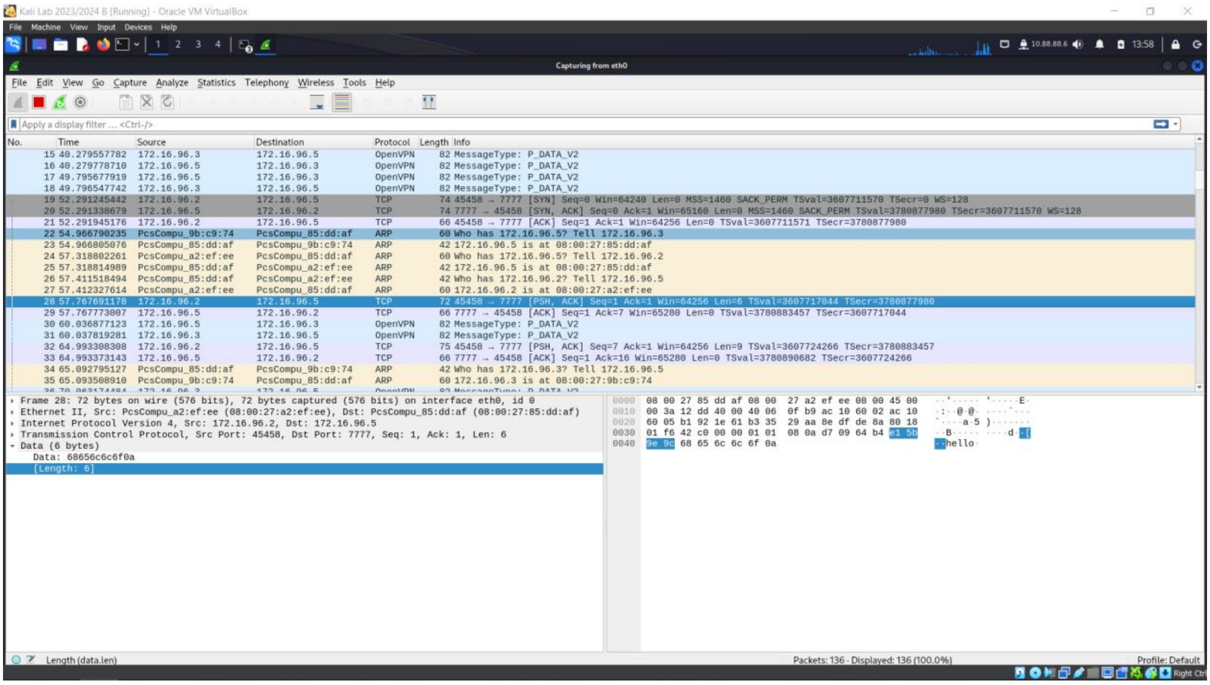



Kali Lab 2023/2024 A [Running] - Oracle VM VirtualBox
Capturing from eth0
Frame 48: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface eth0,
Ethernet II, Src: PcsCompu_85:dd:af (08:00:27:85:dd:af), Dst: PcsCompu_a2:ef:ee
Internet Protocol Version 4, Src: 172.16.96.5, Dst: 172.16.96.2
Transmission Control Protocol, Src Port: 7777, Dst Port: 45458, Seq: 15, Ack: 16, Len:
Data (8 bytes)
Data: 62656c696576650a
[Length: 8]
0000 08 00 27 a2 ef ee 08 00 27 85 dd af 08 00 45 00
0010 00 3c a7 02 40 00 40 06 7b 91 ac 10 60 05 ac 10
0020 60 02 1e 61 b1 92 8e df de 98 b3 35 29 b9 80 18
0030 01 fe c2 02 00 00 01 01 08 0a e1 5c 5c 22 d7 09
0040 d2 50 62 65 6c 69 65 76 65 0a
Pbeliev e
eth0: <live capture in progress>
Packets: 111 · Displayed: 111 (100.0%)
Profile: Default


Kali Lab 2023/2024 A [Running] - Oracle VM VirtualBox
Capturing from eth0
Frame 40: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface eth0,
Ethernet II, Src: PcsCompu_85:dd:af (08:00:27:85:dd:af), Dst: PcsCompu_a2:ef:ee
Internet Protocol Version 4, Src: 172.16.96.5, Dst: 172.16.96.2
Transmission Control Protocol, Src Port: 7777, Dst Port: 45458, Seq: 7, Ack: 16, Len:
Data (8 bytes)
Data: 617261206172610a
[Length: 8]
0000 08 00 27 a2 ef ee 08 00 27 85 dd af 08 00 45 00
0010 00 3c a7 01 40 00 40 06 7b 92 ac 10 60 05 ac 10
0020 60 02 1e 61 b1 92 8e df de 90 b3 35 29 b9 80 18
0030 01 fe 42 fc 00 00 01 01 08 0a e1 5c 21 ab d7 09
0040 a0 16 61 72 61 20 61 72 61 0a
ara ar a
eth0: <live capture in progress>
Packets: 119 · Displayed: 119 (100.0%)
Profile: Default

KALI B:





TCP packets sent between the machines on the eth0 interface allowed the contents of the transmitted message to be read from the packet.

## TASK 3.5

When intercepting traffic from the VPN interface, it was possible to read the message directly from the packet in unencrypted form. However, when analyzing traffic from the physical interface (e.g., eth0), OpenVPN packets were visible being sent between machines, but the data contained within these packets was in encrypted form. It was impossible to directly read the message from the contents of the OpenVPN packets. The content was visible in TCP packets, and the unencrypted message could be directly read from the contents of these TCP packets.

## TASK 3.6

In the scenario involving traffic on the VPN interface, the packets were explicitly assigned the correct source and destination addresses. The sender was aware of the intended recipient, and the receiver had information about the sender's identity.

However, when intercepting packets during analysis of the host interface, only the address of the originating machine and the server were evident. In this context, the recipient lacked the means to determine the address from which the message originated, and the sender remained uninformed about the identity of the recipient.

## TASK 3.7

In VPN traffic, the source and destination addresses, along with the plaintext content of the message, are exposed. Conversely, on the physical interface, only the addresses of the sender and the server are identifiable, while the message's text remains viewable.

# TASKS 4

## Task 4.1

Task 4.1 is the same as 3.1

***ubuntu.iso*** *is empty so debian-12.1.0-amd64-netinst.iso will be used for transfer*

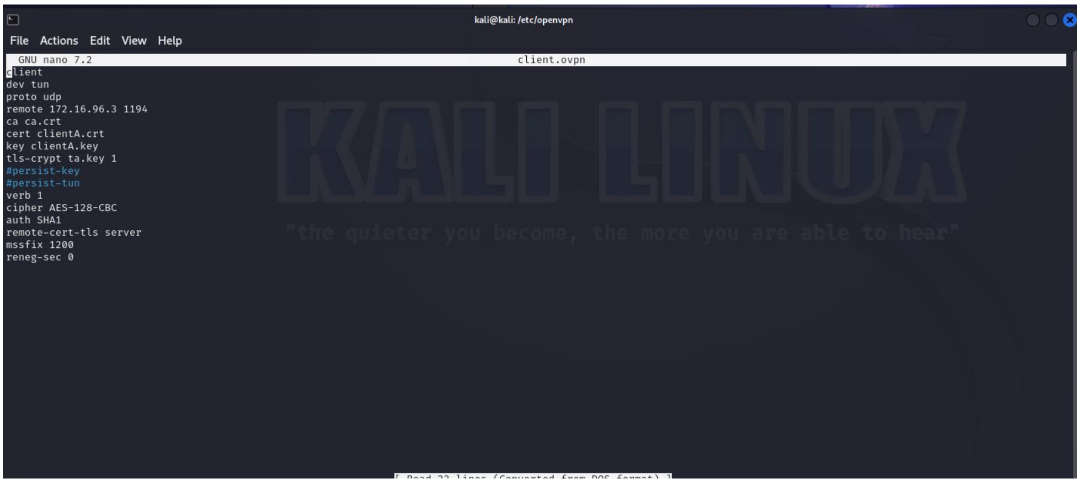
**SETTING 1: 3:37**

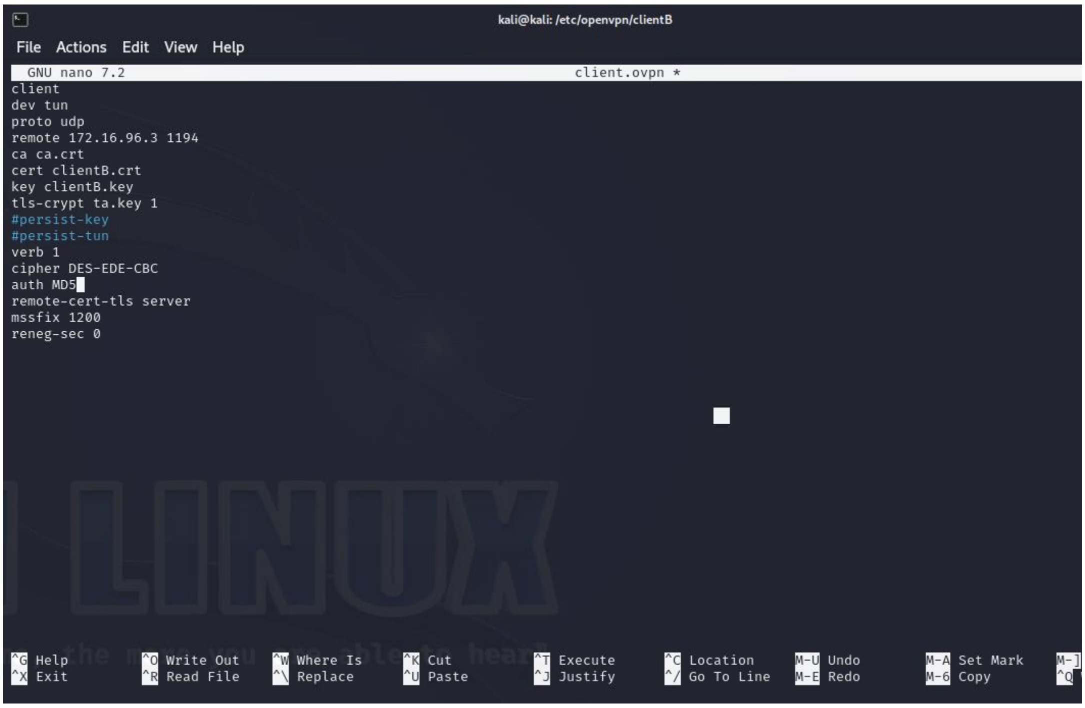
**SETTING 2 : 3:34**

**SETTING 3 : 3:50**

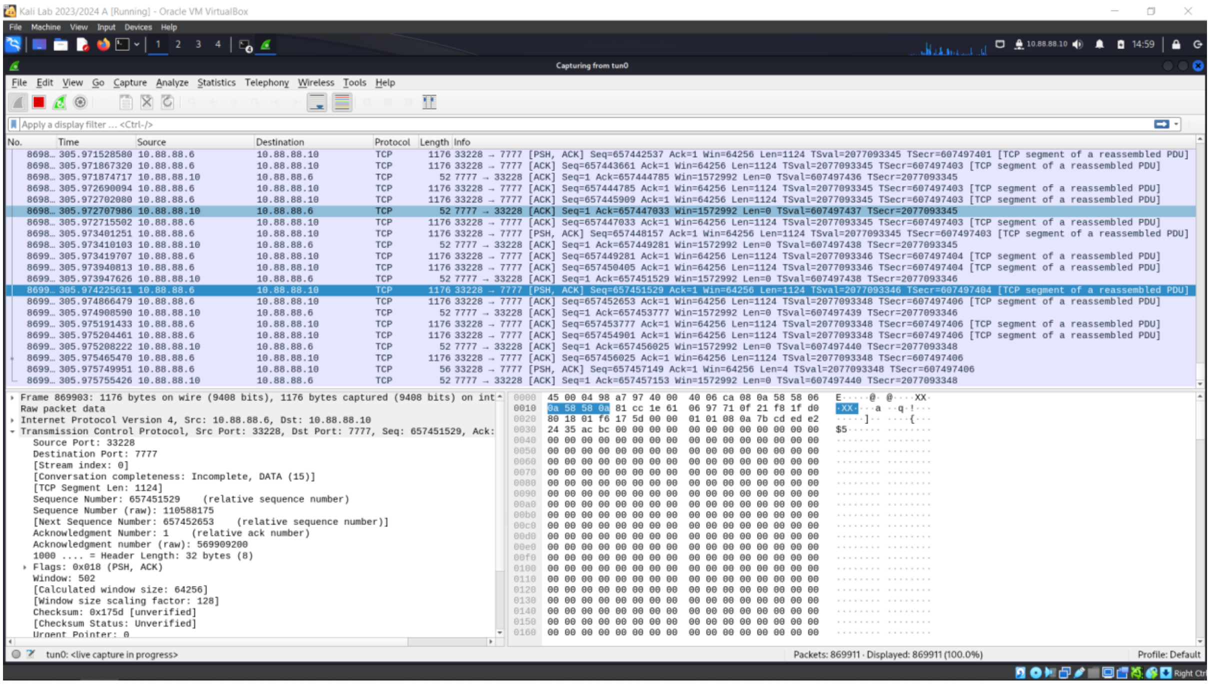
**SETTING 4 : 3:52**

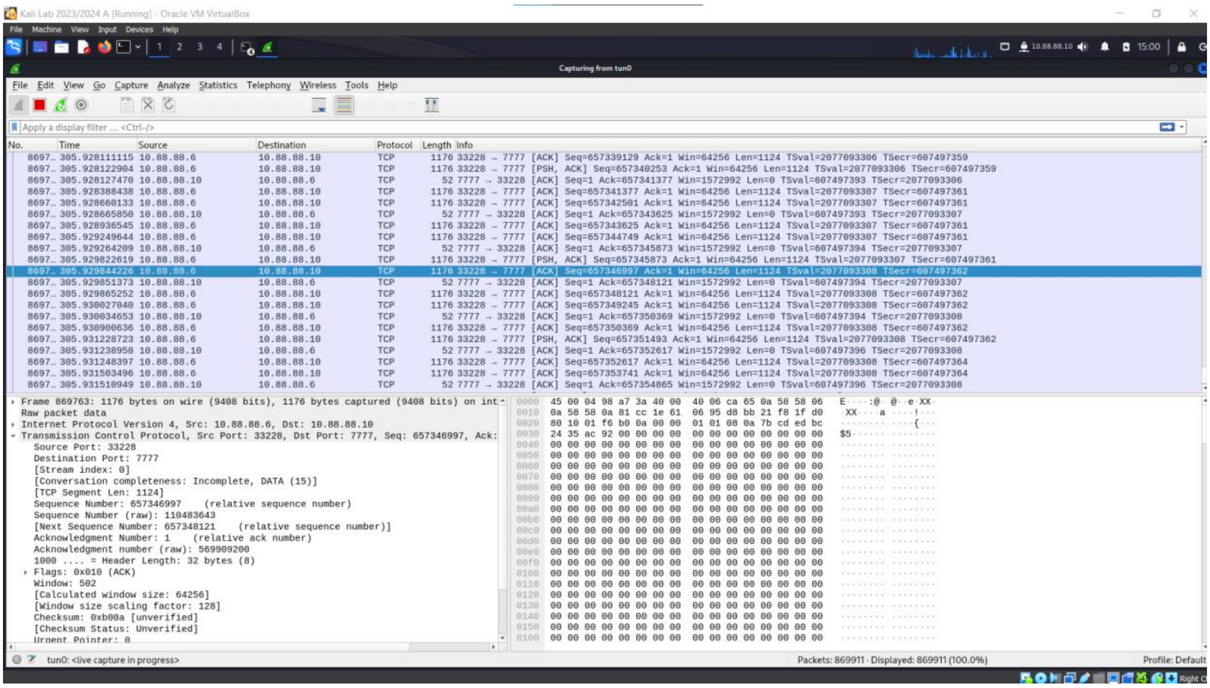
Different settings:




The communication process:





## TASK 4.4

There exists a marginal variance in data transfer durations. Nevertheless, this difference is sufficiently negligible to deduce that the selection of a specific algorithm has minimal impact on file transfer times. Discrepancies in recorded times could potentially arise from concurrent processes running on the machine, occasionally impeding the transfer process.

## TASK 4.5

**Encryption Algorithms**:

**AES-256-GCM**:

Strengths: This is a highly secure encryption algorithm. AES-256 provides strong confidentiality, and GCM (Galois/Counter Mode) adds authenticated encryption, ensuring both privacy and data integrity.

Considerations: This is considered one of the most secure symmetric encryption algorithms available.

**AES-128-CBC**:

Strengths: AES-128 is a strong encryption algorithm, providing a good level of confidentiality.

Considerations: While secure, it is not as robust as AES-256. AES in CBC mode does not provide authenticated encryption; additional measures may be needed for data integrity.

**DES-EDE-CBC**:

Strengths: DES (Data Encryption Standard) was once considered strong, but it is now deprecated due to its vulnerability to brute-force attacks.

Considerations: Triple DES (DES-EDE) involves applying DES three times. However, it is considered slow and less secure compared to modern algorithms. The use of DES-EDE-CBC is not recommended for strong security.

**DES-CBC**:

Strengths: Similar to DES-EDE-CBC, DES-CBC is deprecated and insecure against modern cryptographic attacks.

Considerations: DES is considered broken and unsuitable for secure communication. Its use is strongly discouraged.

**Authentication Algorithm**s:

**SHA512**:

Strengths: SHA-512 is a secure hash function, providing a high level of data integrity.

Considerations: It is considered a robust choice for secure authentication and is widely used.

**SHA1**:

Strengths: SHA-1 was once widely used, but it is now deprecated due to vulnerabilities.

Considerations: SHA-1 is susceptible to collision attacks, making it less secure. It is not recommended for secure authentication.

**MD5:**

Strengths: MD5 was historically used for integrity checking, but it is now considered insecure.

Considerations: MD5 is vulnerable to collision attacks, and its use for authentication is strongly discouraged due to security weaknesses.

## TASK 4.6

The encryption and authentication algorithms used in the provided settings vary in terms of security:

**Strong Security:**

In configurations where AES-256-GCM is employed for encryption and SHA512 for authentication, the security level is considered high. These algorithms are currently robust and provide a strong foundation for secure communication.

**Moderate Security:**

AES-128-CBC, while still secure, is not as robust as AES-256. SHA1, used for authentication in one of the settings, is considered weak due to vulnerabilities. While the encryption remains moderate, the security is compromised by the use of SHA1.

**Weak Security:**

The use of outdated encryption algorithms such as DES-EDE-CBC and DES-CBC, coupled with MD5 for authentication, reflects weak security. These algorithms have known vulnerabilities and are not recommended for secure communication.

**Overall Assessment:**

Prioritize configurations with strong encryption (e.g., AES-256) and robust authentication algorithms (e.g., SHA512) for optimal security.

Avoid configurations with weaker encryption (e.g., DES) and deprecated or insecure authentication algorithms (e.g., MD5) to mitigate potential security risks.