

# MySQL TLS Lab

**Date:** August 8 2025

**Machine:** Ubuntu Server (VM) on Windows Host

**Objective:** Capture and analyze MySQL traffic over an encrypted TLS connection.

## Overview

The goal was to set up a MySQL server on my Ubuntu VM with TLS encryption, connect to it securely using a custom user, and capture the traffic with `tcpdump` to analyze whether the connection protects sensitive data (like credentials and queries). I also wanted to copy the `.pcap` file to my Windows system for analysis in Wireshark.

## 1. SSH Setup

To avoid the copy-paste limitation in the VM terminal, I enabled SSH so I could connect from my Windows CMD or PowerShell.

### Commands used:

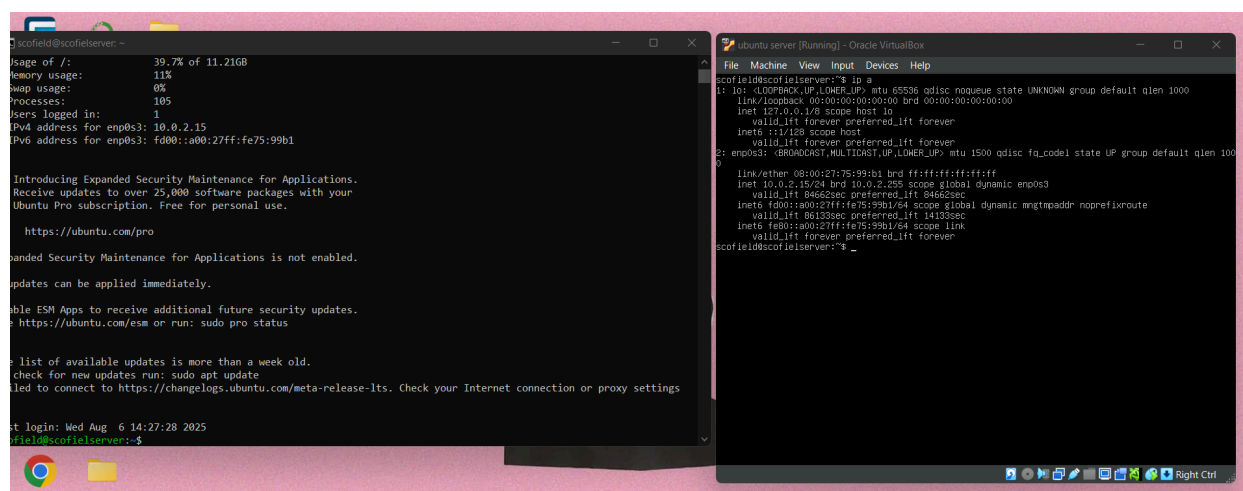
```
sudo apt update
```

```
sudo apt install openssh-server -y
```

```
sudo systemctl enable ssh
```

```
sudo systemctl start ssh
```

### Screenshot: SSH server running status



## Issue Encountered:

Initially, I was confused about copying and pasting in CMD. I realized I had to enable **QuickEdit Mode** or use **Windows Terminal**, which made the process much easier.

## 2. MySQL Installation and TLS Setup

I installed MySQL using:

```
sudo apt install mysql-server -y
```

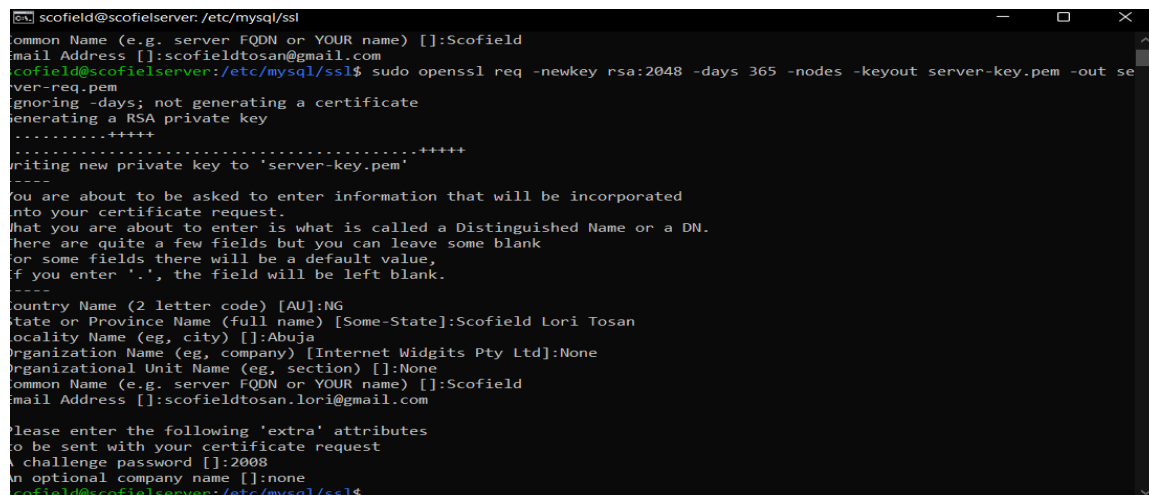
Then I generated custom TLS certificates using **openssl**:

```
sudo openssl genrsa 2048 | sudo tee ca-key.pem > /dev/null
sudo openssl req -new -x509 -nodes -days 365 -key ca-key.pem -out ca.pem
sudo openssl req -newkey rsa:2048 -days 365 -nodes -keyout server-key.pem -out
server-req.pem
sudo openssl x509 -req -in server-req.pem -days 365 -CA ca.pem -CAkey ca-key.pem
-set_serial 01 -out server-cert.pem
```

I moved them to **/etc/mysql/ssl** and fixed permissions:

```
sudo chmod 600 *.pem
sudo chown mysql:mysql *.pem
```

## Screenshot: Certificate files in **/etc/mysql/ssl**



```
scofield@scofieldserver: /etc/mysql/ssl
Common Name (e.g. server FQDN or YOUR name) []:Scofield
Email Address []:scofieldtosan@gmail.com
scofield@scofieldserver:/etc/mysql/ssl$ sudo openssl req -newkey rsa:2048 -days 365 -nodes -keyout server-key.pem -out se
rver-req.pem
Ignoring -days; not generating a certificate
Generating a RSA private key
.....+++++
.....+++++
Writing new private key to 'server-key.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:NG
State or Province Name (full name) [Some-State]:Scofield Lori Tosan
Locality Name (eg, city) []:Abuja
Organization Name (eg, company) [Internet Widgits Pty Ltd]:None
Organizational Unit Name (eg, section) []:None
Common Name (e.g. server FQDN or YOUR name) []:Scofield
Email Address []:scofieldtosan.lori@gmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:2008
An optional company name []:none
scofield@scofieldserver:/etc/mysql/ssl$
```

### 3. MySQL Configuration

I updated the MySQL config file:

```
sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
```

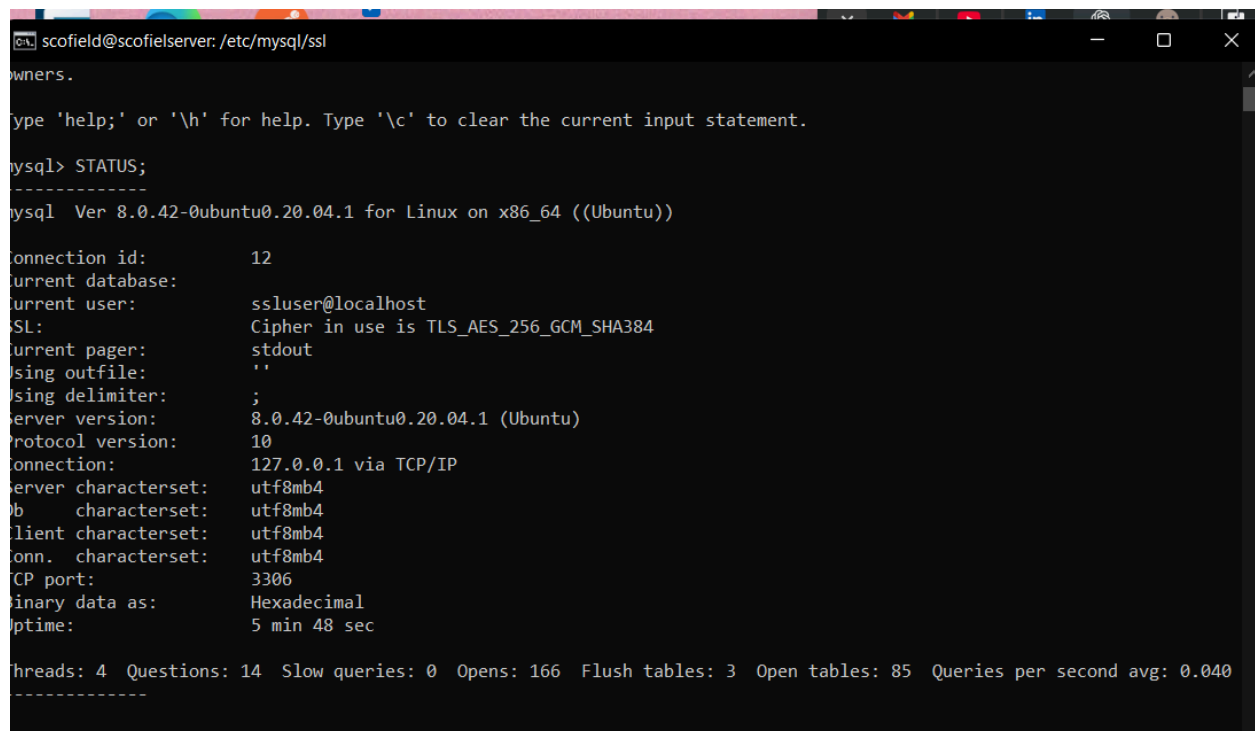
And added:

```
ssl-ca=/etc/mysql/ssl/ca.pem  
ssl-cert=/etc/mysql/ssl/server-cert.pem  
ssl-key=/etc/mysql/ssl/server-key.pem  
require_secure_transport = ON
```

Then restarted MySQL:

```
sudo systemctl restart mysql
```

#### Screenshot: mysqld.cnf file with SSL lines

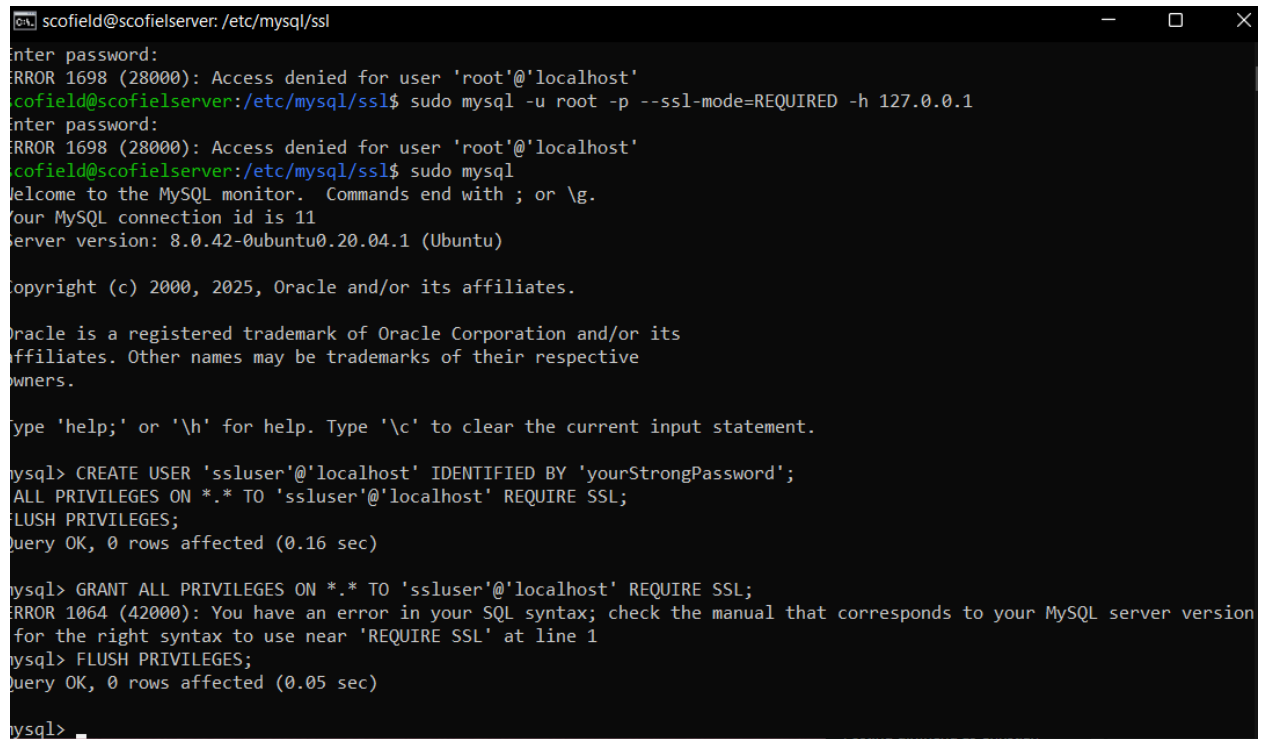
A screenshot of a terminal window titled 'scofield@scofieldserver: /etc/mysql/ssl'. The terminal shows the output of the 'mysql> STATUS;' command. The output displays various MySQL status metrics, including connection ID, current database, user, SSL cipher, pager, outfile, delimiter, server version, protocol version, connection details, character sets, TCP port, binary data format, and uptime. At the bottom, it shows thread counts, questions, slow queries, opens, flush tables, open tables, and queries per second average.

```
scofield@scofieldserver: /etc/mysql/ssl  
mysql> STATUS;  
-----  
mysql Ver 8.0.42-0ubuntu0.20.04.1 for Linux on x86_64 ((Ubuntu))  
  
Connection id:          12  
Current database:         
Current user:           ssluser@localhost  
SSL:                    Cipher in use is TLS_AES_256_GCM_SHA384  
Current pager:          stdout  
Using outfile:          ''  
Using delimiter:        ;  
Server version:         8.0.42-0ubuntu0.20.04.1 (Ubuntu)  
Protocol version:       10  
Connection:             127.0.0.1 via TCP/IP  
Server character set:   utf8mb4  
Database character set: utf8mb4  
Client character set:   utf8mb4  
Conn. character set:    utf8mb4  
TCP port:               3306  
Binary data as:         Hexadecimal  
Uptime:                 5 min 48 sec  
  
Threads: 4 Questions: 14 Slow queries: 0 Opens: 166 Flush tables: 3 Open tables: 85 Queries per second avg: 0.040  
-----
```

## 4. Creating an SSL-Required User

```
CREATE USER 'ssluser'@'localhost' IDENTIFIED BY 'myStrongPassword';  
GRANT ALL PRIVILEGES ON *.* TO 'ssluser'@'localhost' REQUIRE SSL;  
FLUSH PRIVILEGES;
```

This user is forced to use SSL for any connection.



```
scofield@scofieldserver: /etc/mysql/ssl  
Enter password:  
ERROR 1698 (28000): Access denied for user 'root'@'localhost'  
scofield@scofieldserver:/etc/mysql/ssl$ sudo mysql -u root -p --ssl-mode=REQUIRED -h 127.0.0.1  
Enter password:  
ERROR 1698 (28000): Access denied for user 'root'@'localhost'  
scofield@scofieldserver:/etc/mysql/ssl$ sudo mysql  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 11  
Server version: 8.0.42-0ubuntu0.20.04.1 (Ubuntu)  
  
Copyright (c) 2000, 2025, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
mysql> CREATE USER 'ssluser'@'localhost' IDENTIFIED BY 'yourStrongPassword';  
GRANT ALL PRIVILEGES ON *.* TO 'ssluser'@'localhost' REQUIRE SSL;  
FLUSH PRIVILEGES;  
Query OK, 0 rows affected (0.16 sec)  
  
mysql> GRANT ALL PRIVILEGES ON *.* TO 'ssluser'@'localhost' REQUIRE SSL;  
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version  
for the right syntax to use near 'REQUIRE SSL' at line 1  
mysql> FLUSH PRIVILEGES;  
Query OK, 0 rows affected (0.05 sec)  
  
mysql>
```

## 5. Testing the Connection

I tested the secure connection with:

```
mysql -u ssluser -p --ssl-mode=REQUIRED -h 127.0.0.1
```

Initially, I got an error saying "**server doesn't support SSL**", which I fixed by properly generating the certificates and restarting MySQL after adding them to the config file.

**Screenshot: SSL-enabled connection success**

```
scofield@scofieldserver: ~
scofield@scofieldserver:~$ sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
scofield@scofieldserver:~$ sudo systemctl restart mysql
scofield@scofieldserver:~$ sudo apt install tcpdump
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be upgraded:
  tcpdump
1 upgraded, 0 newly installed, 0 to remove and 60 not upgraded.
Need to get 370 kB of archives.
After this operation, 0 B of additional disk space will be used.
Get:1 http://ng.archive.ubuntu.com/ubuntu focal-updates/main amd64 tcpdump amd64 4.9.3-4ubuntu0.3 [370 kB]
Fetched 370 kB in 5s (78.4 kB/s)
(Reading database ... 72885 files and directories currently installed.)
Preparing to unpack .../tcpdump_4.9.3-4ubuntu0.3_amd64.deb ...
Unpacking tcpdump (4.9.3-4ubuntu0.3) over (4.9.3-4ubuntu0.2) ...
Setting up tcpdump (4.9.3-4ubuntu0.3) ...
Installing new version of config file /etc/apparmor.d/usr.sbin.tcpdump ...
Processing triggers for man-db (2.9.1-1) ...
scofield@scofieldserver:~$ sudo tcpdump -i any port 3306 -w mysql_traffic.pcap
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked v1), capture size 262144 bytes
```

Inside the session, I ran:

```
CREATE DATABASE test;
USE test;
CREATE TABLE secure_data (id INT, name VARCHAR(100));
INSERT INTO secure_data VALUES (1, 'Scofield'), (2, 'TestUser');
SELECT * FROM secure_data;
```

---

## 6. Capturing Traffic

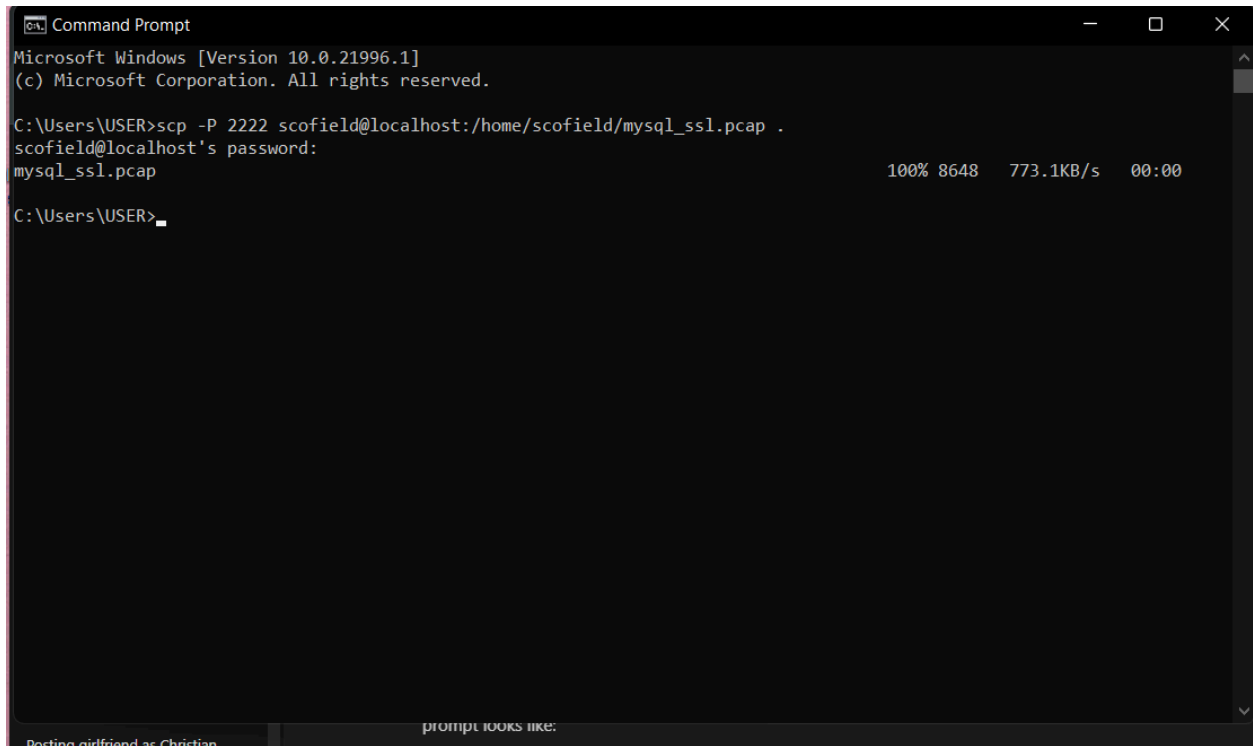
In a new terminal, I ran:

```
sudo tcpdump -i any port 3306 -w mysql_ssl.pcap
```

While the capture was running, I repeated the MySQL commands above using the `ssluser` account.

Then I stopped tcpdump with `Ctrl + C`.

### Screenshot: tcpdump capturing traffic



```
Command Prompt
Microsoft Windows [Version 10.0.21996.1]
(c) Microsoft Corporation. All rights reserved.

C:\Users\USER>scp -P 2222 scofield@localhost:/home/scofield/mysql_ssl.pcap .
scofield@localhost's password:
mysql_ssl.pcap                                     100% 8648   773.1KB/s   00:00

C:\Users\USER>
```

## 7. Copying the .pcap File to Windows

This was tricky. I initially ran the `scp` command from inside the VM and kept getting errors like:

```
ssh: connect to host localhost port 2222: Connection refused
2222: No such file or directory
```

Eventually, I realized I needed to run the command from **Windows CMD**, not inside the VM.

Correct command (run from Windows):

```
scp -P 2222 scofield@localhost:/home/scofield/mysql_ssl.pcap .
```

I had to make sure:

- Port forwarding was enabled in VirtualBox (Host: 2222 → Guest: 22)
- The file path was correct (`/home/scofield/mysql_ssl.pcap`)

After that, the file transferred successfully.

```
Command Prompt
Microsoft Windows [Version 10.0.21996.1]
(c) Microsoft Corporation. All rights reserved.

C:\Users\USER>scp -P 2222 scofield@localhost:/home/scofield/mysql_ssl.pcap .
scofield@localhost's password:
mysql_ssl.pcap                                     100% 8648   773.1KB/s   00:00

C:\Users\USER>
```

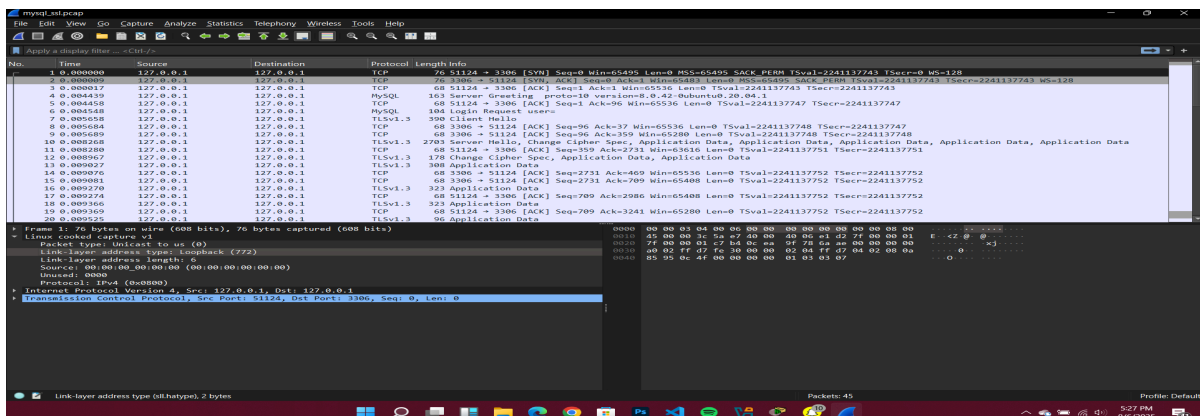
## 8. Analyzing in Wireshark

Opened the file in Wireshark and filtered for:

tcp.port == 3306

I saw a successful TLS handshake (**Client Hello**, **Server Hello**, **Certificate**, etc.) and all subsequent traffic appeared as “**Encrypted Application Data**”, confirming that the credentials and queries were protected.

### Screenshot: TLS handshake and encrypted packets in Wireshark



## Conclusion (for Q5)

- The TLS setup was successful.
- Passwords and data were not visible in plaintext.
- TLS handshake and encryption worked as expected.
- Key mistake: trying to use `scp` from inside the VM instead of from the Windows host.