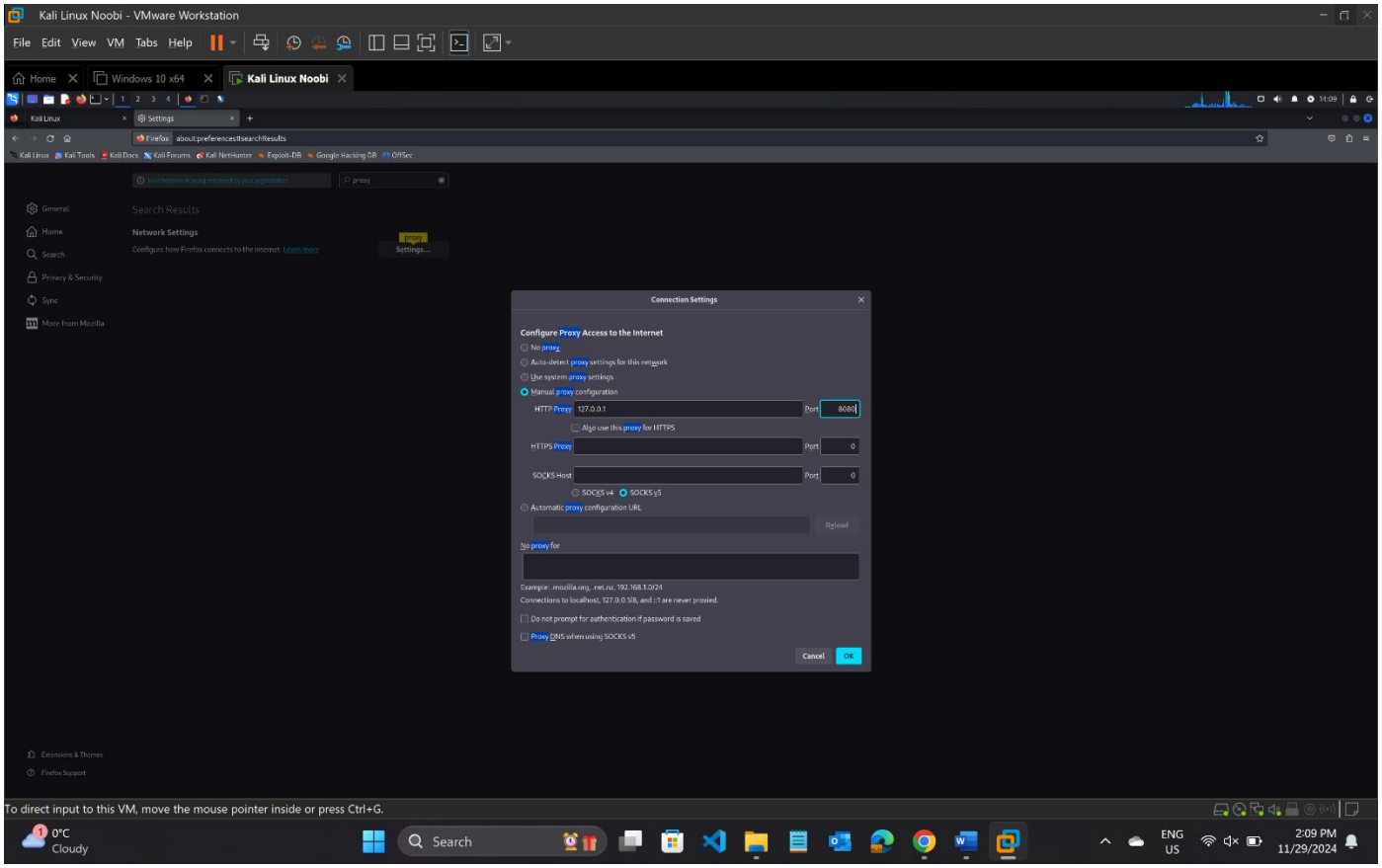
Lab 3: Exploiting Application Vulnerabilities using ZAP, XSS and URL manipulation

Activity 1: Using the ZAP Proxy

In this exercise, you will install the ZAP interception proxy on your system and use it to intercept and modify a request before it is sent to a website.

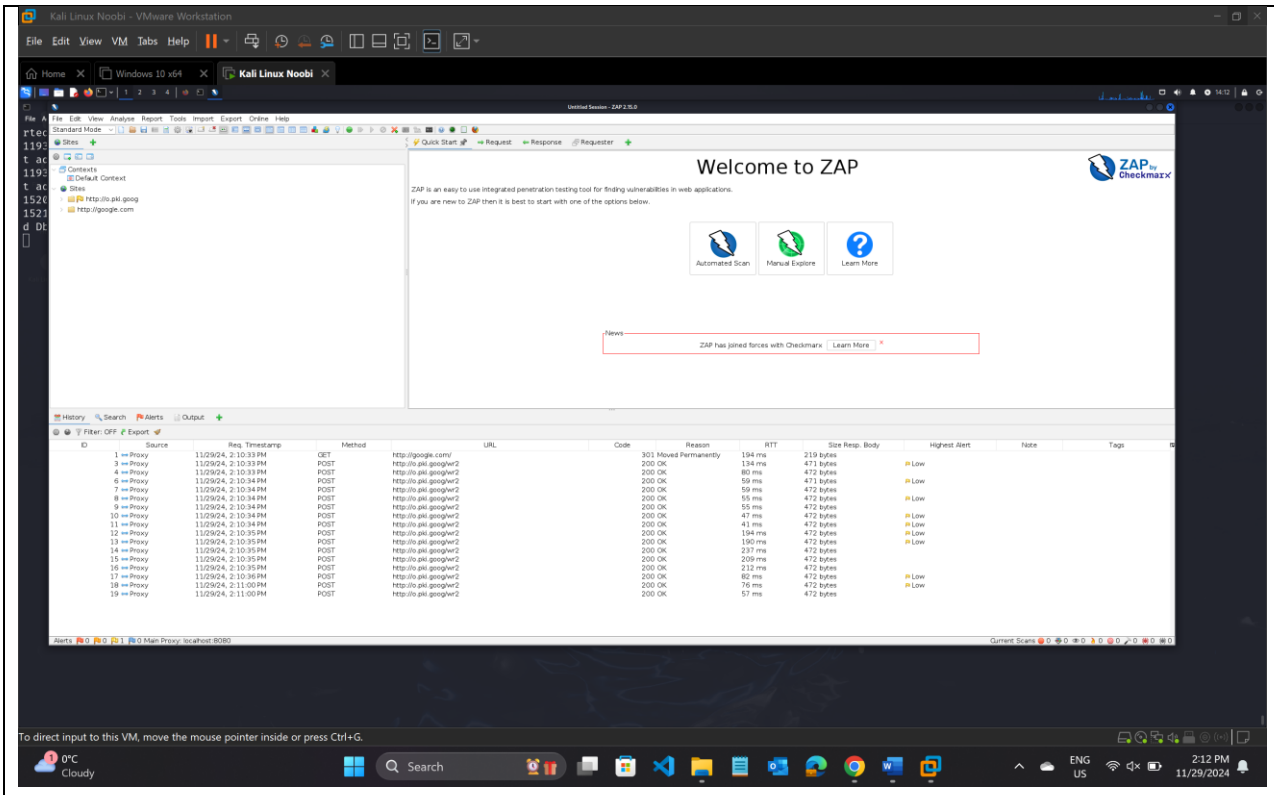
1. Visit the OWASP ZAP project homepage at https://owasp.org/www-project-developer-guide/draft/verification/tools/zed_attack_proxy/
2. Download and install the version of ZAP appropriate for your operating system.
3. Review the OWASP ZAP Getting Started Guide at:
   https://www.zaproxy.org/getting-started/
4. You may go through the following PDF document about ZAP.

20200120-OWASPD
orset-ZAP-DanielW.

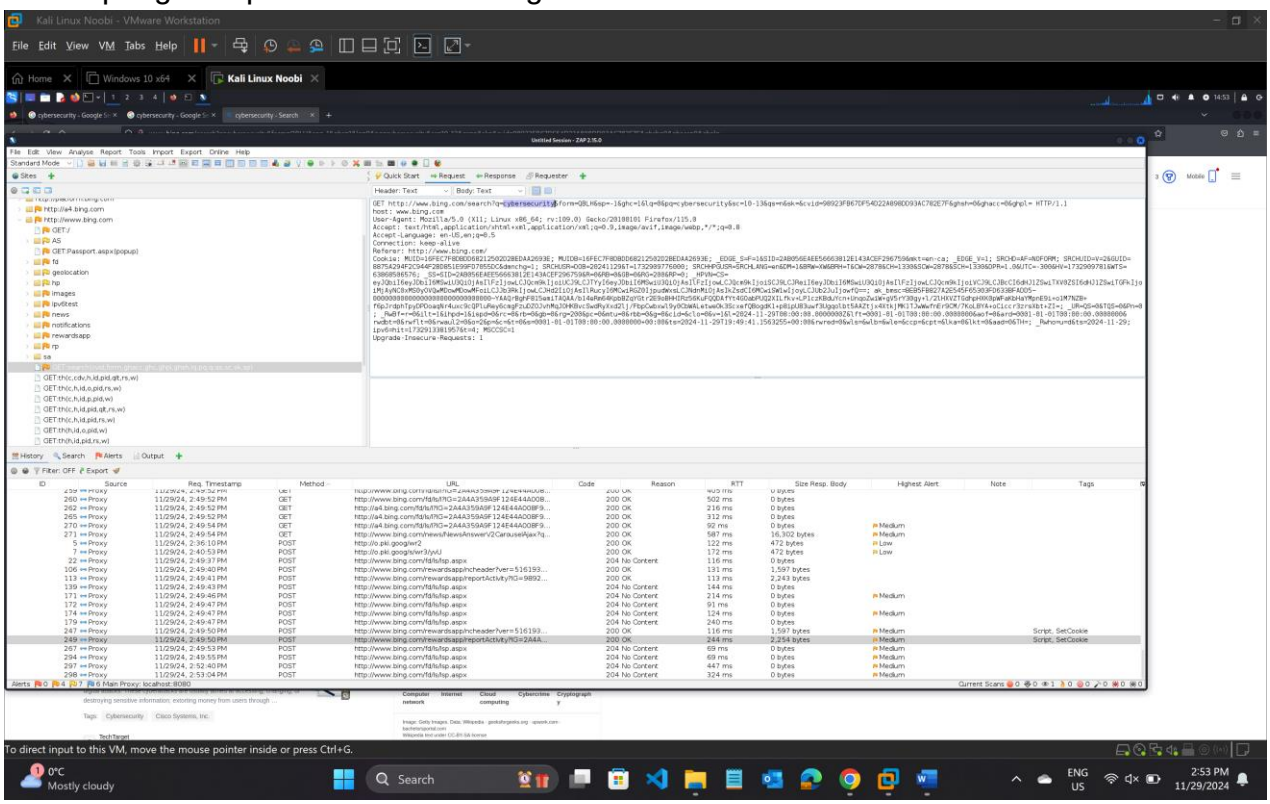


Changing the proxy settings in browser.

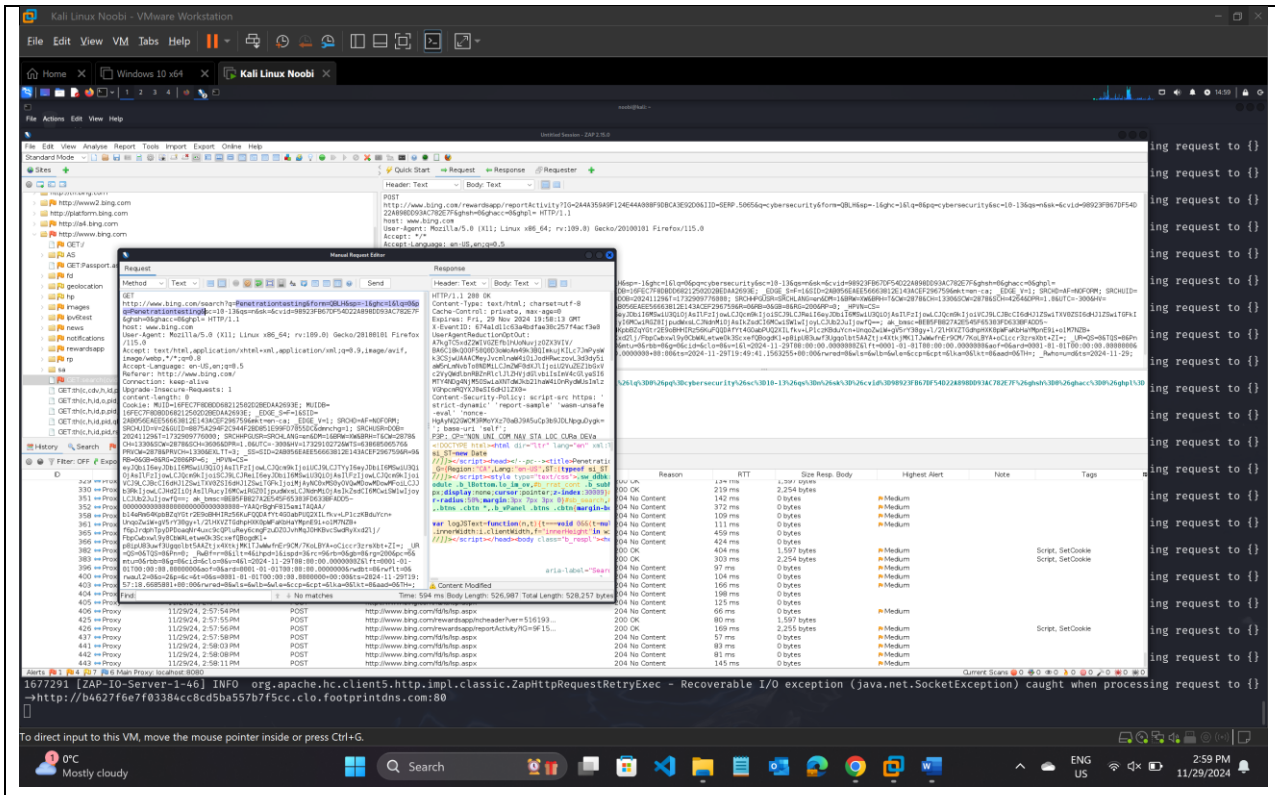
5. Use ZAP to intercept a request sent from your browser to a search engine.



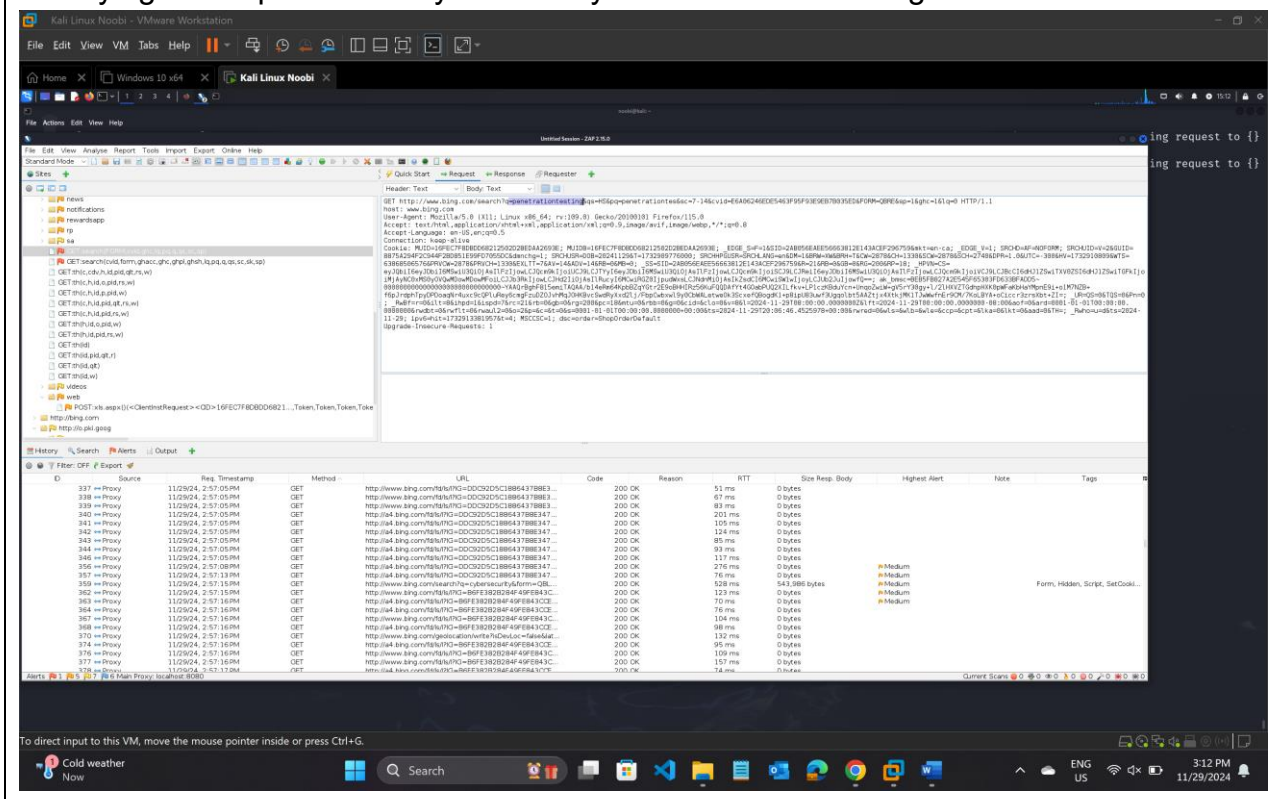
Intercepting a request to a search engine.



Using ZAP, modify the request to change the search term sent to the remote site. **Take the screen shot.**

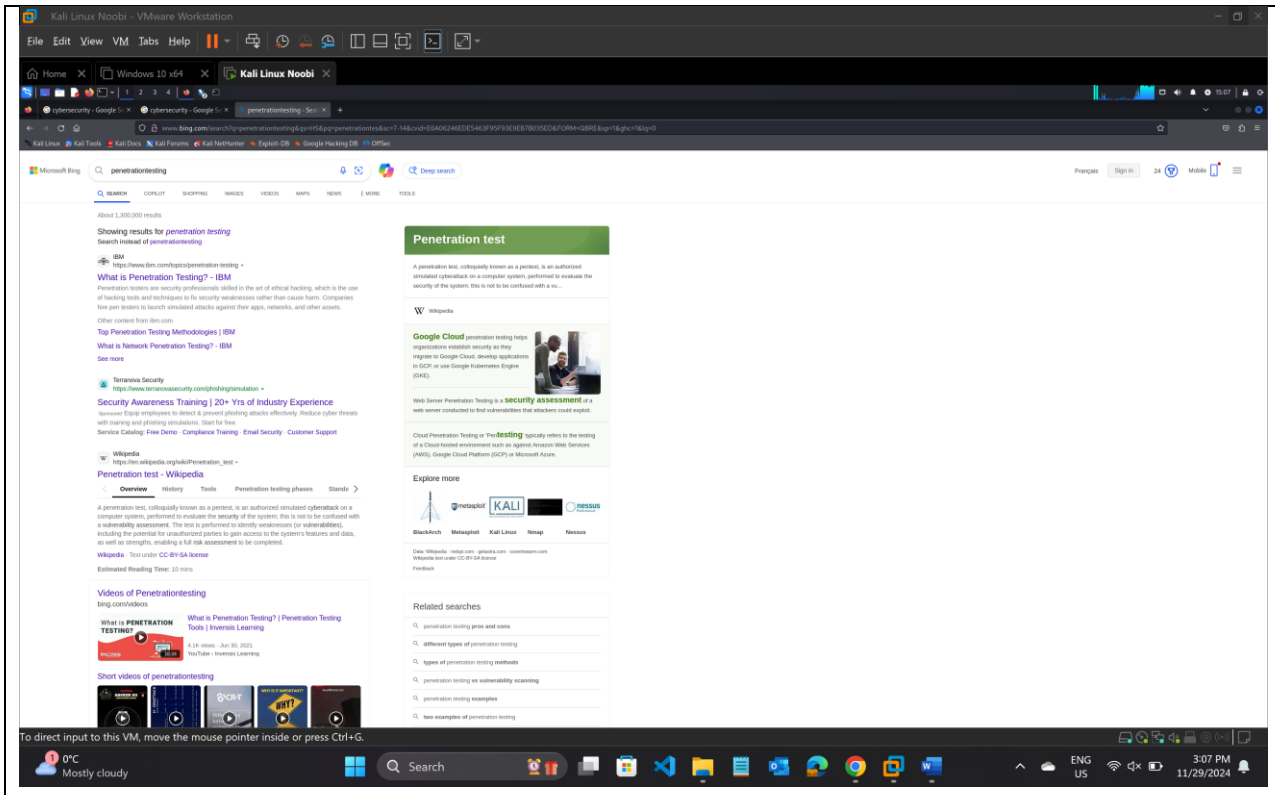


Modifying the request from “cybersecurity’ to “Penetrationtesting” and send the modified request



Changed search term after sending in GET request

6. View the results.



After sending the modified request.

Q1: Did your browser display the results for the term that you typed into the browser, or did it display the results for the search term that you changed using ZAP?

The browser displayed the results for the search term that was changed in ZAP, as the modified request is what the search engine processes. The modified result was shown after the browser page was refreshed. The modified page was able to be viewed from the ZAP by right clicking on the modified request and copying the URL.

Summary:

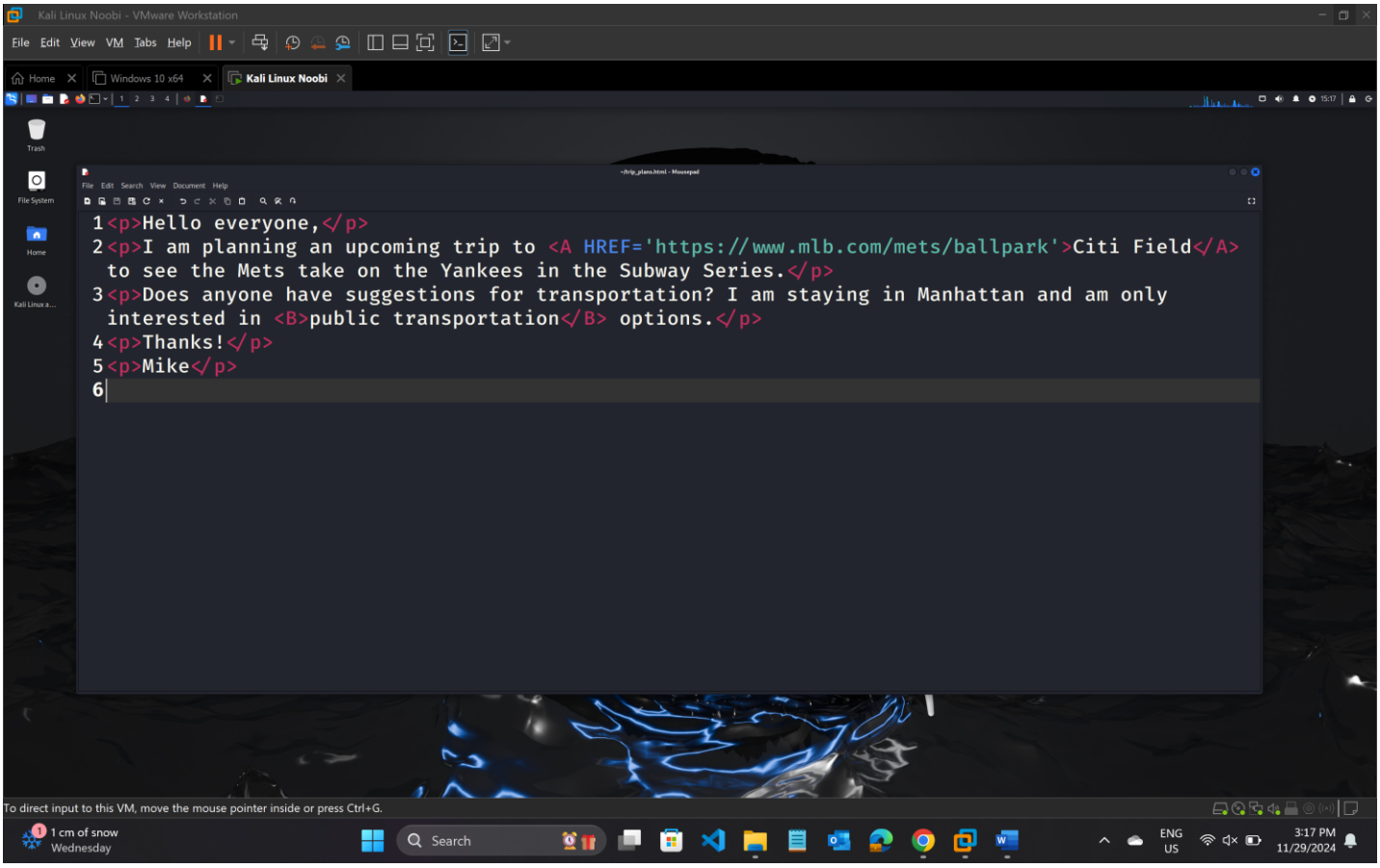
In this lab, I installed and configured OWASP ZAP on Kali Linux to intercept and modify HTTP requests sent from a browser to a website. I set up ZAP as a proxy for Mozilla Firefox and intercepted a GET request sent to Bing with a search query for "Cybersecurity". I modified the search term in the request to "Penetrationtesting" and forwarded the modified request. As a result, Bing displayed search results for the new term, showing how ZAP can be used to manipulate web traffic.

Activity 2: Creating a Cross-Site Scripting Vulnerability

In this activity, you will create a cross-site scripting vulnerability using an HTML page saved on your local computer.

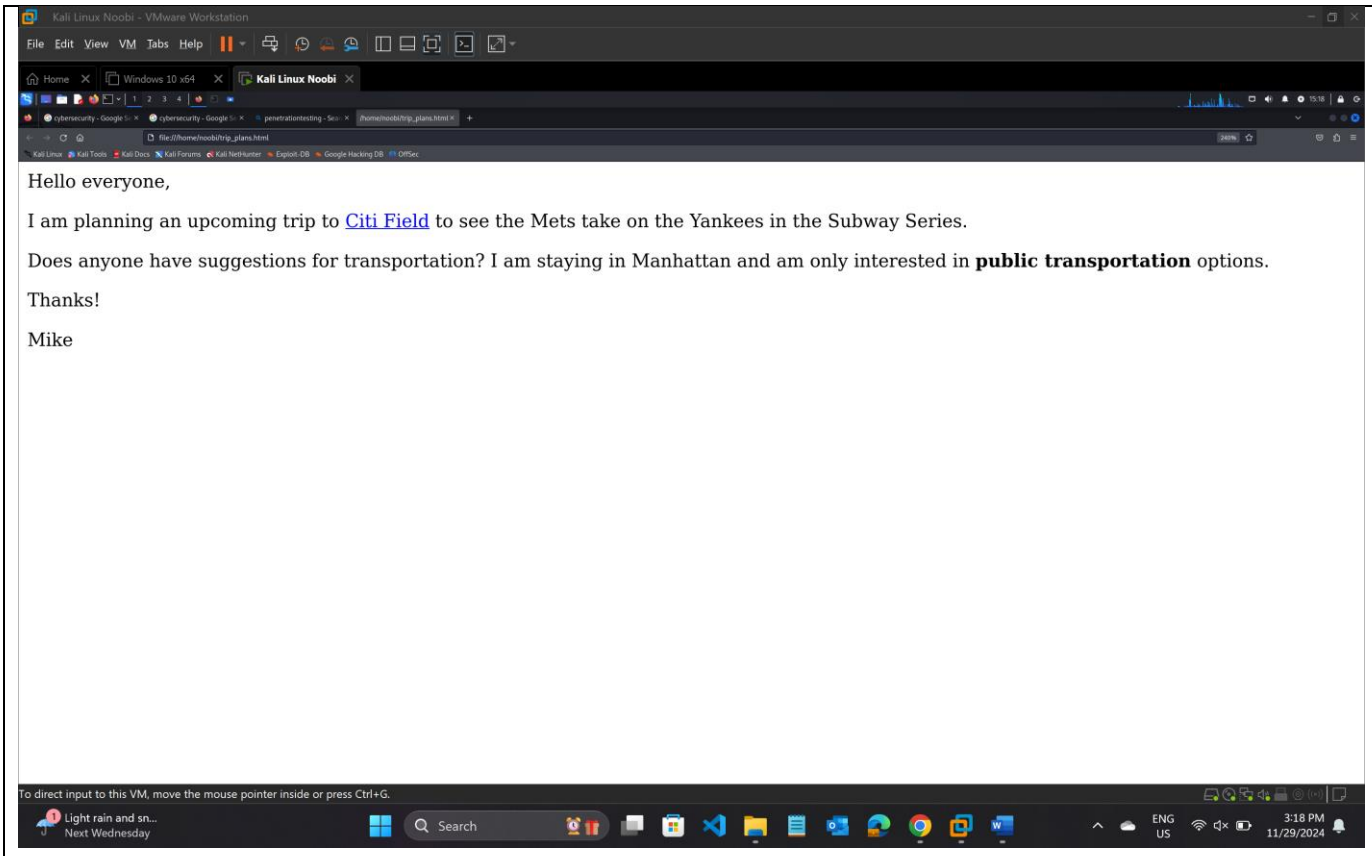
1. Using a text editor of your choice, create an HTML file containing some simple content of your choice. For example, you might want to model your code after the following sample page:

   * Hello everyone, *
   * I am planning an upcoming trip to <A HREF= 'https://www.mlb.com/mets/ballpark'>Citi Field</A> to see the Mets take on the Yankees in the Subway Series. *
   * Does anyone have suggestions for transportation? I am staying in Manhattan and am only interested in <B>public transportation</B> options. *
   * Thanks! *
   * Mike *



Saved the file

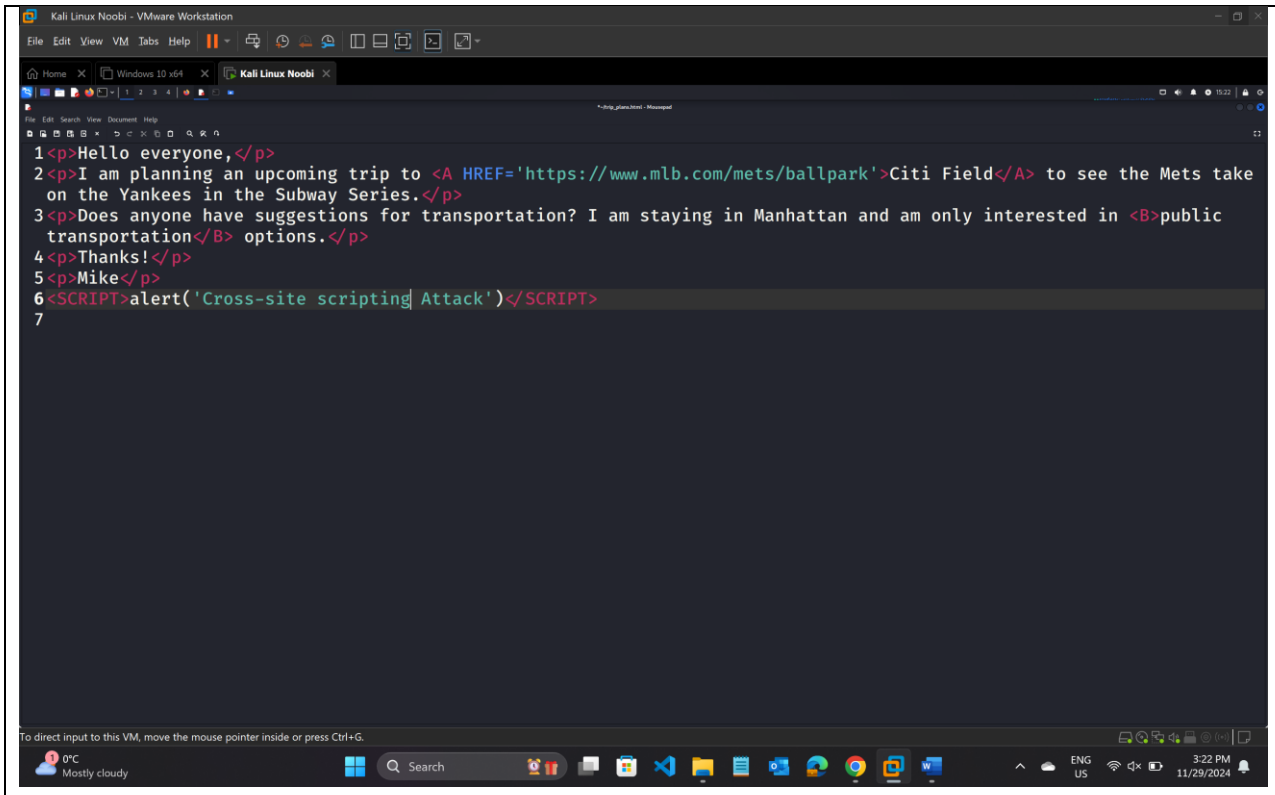
2. Open the file stored on your local computer and view it using your favorite browser.



Opened the file in a browser.

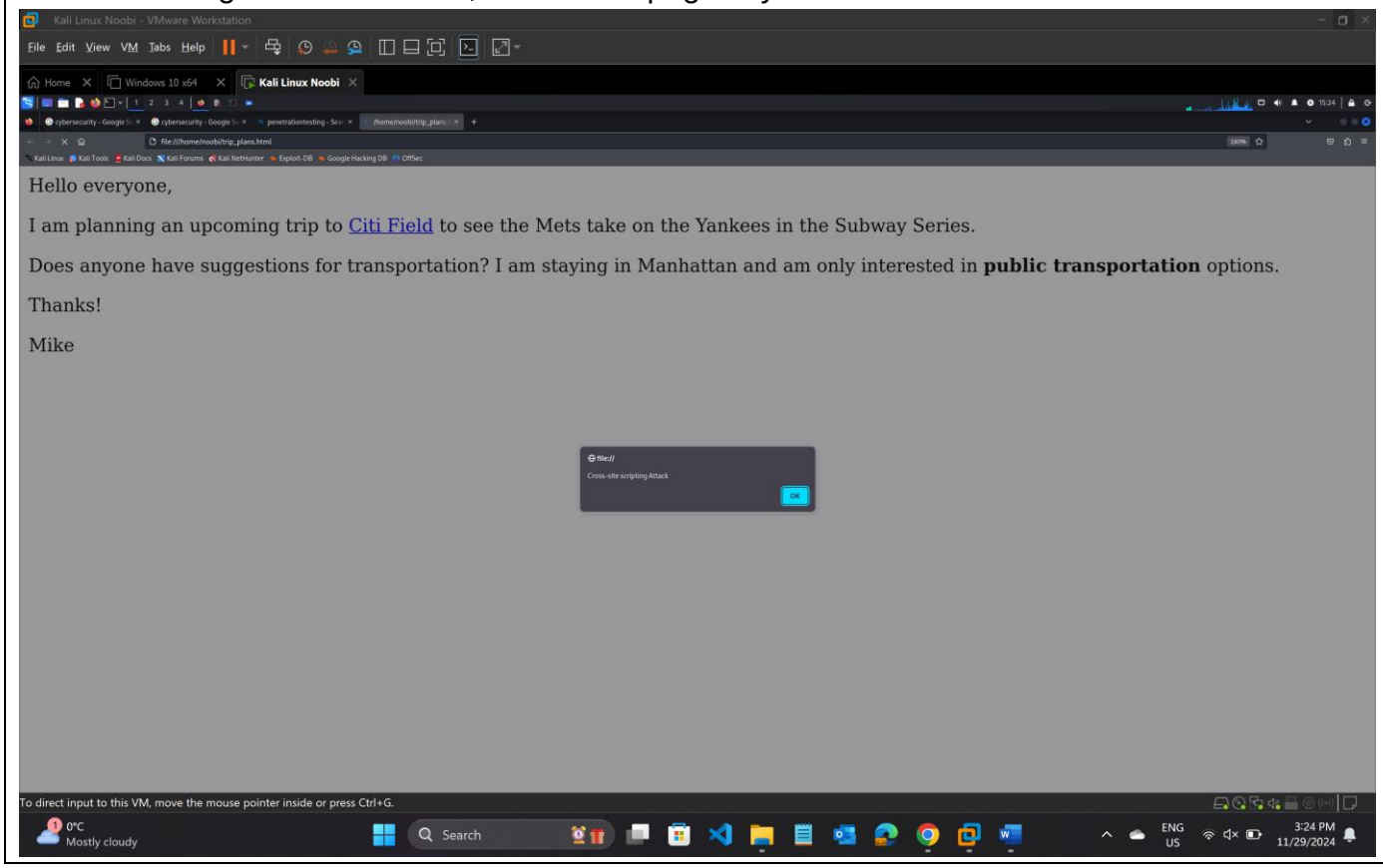
3. In your text editor, modify the file that you created in step 1 to include a cross-site scripting attack. You may wish to refer to the example in the section "Cross-Site Scripting (XSS)" did earlier, if you need assistance.
   * Hello everyone, *
   * I am planning an upcoming trip to <A HREF= 'https://www.mlb.com/mets/ballpark'>Citi Field</A> to see the Mets take on the Yankees in the Subway Series. *
   * Does anyone have suggestions for transportation? I am staying in Manhattan and am only interested in <B>public transportation</B> options. *
   * Thanks! *
   * Mike *
   <SCRIPT>alert('Cross-site scripting!')</SCRIPT>

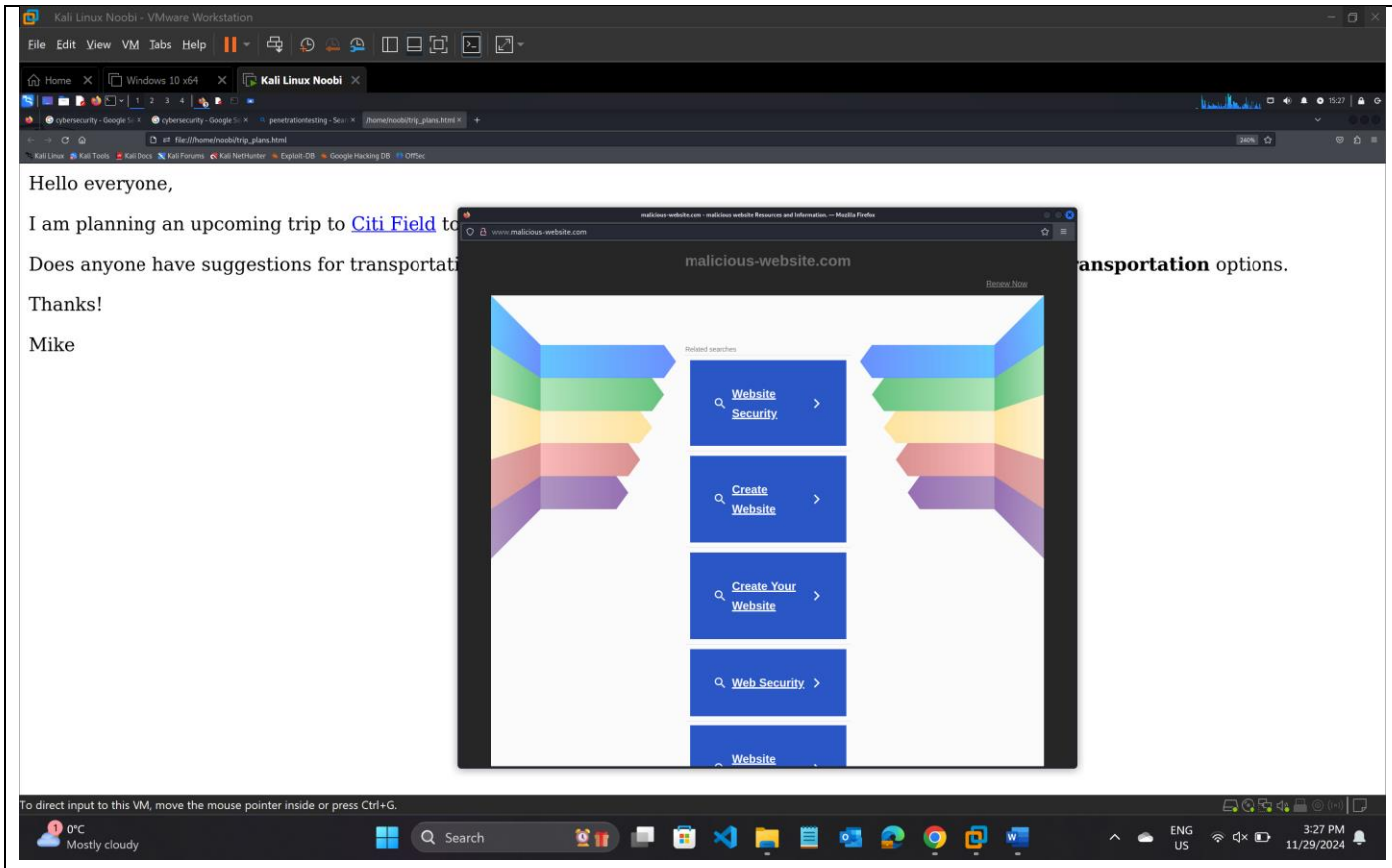
(You must change the last line to another line or lines of HTML code to include a cross site scripting attack.) **Take the screen shot of the modified code**.



Modified code.

4. After saving the modified file, refresh the page in your browser. **Take the screen shot.**





XSS script creates a pop-up window that display misleading information through the code : <SCRIPT> window.open('http://www.malicious-website.com', 'Malicious Popup', 'width=400,height=400'); </SCRIPT>

Q2: Did you see the impact of your cross-site scripting attack?

Yes, the impact of the cross-site scripting attack is visible as a popup message that says "Cross-site scripting Attack". This occurs because the browser executes the code embedded in the <SCRIPT> tag, which leads to the alert being triggered. This shows how XSS can be used to inject and execute malicious scripts in a webpage, leading to more serious security issues like stealing session cookies or performing other malicious actions.

Summary:

In this lab, I created a basic HTML page and then introduced a Cross-Site Scripting vulnerability by adding a <SCRIPT> tag that triggers an alert box. After saving and refreshing the page in my browser, I observed the alert with the message "Cross-site scripting Attack" which confirmed the success of the attack.

Activtiy#3: Exploiting Insecure Direct Object Reference (URL Manipulation)

First, we need to setup the lab. environment by creating a LAMP Server in Kali:

- Start Kali vm in VMWare or VirtualBox.
- Make sure Apache webserver is installed, it should be by default, so try to start it, if the service is not found then use apt-get to install it.

  systemctl start apache2

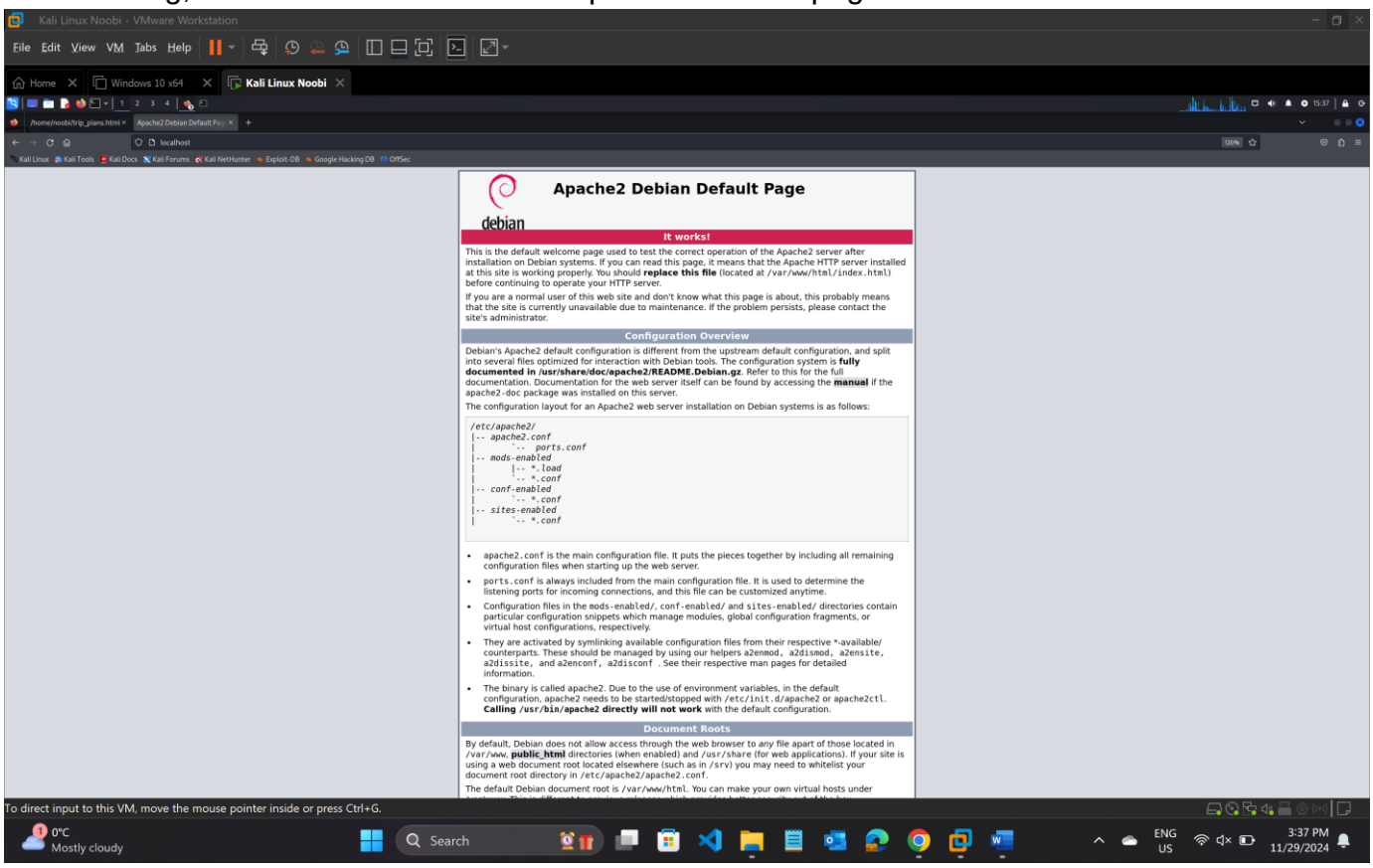
  The standard password is set as: kali

  (If you have changed Kali's password previously, then use that one.)

  - After starting Apache, check the status to make sure it is up and running.

    systemctl status apache2

- Open a web browser and go to the "localhost" address to make sure the website is up and running, it should show the default Apache2 Debian page.



default Apache2 Debian page.

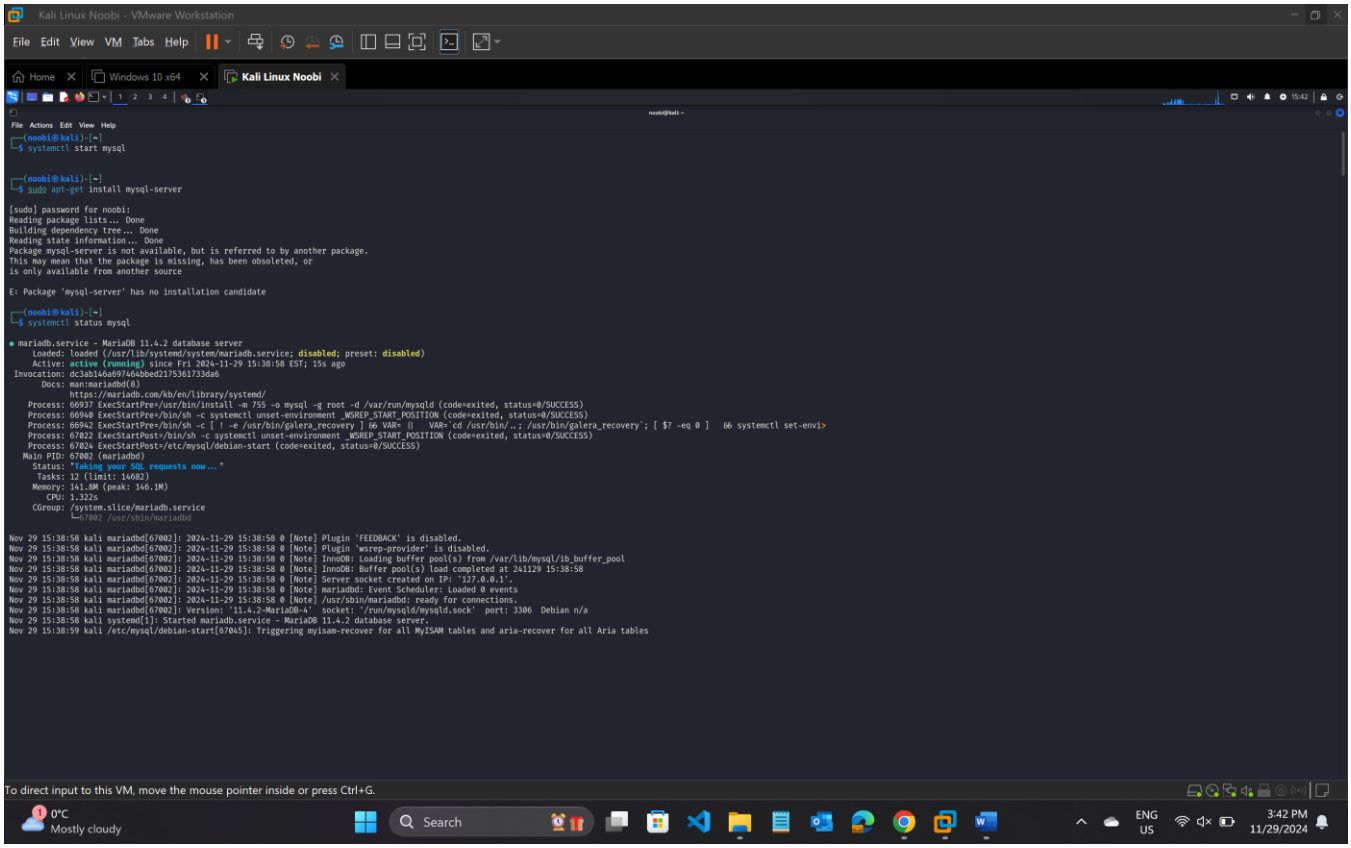
- Now that Apache is installed and running, make sure that mysql is installed. Try to start the mysql service, if it is not found, install it with apt-get.

  systemctl start mysql

  The standard password is set as: kali

- Check the status of mysql to make sure it is running.

  systemctl status mysql



Install and start MySQL

MySQL database and user is setup

- Now that mysql is up and running, we have to setup the database. Login to mysql as the root user.

  sudo mysql --user=root –password

- Create the database - call it CYB302. NOTE the capital, it is important to make sure it is capitalized because the PHP files that connect to the database is case-sensitive. Also make sure to use the semi-colon ; to end the statement

  CREATE DATABASE CYB302;

- Verify that the database was created correctly by using the show databases command.

  SHOW DATABASES;

- Now we have to create a user for accessing the database and setup the user's privileges. The username is "mohamed" and the password is "S!d@q!##". Copy and paste this command, it is actually several commands linked together by statement terminating semi-colons ; make sure they all respond with Query OK. <u>Do not change anything in the below commands at all.</u>

  CREATE USER 'mohamed'@'%' IDENTIFIED BY 'S!d@q!##';

GRANT SELECT ON *.* TO 'mohamed'@'%';
ALTER USER 'mohamed'@'%' REQUIRE NONE WITH MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0 MAX_UPDATES_PER_HOUR 0 MAX_USER_CONNECTIONS 0;
GRANT ALL PRIVILEGES ON `mohamed`.* TO 'mohamed'@'%';

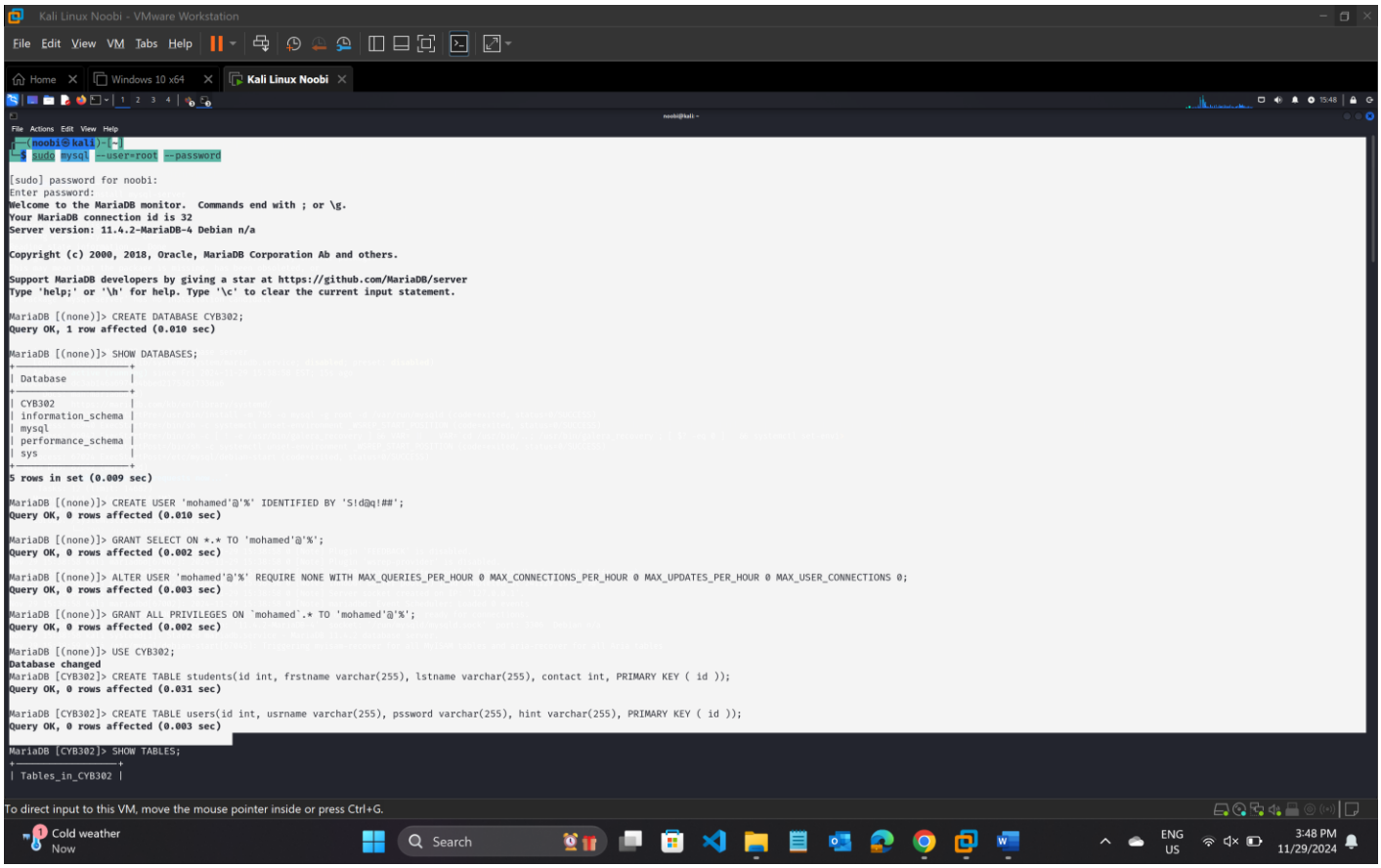
- Now create the tables. First select the database.

  USE CYB302;

- Now make two tables, a students table that holds first and last name of students, and a users table that holds users usernames and passwords.

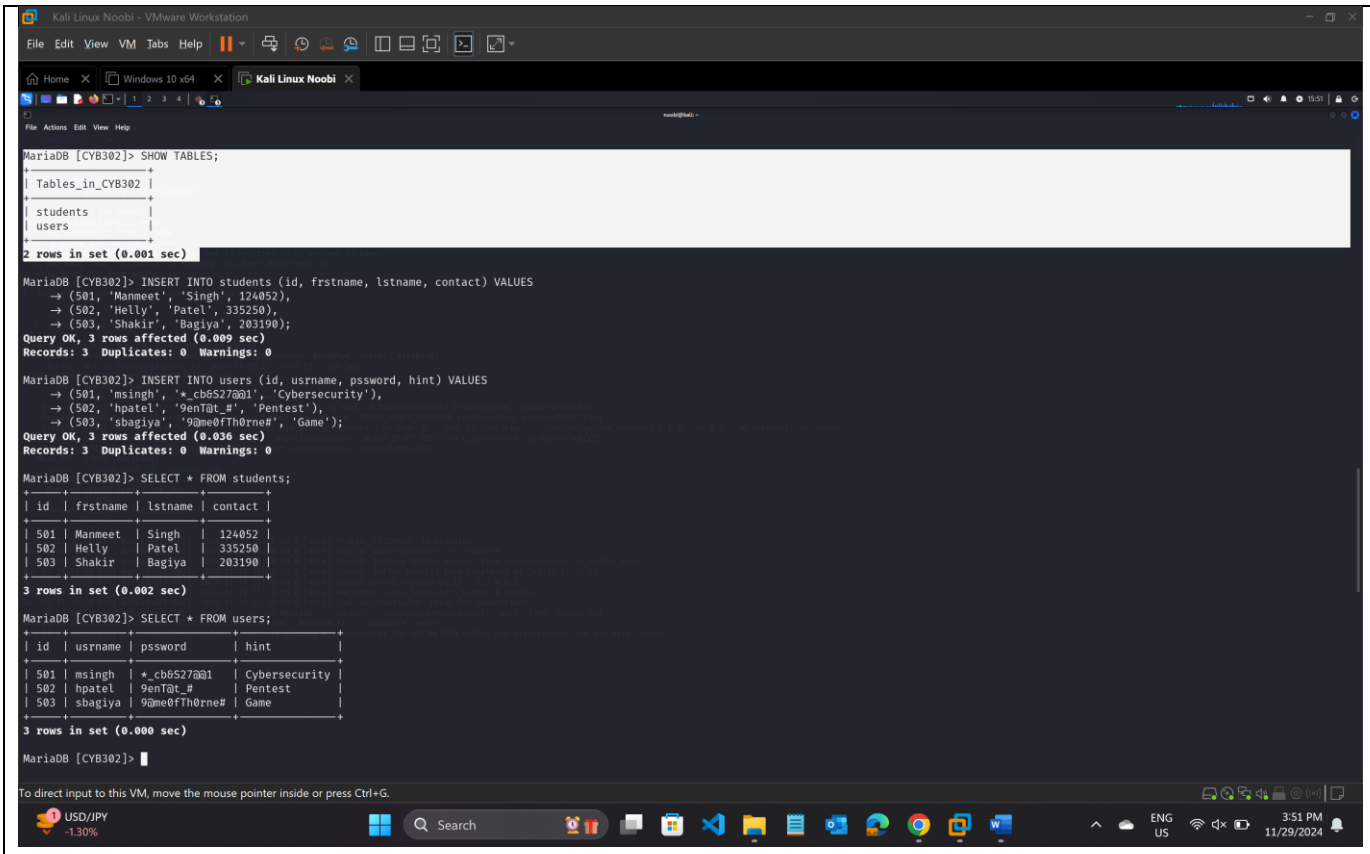
  CREATE TABLE students(id int, frstname varchar(255), lstname varchar(255), contact int, PRIMARY KEY (id));

  CREATE TABLE users(id int, usrname varchar(255), pssword varchar(255), hint varchar(255), PRIMARY KEY (id));



- Finally verify the tables were created correctly by display the tables.

  SHOW TABLES; (Take the screen shot)



Verify the tables were created for students and users.

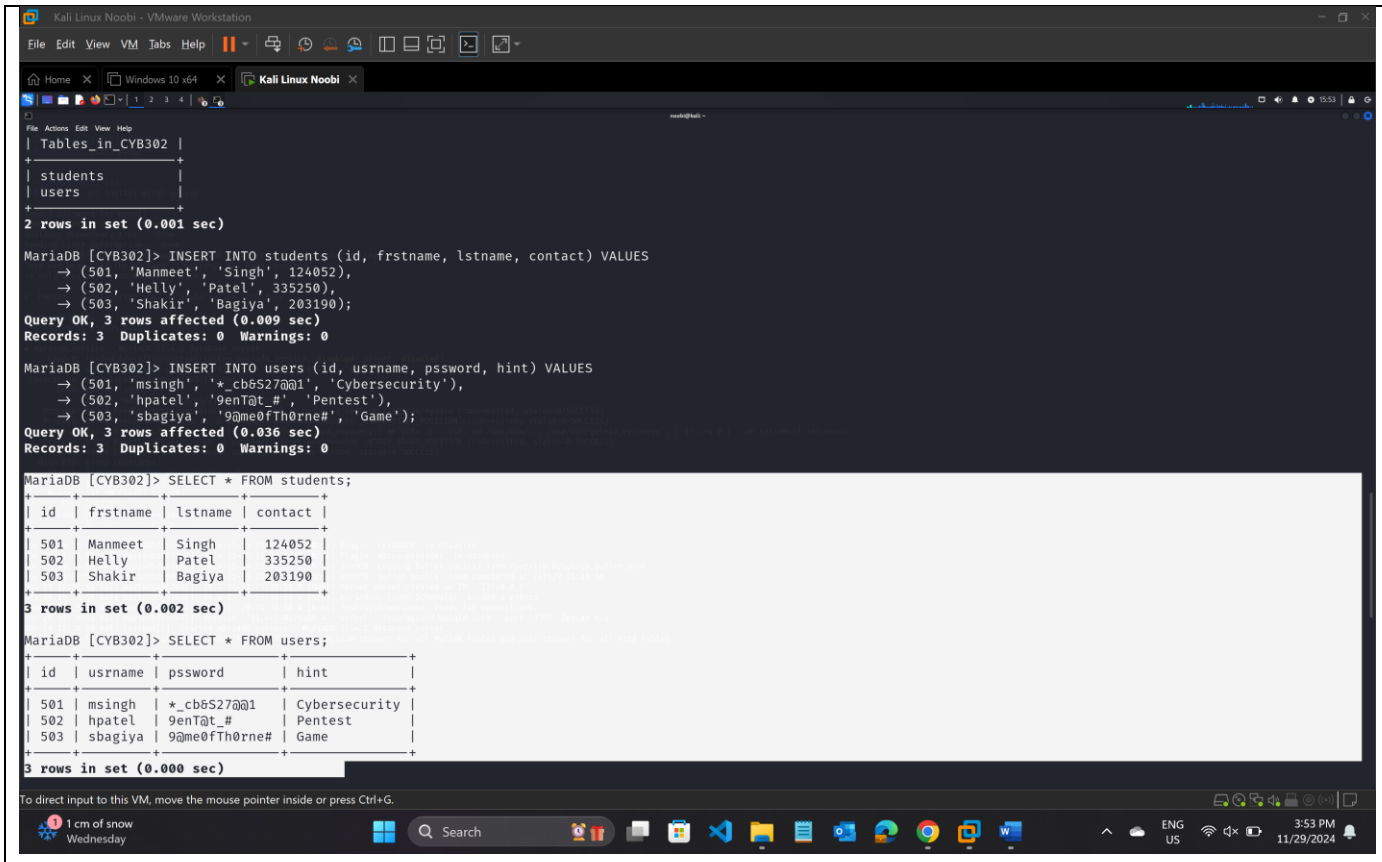
- Insert some data into the “students” table and the “users” table. Feel free to change the values to other names, usernames, and passwords.

  INSERT INTO `students` (`id`, `frstname`, `lstname`, `contact`) VALUES ('501', 'Manmeet', 'Singh', '124052'), ('502', 'Helly', 'Patel', '335250'), ('503', 'Shakir', 'Bagiya', '203190');

  INSERT INTO `users` (`id`, `usrname`, `pssword`, `hint`) VALUES ('501', 'msingh', '*_cb&S27@@1', 'Cybersecurity'), ('502', 'hpatel', '9enT@t_#', 'Pentest'), ('503', 'sbagiya', '9@me0fTh0rne#', 'Game');

- Read back the data from the tables to make sure that it was inserted correctly.

  SELECT * from students;
  SELECT * from users;
  (Take the screen shot showing output of both above-mentioned commands)



Verifying the inserted data using select * from *table_name;*

- Finally, exit out of the mysql command terminal.

  exit

- Download the following two PHP files form.php and doit.php

- Open a terminal and change directory to your Downloads directory. Make sure that the two files are there in the downloads directory by using the **ls command**.
- Make a new directory called "**cyb**" in the webserver root directory with the following command:

  sudo mkdir /var/www/html/cyb

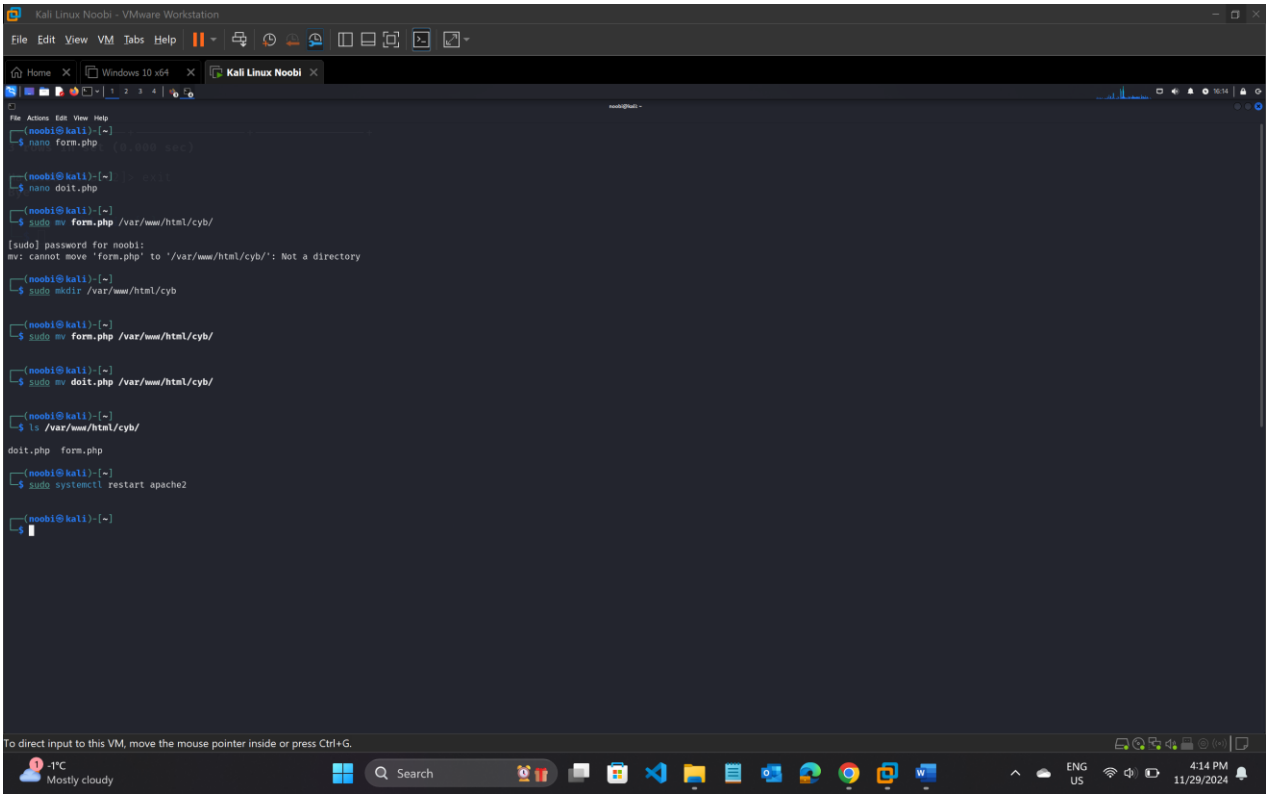
  Note that the /var/www/html directory is the default webserver root directory for apache, also make sure you use **sudo** with the **mkdir** command because this directory is owned by root and regular users will not have permission to make new directories.
- Finally, move the two PHP files from the current directory to the **cyb** directory in the webserver root with the following command:

  sudo mv doit.php form.php /var/www/html/cyb

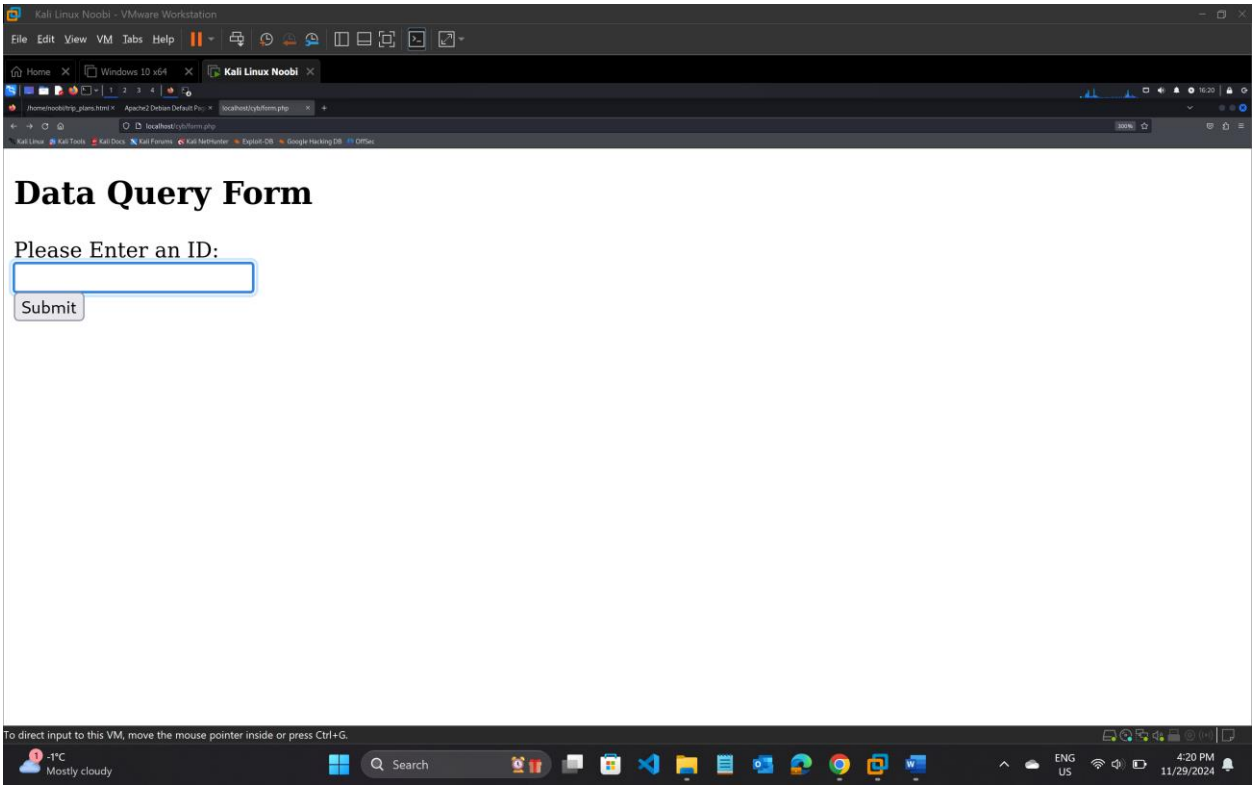
  Note once again that you must use **sudo** since the directory is owned by the root user.

- Restart the Apache webserver

systemctl restart apache2



Moved the PHP files to the web server directory after making the directory /var/www/html/cyb/

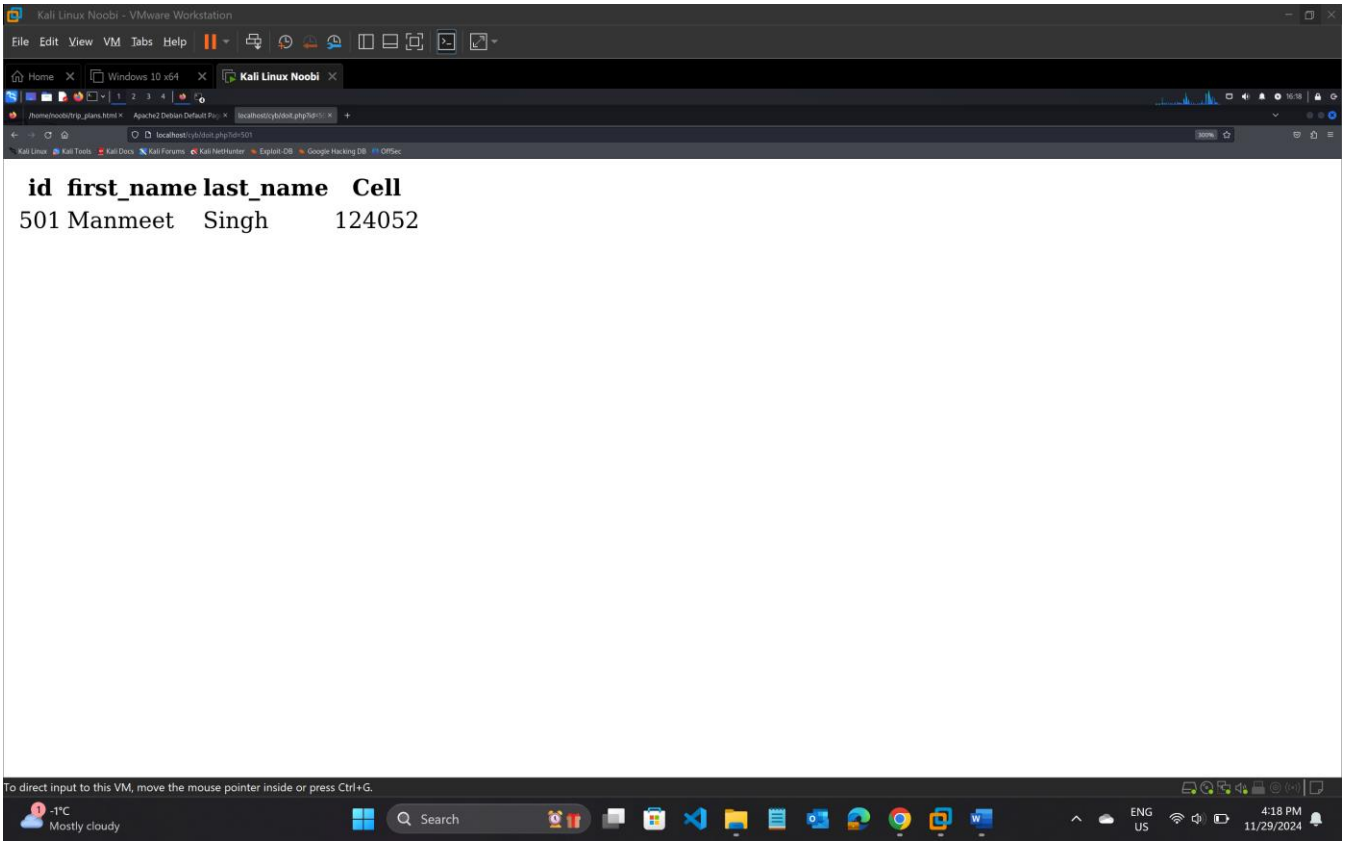


visited the address “localhost/cyb/form.php” in your web browser: The form page.

URL Manipulation:

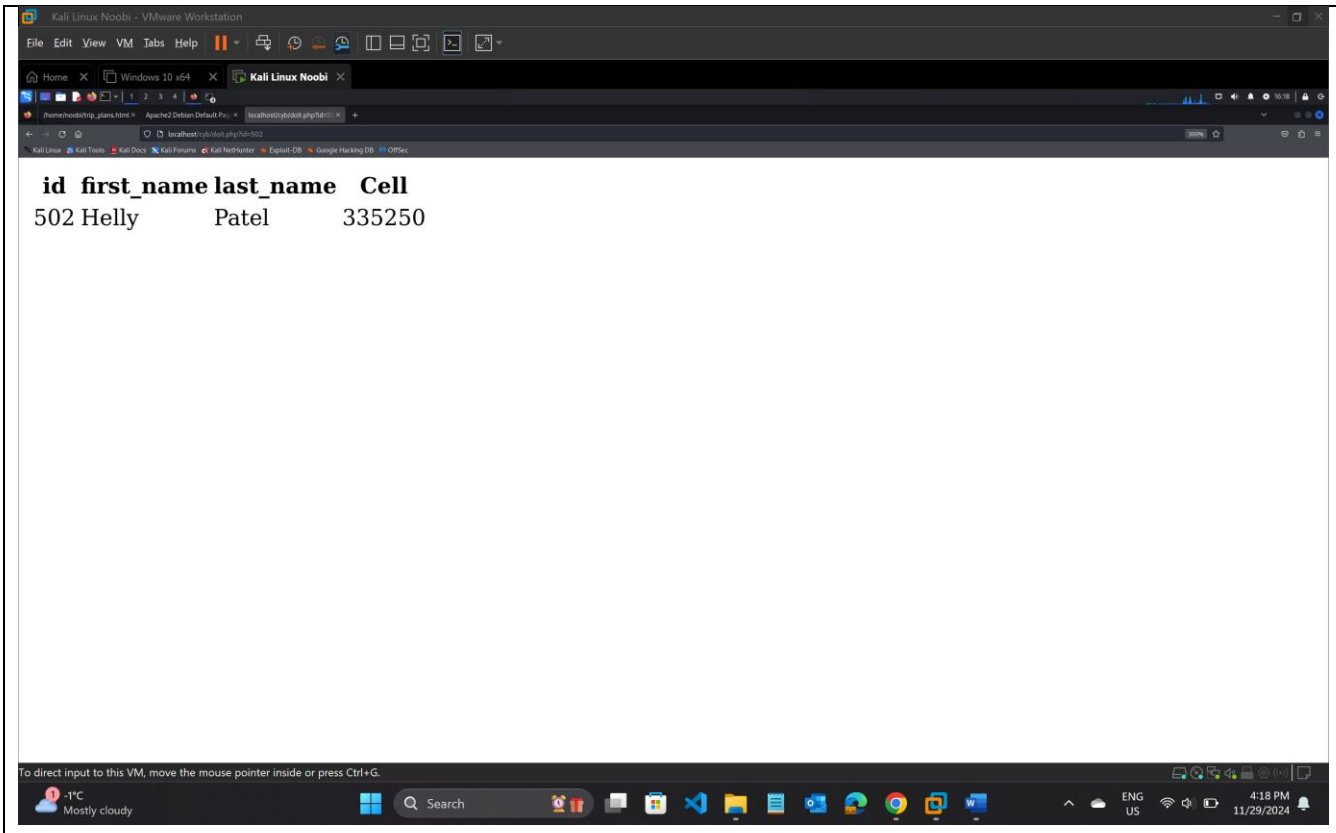
URL manipulation is a starting point with SQL injection that allows you to change the variables that websites use to communicate between the back and front end.

- Navigate to the site “**localhost/cyb/form.php**” and enter the number 501 in the ID field, click submit. **Take the screen shot.**

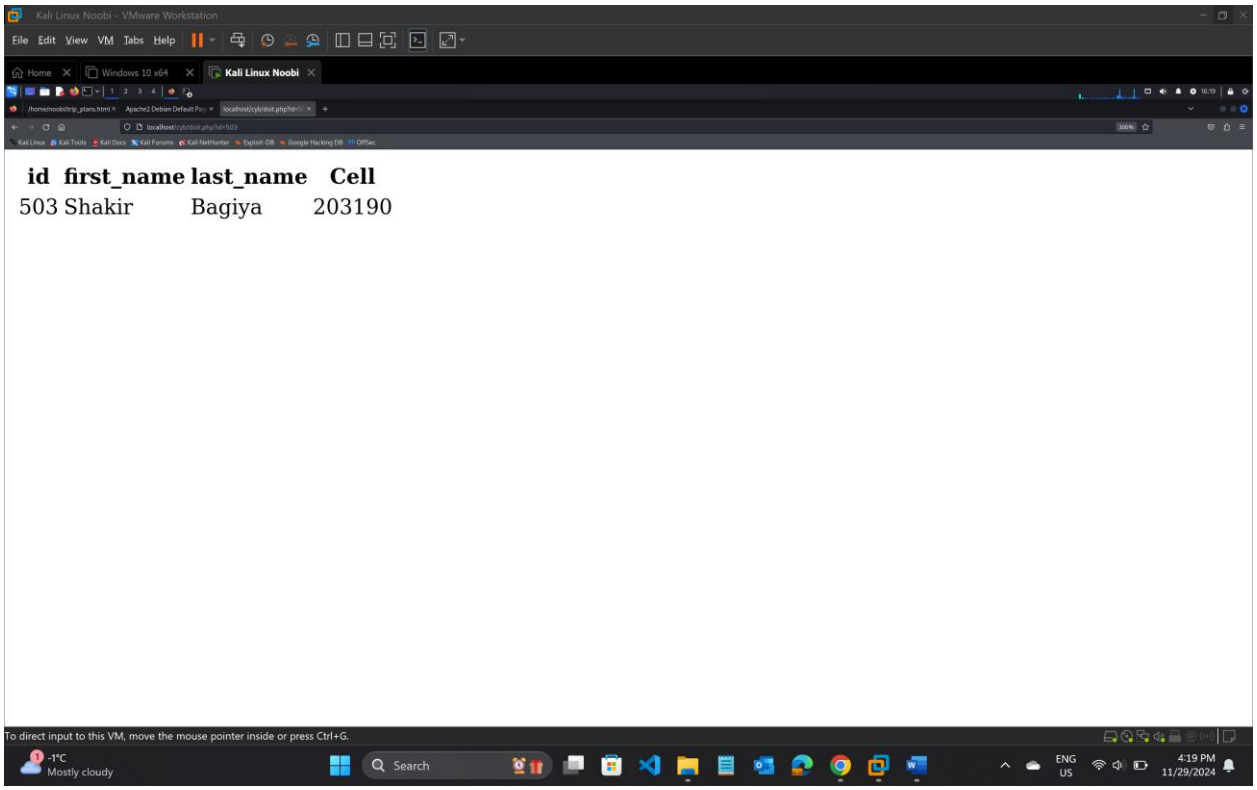


Resulting PHP code returns a page that lists the record associated with ID number 501. This displays the record for Manmeet Singh.

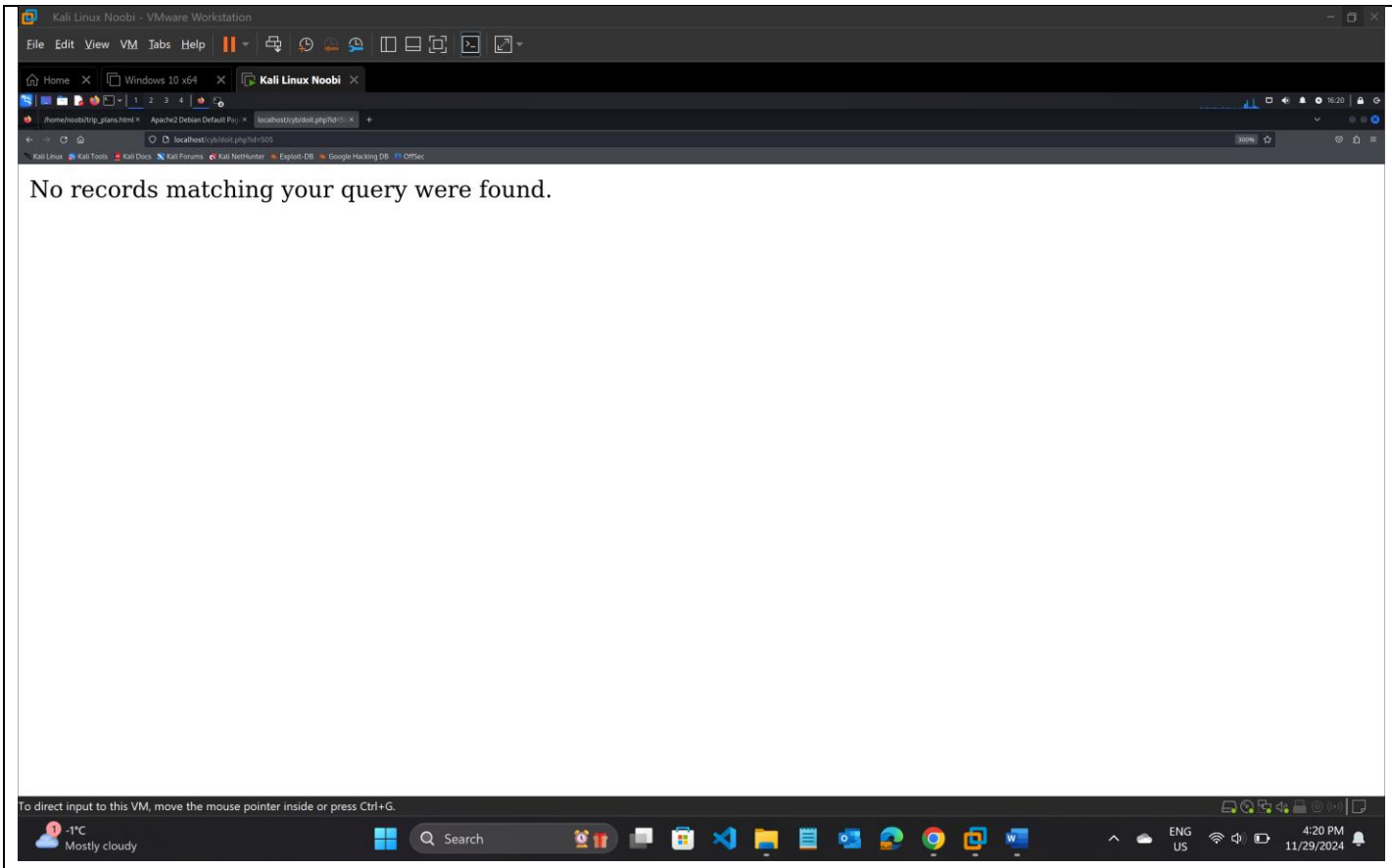
- Modify the URL to show you the record associated with the following ID numbers:



- 502**.** This displays the record for Helly Patel



- 503**.** This displays the record for Shakir Bagiya



- 505 This ID might not exist, it returns an error.

Summary:

In this lab, I studied Insecure Direct Object References (IDOR) through URL manipulation. IDOR occurs when users can modify URL parameters to manipulate data, they should not have access to. By changing the id parameter in the URL like that in localhost/cyb/form.php?id=501, I was able to view records that were not intended for me or other unintended users, showing the vulnerability. To prevent IDOR, web applications should validate and restrict user access to resources based on proper permissions.