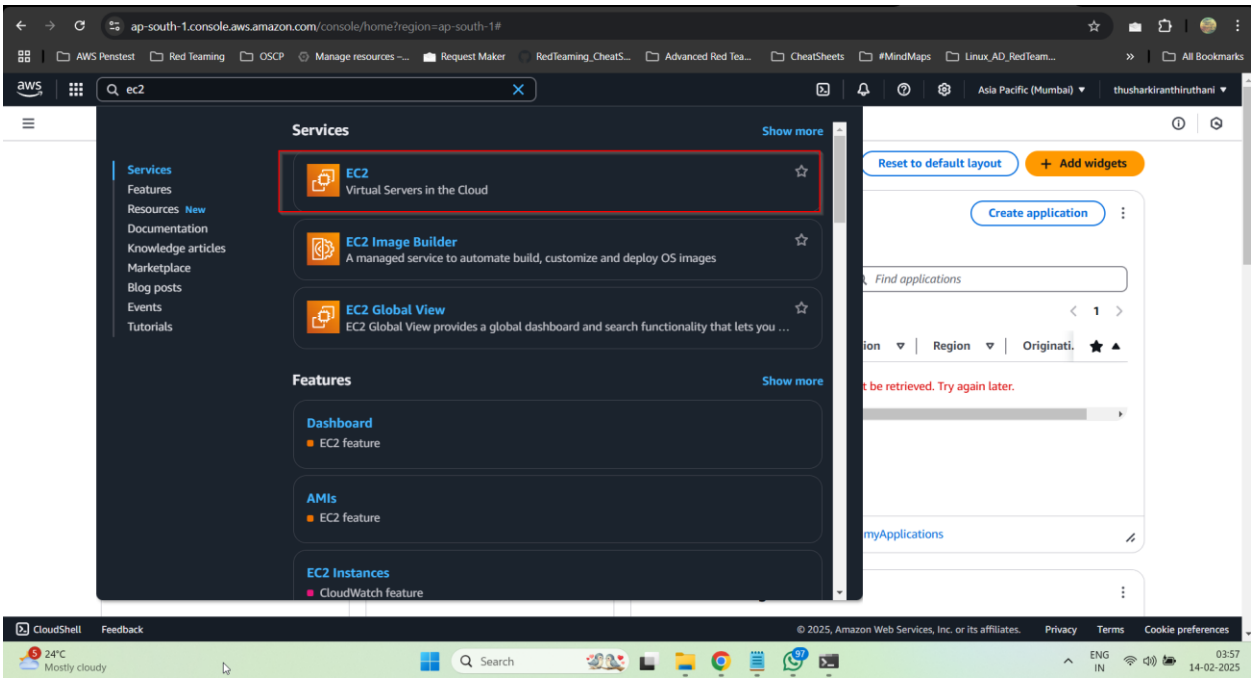
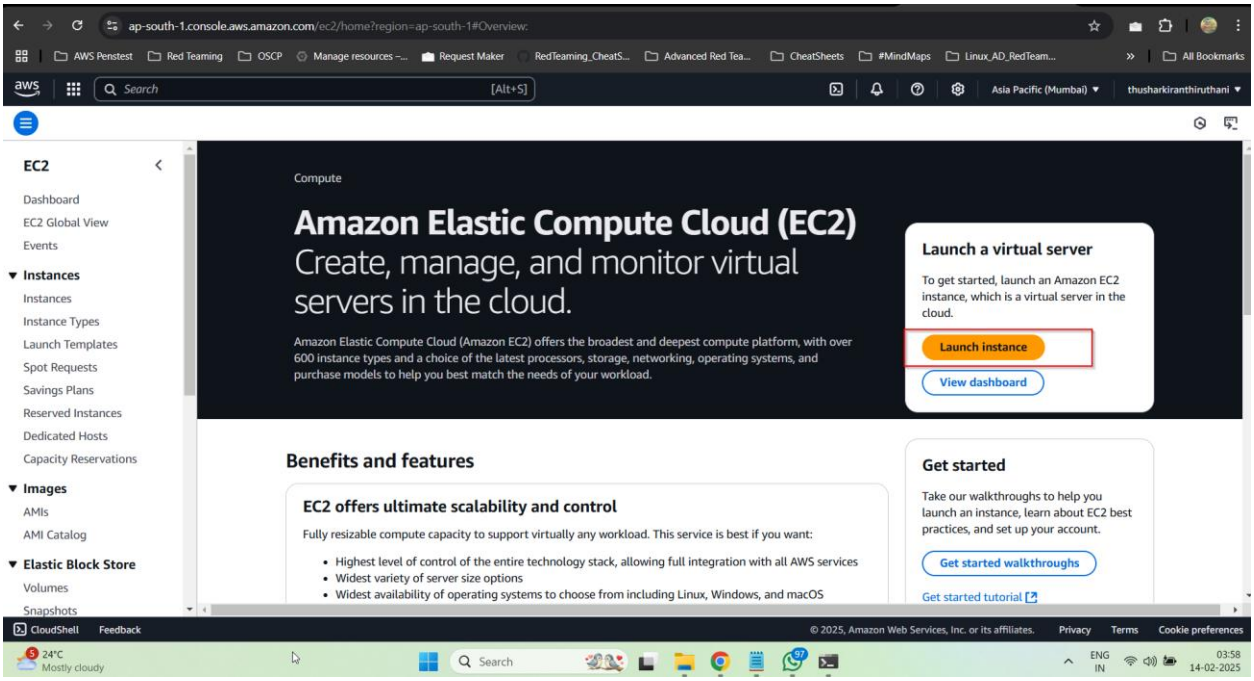
Part 1: Setup & Vulnerable Form

1.1 Spin Up an EC2 Instance

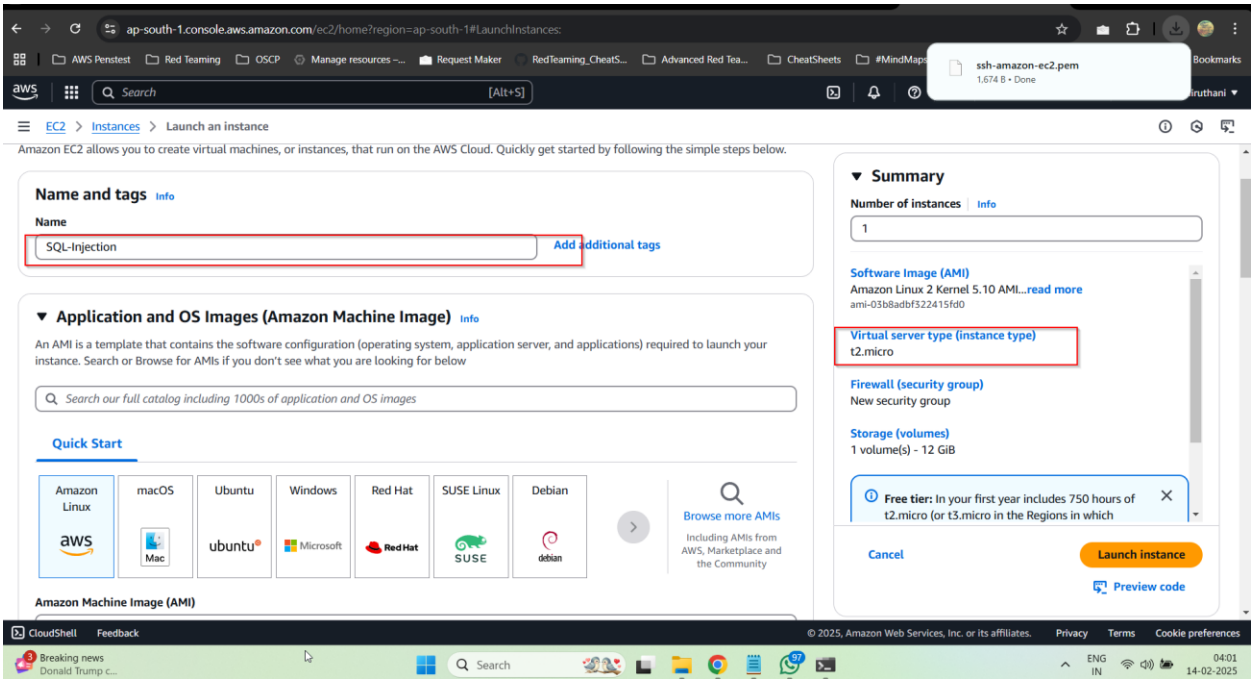
1.Log in to AWS and go to the EC2 dashboard.



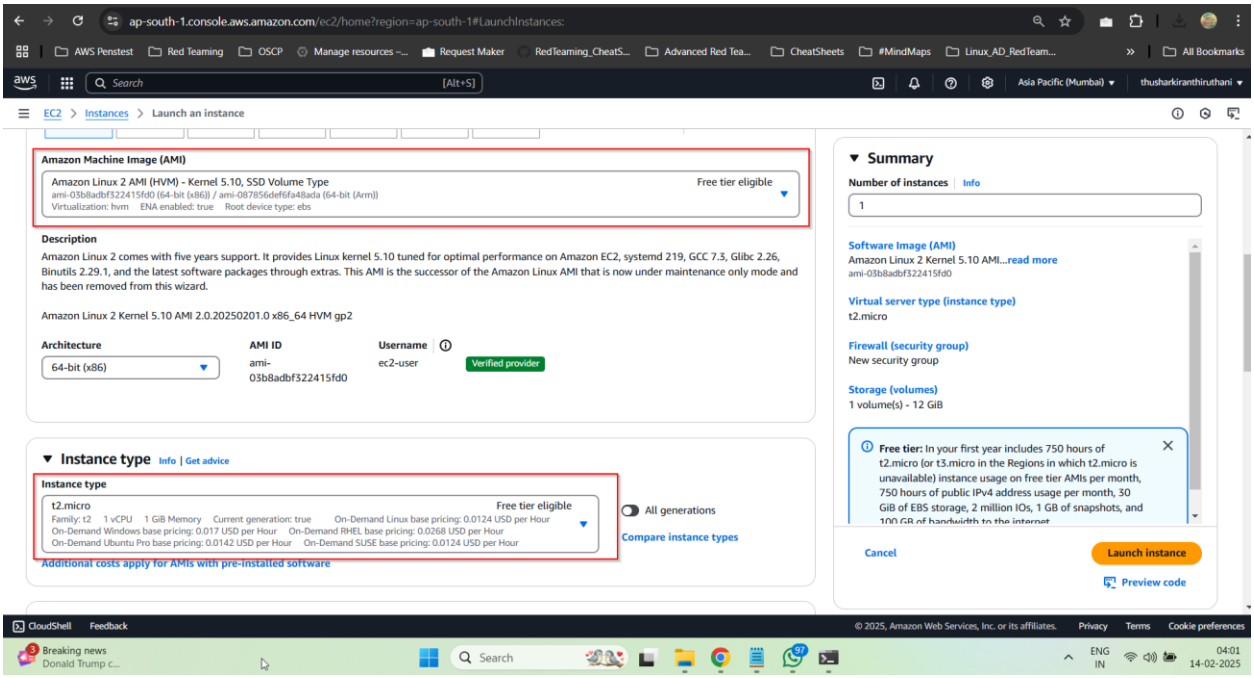


2.Launch Instance:

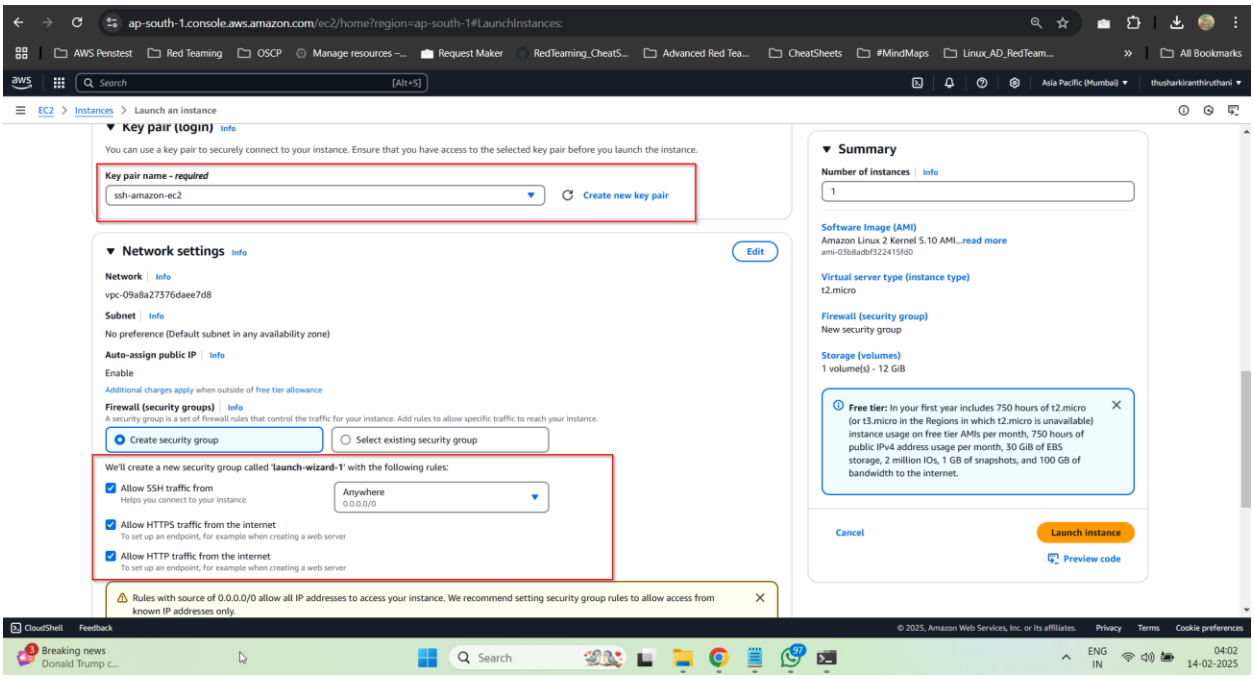
a) AMI: Choose Amazon Linux 2 .



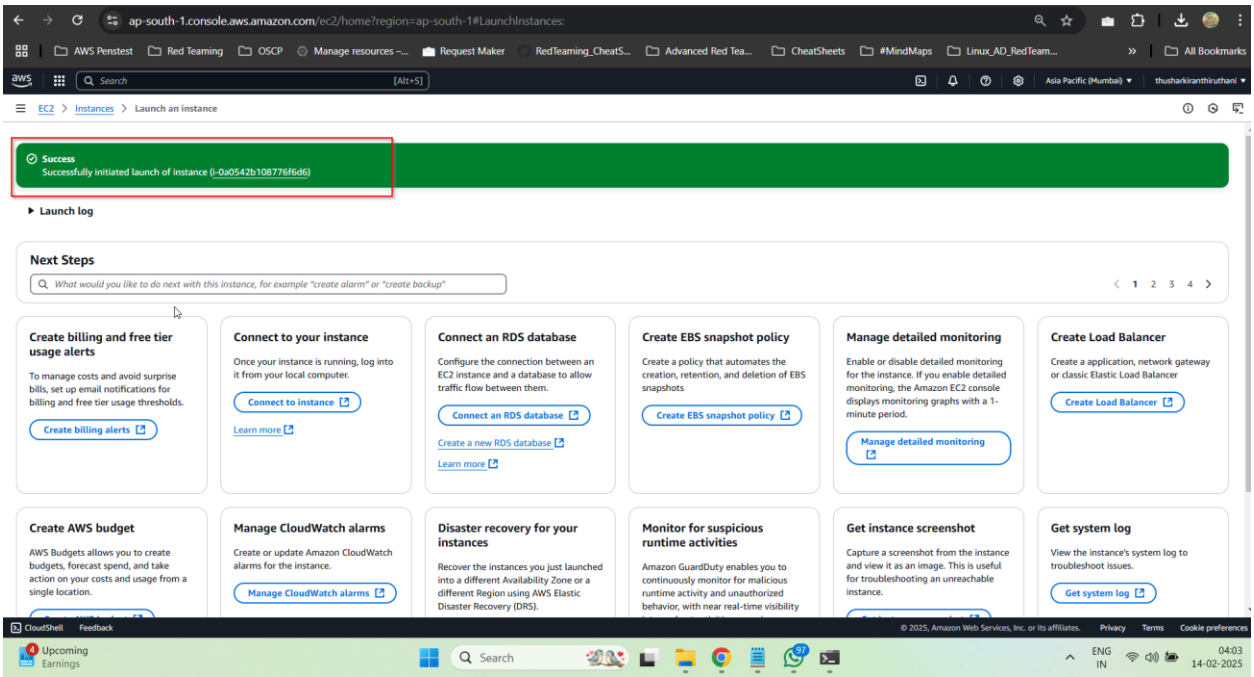
b)Instance Type: t2.micro (free tier eligible).

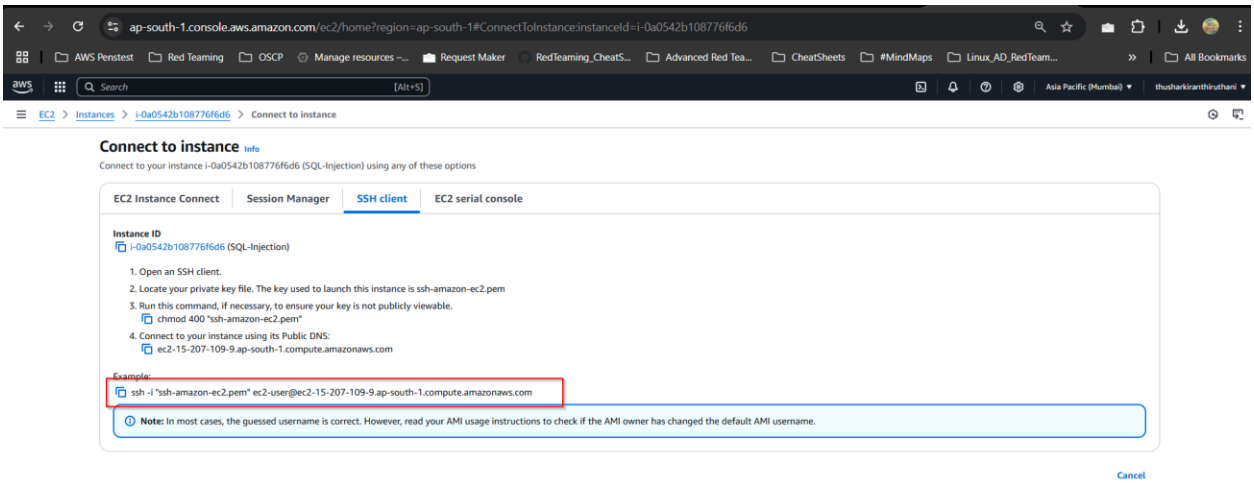


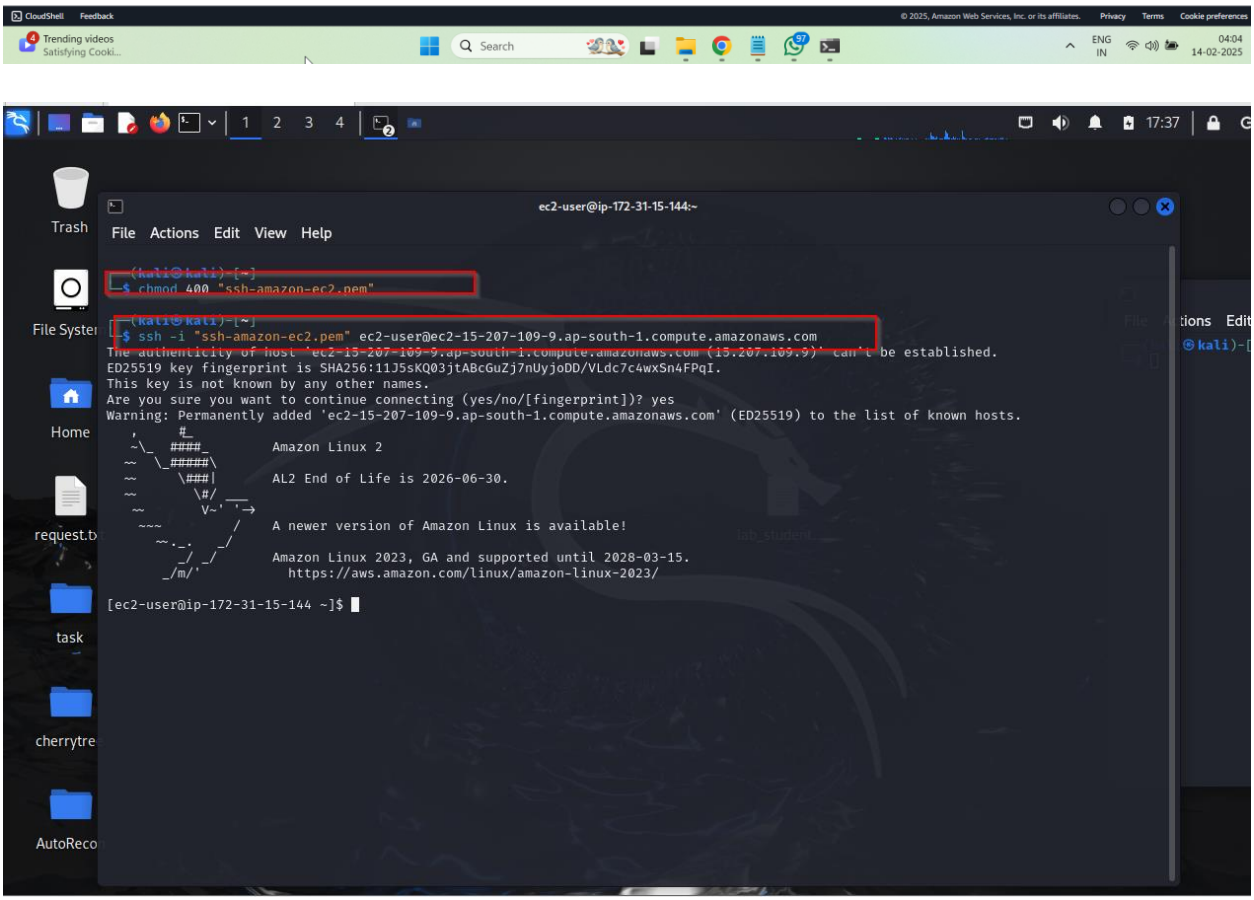
c)Security Group: Allow inbound HTTP (port 80) and SSH (port 22).



3.Launch and note your instance's Public IP or Public DNS.



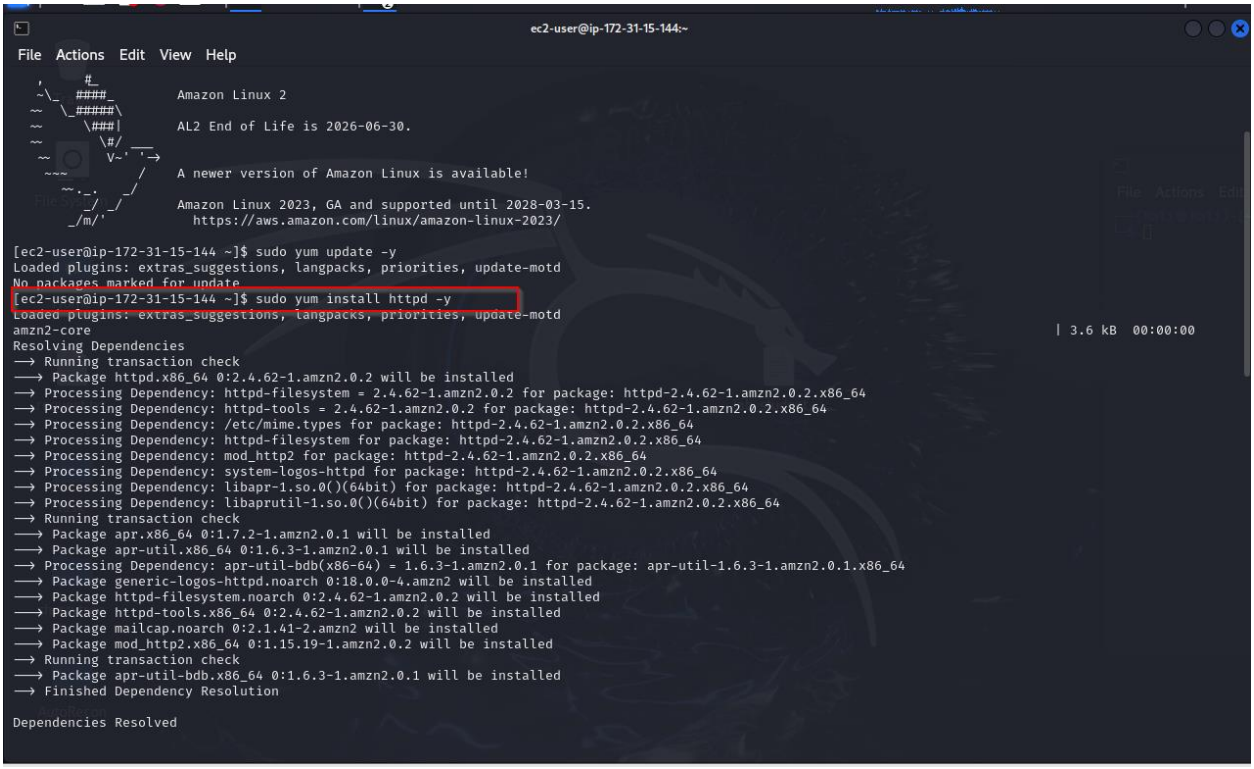


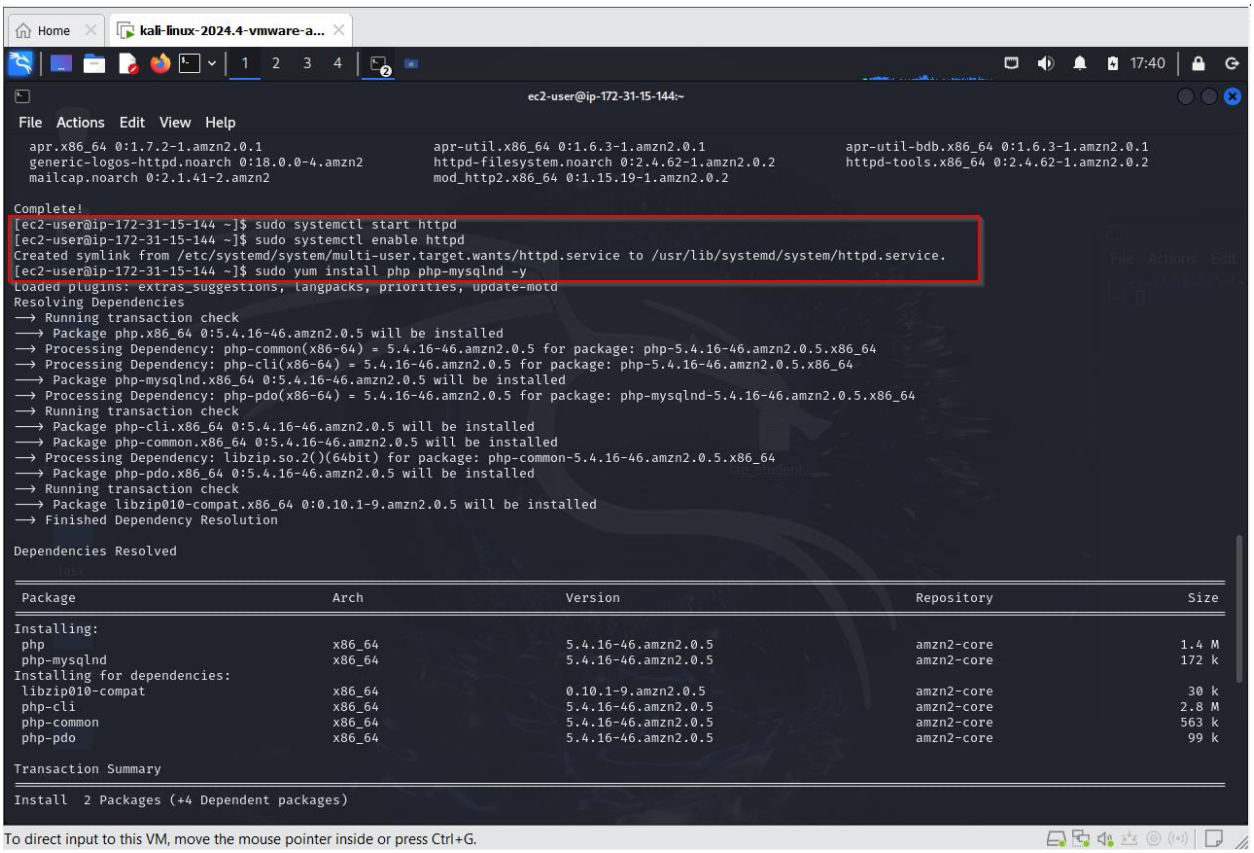


1.2 Install Web Server (Apache or Nginx)

Apache on Amazon Linux 2

```
# Update system packages
sudo yum update -y

# Install Apache
sudo yum install httpd -y

# Start and enable Apache
sudo systemctl start httpd
sudo systemctl enable httpd
```

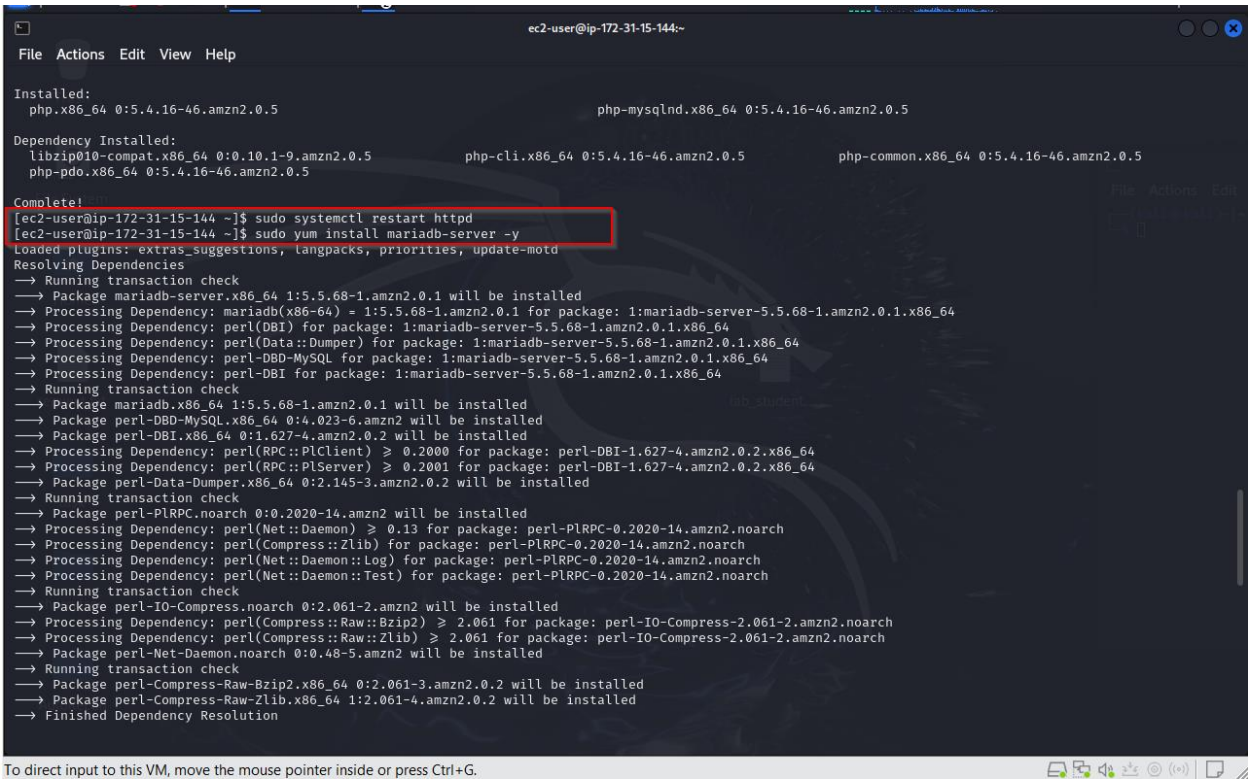




```
ec2-user@ip-172-31-15-144:~
File  Actions  Edit  View  Help
  apr.x86_64 0:1.7.2-1.amzn2.0.1                    apr-util.x86_64 0:1.6.3-1.amzn2.0.1                apr-util-bdb.x86_64 0:1.6.3-1.amzn2.0.1
  generic-logos-httpd.noarch 0:18.0.0-4.amzn2       httpd-filesystem.noarch 0:2.4.62-1.amzn2.0.2       httpd-tools.x86_64 0:2.4.62-1.amzn2.0.2
  mailcap.noarch 0:2.1.41-2.amzn2                   mod_http2.x86_64 0:1.15.19-1.amzn2.0.2

Complete!
[ec2-user@ip-172-31-15-144 ~]$ sudo systemctl start httpd
[ec2-user@ip-172-31-15-144 ~]$ sudo systemctl enable httpd
Created symlink from /etc/systemd/system/multi-user.target.wants/httpd.service to /usr/lib/systemd/system/httpd.service.
[ec2-user@ip-172-31-15-144 ~]$ sudo yum install php php-mysqlnd -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
---> Package php.x86_64 0:5.4.16-46.amzn2.0.5 will be installed
--> Processing Dependency: php-common(x86-64) = 5.4.16-46.amzn2.0.5 for package: php-5.4.16-46.amzn2.0.5.x86_64
--> Processing Dependency: php-cli(x86-64) = 5.4.16-46.amzn2.0.5 for package: php-5.4.16-46.amzn2.0.5.x86_64
---> Package php-mysqlnd.x86_64 0:5.4.16-46.amzn2.0.5 will be installed
--> Processing Dependency: php-pdo(x86-64) = 5.4.16-46.amzn2.0.5 for package: php-mysqlnd-5.4.16-46.amzn2.0.5.x86_64
--> Running transaction check
---> Package php-cli.x86_64 0:5.4.16-46.amzn2.0.5 will be installed
---> Package php-common.x86_64 0:5.4.16-46.amzn2.0.5 will be installed
--> Processing Dependency: libzip.so.2()(64bit) for package: php-common-5.4.16-46.amzn2.0.5.x86_64
---> Package php-pdo.x86_64 0:5.4.16-46.amzn2.0.5 will be installed
--> Running transaction check
---> Package libzip010-compat.x86_64 0:0.10.1-9.amzn2.0.5 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

 Package                    Arch        Version                    Repository        Size
Installing:
 php                        x86_64      5.4.16-46.amzn2.0.5        amzn2-core        1.4 M
 php-mysqlnd                x86_64      5.4.16-46.amzn2.0.5        amzn2-core        172 k
Installing for dependencies:
 libzip010-compat           x86_64      0.10.1-9.amzn2.0.5         amzn2-core         30 k
 php-cli                    x86_64      5.4.16-46.amzn2.0.5        amzn2-core        2.8 M
 php-common                 x86_64      5.4.16-46.amzn2.0.5        amzn2-core        563 k
 php-pdo                    x86_64      5.4.16-46.amzn2.0.5        amzn2-core         99 k

Transaction Summary
Install  2 Packages (+4 Dependent packages)
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
```

1.3 Install PHP & Database (MariaDB/MySQL) 1.PHP (for Amazon Linux 2 + Apache):

```
sudo yum install php php-mysqlnd -y
 sudo systemctl restart httpd
```



2.MariaDB (or MySQL):

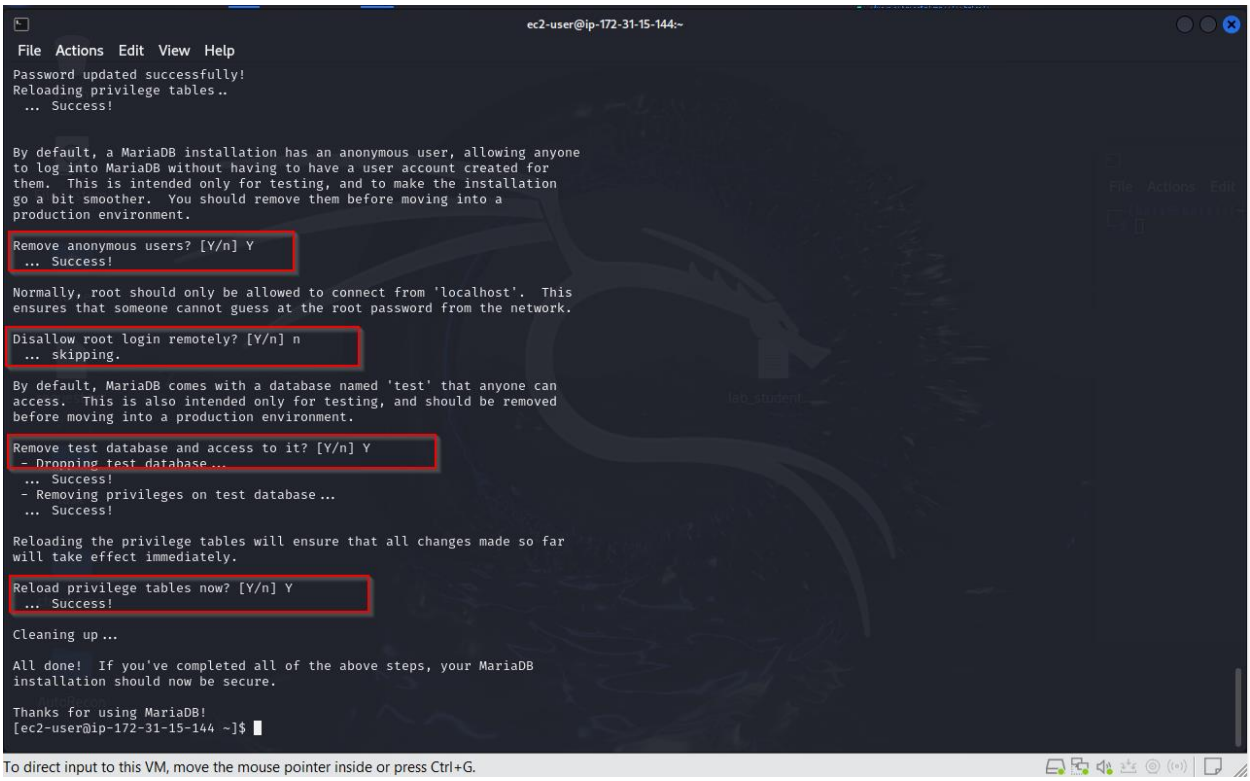
```
sudo yum install mariadb-server -y
sudo systemctl start mariadb
sudo systemctl enable MariaDB
```

```
ec2-user@ip-172-31-15-144:~
File  Actions  Edit  View  Help
  Installing : 1:mariadb-5.5.68-1.amzn2.0.1.x86_64                                   2/10
  Installing : perl-Compress-Raw-Bzip2-2.061-3.amzn2.0.2.x86_64                      3/10
  Installing : perl-Net-Daemon-0.48-5.amzn2.noarch                                   4/10
  Installing : 1:perl-Compress-Raw-Zlib-2.061-4.amzn2.0.2.x86_64                     5/10
  Installing : perl-IO-Compress-2.061-2.amzn2.noarch                                 6/10
  Installing : perl-PlRPC-0.2020-14.amzn2.noarch                                     7/10
  Installing : perl-DBI-1.627-4.amzn2.0.2.x86_64                                     8/10
  Installing : perl-DBD-MySQL-4.023-6.amzn2.x86_64                                   9/10
  Installing : 1:mariadb-server-5.5.68-1.amzn2.0.1.x86_64                           10/10
  Verifying  : 1:perl-Compress-Raw-Zlib-2.061-4.amzn2.0.2.x86_64                     1/10
  Verifying  : perl-IO-Compress-2.061-2.amzn2.noarch                                 2/10
  Verifying  : perl-Net-Daemon-0.48-5.amzn2.noarch                                   3/10
  Verifying  : perl-Data-Dumper-2.145-3.amzn2.0.2.x86_64                             4/10
  Verifying  : perl-DBD-MySQL-4.023-6.amzn2.x86_64                                   5/10
  Verifying  : perl-Compress-Raw-Bzip2-2.061-3.amzn2.0.2.x86_64                      6/10
  Verifying  : 1:mariadb-5.5.68-1.amzn2.0.1.x86_64                                   7/10
  Verifying  : perl-DBI-1.627-4.amzn2.0.2.x86_64                                     8/10
  Verifying  : perl-PlRPC-0.2020-14.amzn2.noarch                                     9/10
  Verifying  : 1:mariadb-server-5.5.68-1.amzn2.0.1.x86_64                           10/10

Installed:
  mariadb-server.x86_64 1:5.5.68-1.amzn2.0.1

Dependency Installed:
  mariadb.x86_64 1:5.5.68-1.amzn2.0.1         perl-Compress-Raw-Bzip2.x86_64 0:2.061-3.amzn2.0.2     perl-Compress-Raw-Zlib.x86_64 1:2.061-4.amzn2.0.2
  perl-DBD-MySQL.x86_64 0:4.023-6.amzn2       perl-DBI.x86_64 0:1.627-4.amzn2.0.2                    perl-Data-Dumper.x86_64 0:2.145-3.amzn2.0.2
  perl-IO-Compress.noarch 0:2.061-2.amzn2     perl-Net-Daemon.noarch 0:0.48-5.amzn2                  perl-PlRPC.noarch 0:0.2020-14.amzn2

Complete!
[ec2-user@ip-172-31-15-144 ~]$ sudo systemctl start mariadb
[ec2-user@ip-172-31-15-144 ~]$ sudo systemctl enable mariadb
Created symlink from /etc/systemd/system/multi-user.target.wants/mariadb.service to /usr/lib/systemd/system/mariadb.service.
[ec2-user@ip-172-31-15-144 ~]$ sudo mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user.  If you've just installed MariaDB, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.

Enter current password for root (enter for none):
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
```

3.Secure your database:

sudo mysql_secure_installation

(Set root password, remove test databases, etc.)



1.4 Create a Database and Users Table

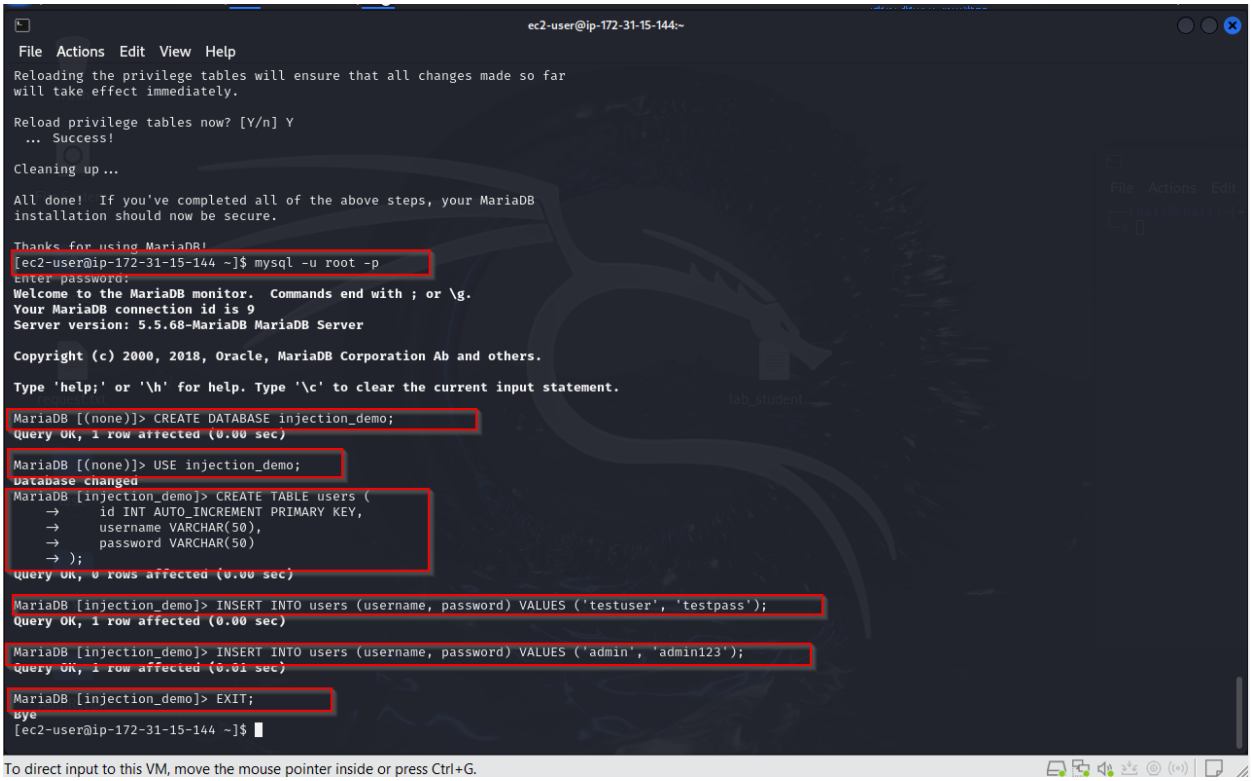
1.Log in to MariaDB/MySQL:

```
mysql -u root -p
```

2.Create a database and table:

```
CREATE DATABASE injection_demo;
USE injection_demo;

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50),
    password VARCHAR(50)
);

INSERT INTO users (username, password) VALUES ('testuser',
```

```
'testpass');
    INSERT INTO users (username, password) VALUES ('admin',
'admin123');
    EXIT;
```

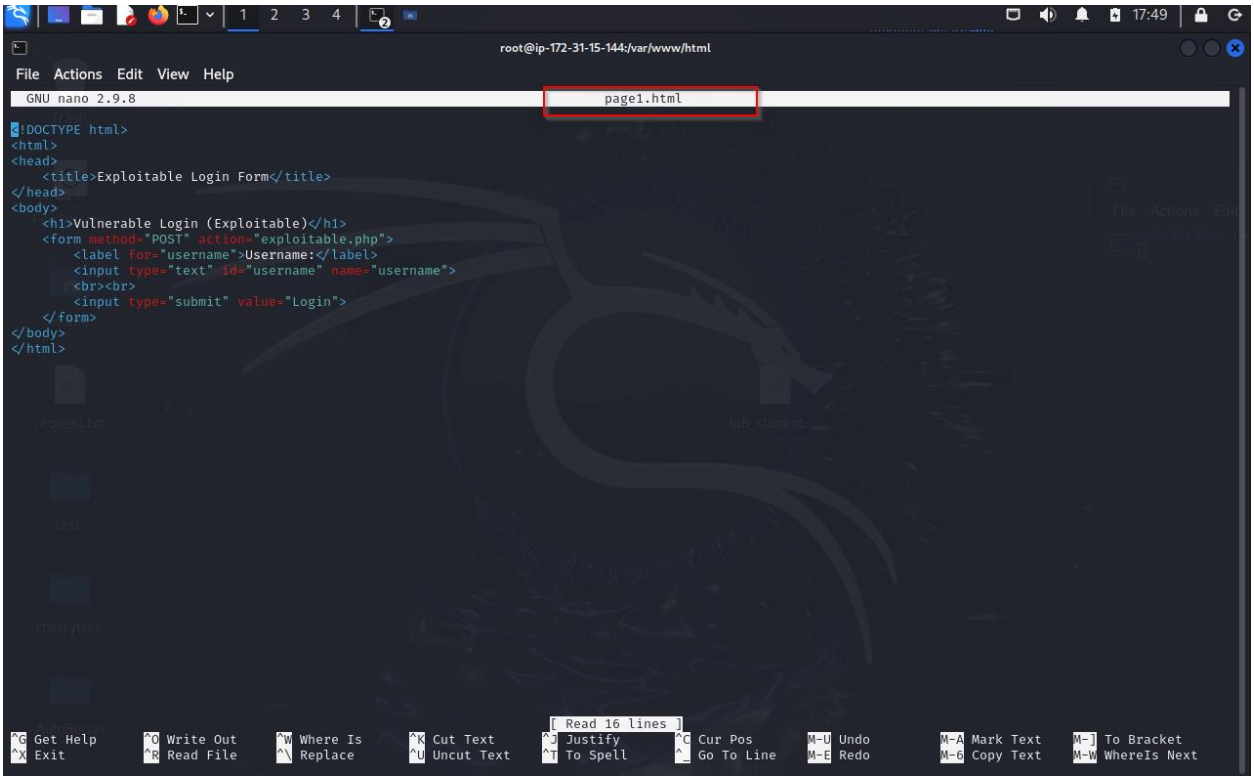


1.5 Create the Vulnerable Form (page1.html + exploitable.php)

1.page1.html (Exploitable Form)

```
<!DOCTYPE html>

<html>

<head>

    <title>Exploitable Login Form</title>

</head>

<body>

    <h1>Vulnerable Login (Exploitable)</h1>
```

```
    <form method="POST" action="exploitable.php">
        <label for="username">Username:</label>
        <input type="text" id="username" name="username">
        <br><br>
        <input type="submit" value="Login">
    </form>
</body>
</html>
```



2.exploitable.php (Intentionally vulnerable)

```php
<?php
// exploitable.php

// Database credentials
$host = "localhost";
$dbname = "injection_demo";
$user = "root";
$pass = "YOUR_DB_ROOT_PASSWORD"; // Replace with actual password

// Connect to database
$conn = new mysqli($host, $user, $pass, $dbname);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Capture user input
$username = $_POST['username'];

// Intentionally vulnerable SQL query
$sql = "SELECT * FROM users WHERE username = '$username'";

$result = $conn->query($sql);

if ($result && $result->num_rows > 0) {
    echo "Login successful (but insecure)!";
} else {
    echo "Login failed!";
}

$conn->close();
?>
```

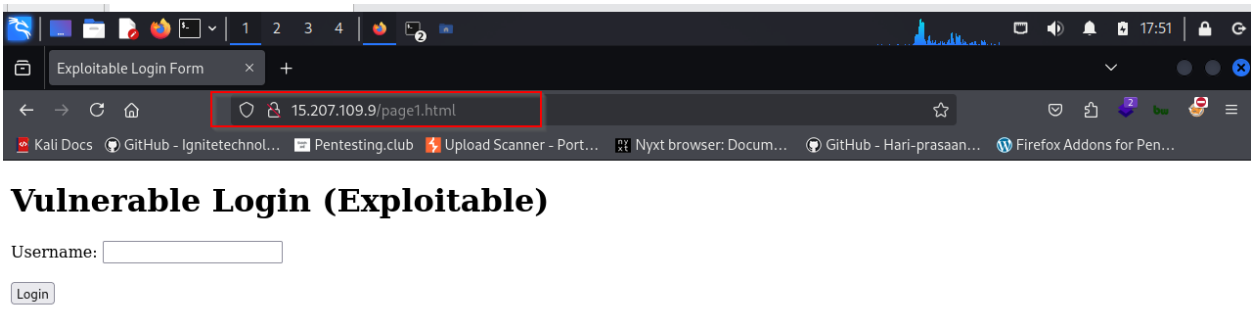
```
root@ip-172-31-15-144:/var/www/html
File  Actions  Edit  View  Help
  GNU nano 2.9.8                    exploitable.php                    Modifie

<?php
// exploitable.php

// Database credentials
$host = "localhost";
$dbname = "injection_demo";
$user = "root";
$pass = "toor"; // Replace with actual password

// Connect to database
$conn = new mysqli($host, $user, $pass, $dbname);
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Capture user input
$username = $_POST['username'];

// Intentionally vulnerable SQL query
$sql = "SELECT * FROM users WHERE username = '$username'";

$result = $conn->query($sql);

if ($result && $result->num_rows > 0) {
    echo "Login successful (but insecure)!";
} else {
    echo "Login failed!";
}

$conn->close();
?>
```

```
root@ip-172-31-15-144:/var/www/html
File  Actions  Edit  View  Help

All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
[ec2-user@ip-172-31-15-144 ~]$ mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 5.5.68-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE injection_demo;
Query OK, 1 row affected (0.00 sec)

MariaDB [(none)]> USE injection_demo;
Database changed
MariaDB [injection_demo]> CREATE TABLE users (
    ->     id INT AUTO_INCREMENT PRIMARY KEY,
    ->     username VARCHAR(50),
    ->     password VARCHAR(50)
    -> );
Query OK, 0 rows affected (0.00 sec)

MariaDB [injection_demo]> INSERT INTO users (username, password) VALUES ('testuser', 'testpass');
Query OK, 1 row affected (0.00 sec)

MariaDB [injection_demo]> INSERT INTO users (username, password) VALUES ('admin', 'admin123');
Query OK, 1 row affected (0.01 sec)

MariaDB [injection_demo]> EXIT;
Bye
[ec2-user@ip-172-31-15-144 ~]$ nano
[ec2-user@ip-172-31-15-144 ~]$ sudo su
[root@ip-172-31-15-144 ec2-user]# cd /var/www/html
[root@ip-172-31-15-144 html]# nano page1.html
[root@ip-172-31-15-144 html]#
[root@ip-172-31-15-144 html]# nano exploitable.php
[root@ip-172-31-15-144 html]# nano page1.html
[root@ip-172-31-15-144 html]#
```

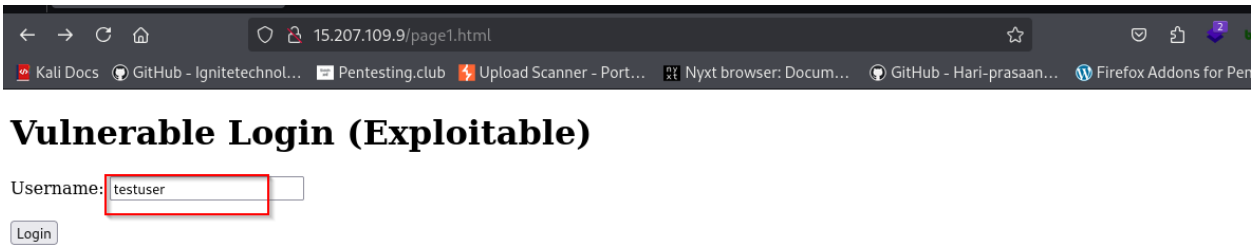
Location: Place both files in /var/www/html/ (if Apache).
URL: http://15.207.109.9/page1.html

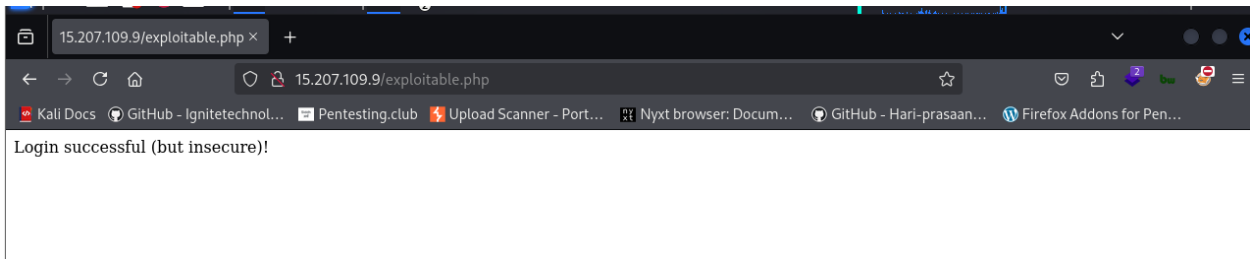
1.6 Test SQL Injection Exploit

1.Visit: http:///page1.html.

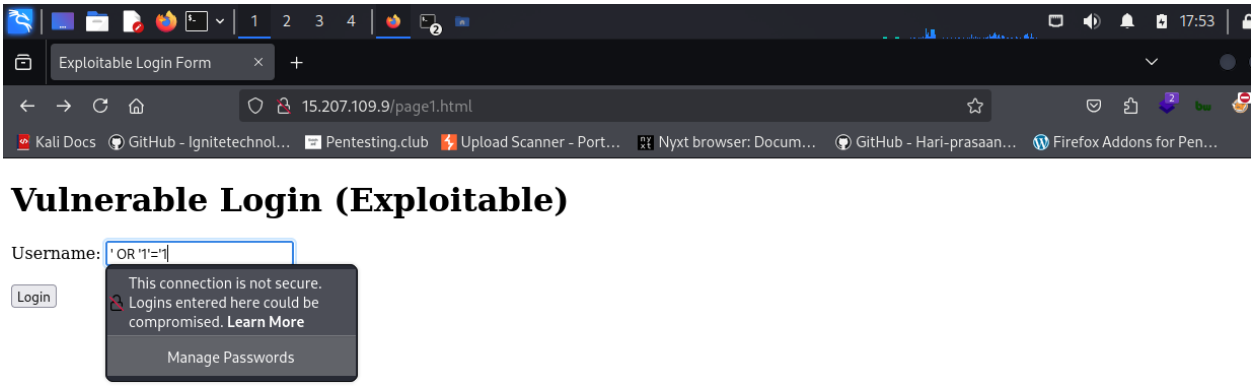


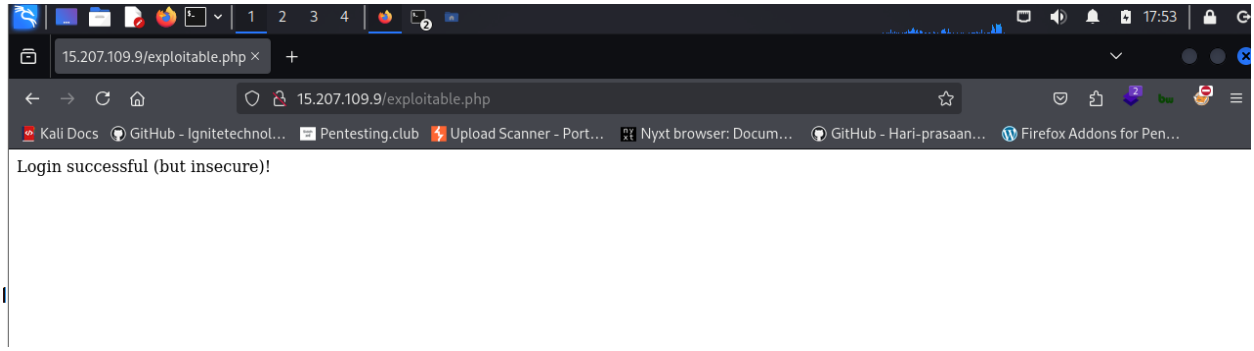
2.Normal Test: Enter testuser to confirm “Login successful.”





3.Injection Test: Enter ' OR '1'='1 in the username field






```
The SQL query becomes:
    SELECT * FROM users WHERE username = '' OR '1'='1'
```

This condition is always true, so it returns “Login successful” even without a valid username.