ACMEGRADE CyberSecurity
December’23

PROJECT – 1

BY – Priyanka Hichkad

INDEX

WHAT IS SQL (Structured query language)?

SQL injection, also known as SQLI, is a common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed. This information may include any number of items, including sensitive company data, user lists or private customer details.

The impact SQL injection can have on a business is far-reaching. A successful attack may result in the unauthorized viewing of user lists, the deletion of entire tables and, in certain cases, the attacker gaining administrative rights to a database, all of which are highly detrimental to a business.

When calculating the potential cost of an SQLi, it's important to consider the loss of customer trust should personal information such as phone numbers, addresses, and credit card details be stolen.

While this vector can be used to attack any SQL database, websites are the most frequent targets.

Types of SQL Injections:

1) In-band SQLi-

The attacker uses the same channel of communication to launch their attacks and to gather their results. In-band SQLi's simplicity and efficiency make it one of the most common types of SQLi attack. There are two sub-variations of this method:

- **Error-based SQLi**—the attacker performs actions that cause the database to produce error messages. The attacker can potentially use the data provided by these error messages to gather information about the structure of the database.
- **Union-based SQLi**—this technique takes advantage of the UNION SQL operator, which fuses multiple select statements generated by the database to get a single HTTP response. This response may contain data that can be leveraged by the attacker.

2) Inferential (Blind) SQLi-

The attacker sends data payloads to the server and observes the response and behaviour of the server to learn more about its structure. This method is called blind SQLi because the data is not transferred from the website database to the attacker, thus the attacker cannot see information about the attack in-band.

Blind SQL injections rely on the response and behavioural patterns of the server so they are typically

slower to execute but may be just as harmful. Blind SQL injections can be classified as follows:

- **Boolean**—that attacker sends a SQL query to the database prompting the application to return a result. The result will vary depending on whether the query is true or false. Based on the result, the information within the HTTP response will modify or stay unchanged. The attacker can then work out if the message generated a true or false result.
- **Time-based**—attacker sends a SQL query to the database, which makes the database wait (for a period in seconds) before it can react. The attacker can see from the time the database takes to respond, whether a query is true or false. Based on the result, an HTTP response will be generated instantly or after a waiting period. The attacker can thus work out if the message they used returned true or false, without relying on data from the database.

3) Out-of-band SQLi-

The attacker can only carry out this form of attack when certain features are enabled on the database server used by the web application.

Out-of-band SQLi is performed when the attacker can’t use the same channel to launch the attack and gather information, or when a server is too slow or unstable for these actions to be performed. These techniques count on the capacity of the server to create DNS or HTTP requests to transfer data to an attacker.

What is XSS?

Cross site scripting (XSS) is a web vulnerability that lets a malicious hacker introduce (inject) undesired commands into legitimate client-side code (usually JavaScript) executed by a browser on behalf of the web application.

Most websites and web applications run client-side code in the web browser using some kind of dynamic scripting language, mostly-JavaScript. You can assume that more than 99% of websites and web applications include this code. Thus, it means that user browsers must be able to interpret any JavaScript code on behalf of the web application.

Most web applications and websites also interact with users in some way, even without JavaScript. For example, the user may need to type their username to log in and the application may display that username later in the user interface. This means that the application processes user input and then outputs it in the web browser.

Combined, these two conditions lay the foundation for cross-site scripting, (an injection attack). If an attacker is able to include JavaScript code in a user input parameter and the application directly returns that code in its HTML output and sends it to the client browser, the browser will execute the malicious JavaScript. Whenever a web page directly echoes user input, attackers will be able to run malicious scripts in the client browser, even if the page itself is built only with static HTML tags and includes no JavaScript.

Unlike most other web application vulnerabilities, this one does not directly affect the back end. It affects regular users of the web application or victims who are tricked into accessing it. XSS is also possible for some APIs that allow JavaScript.

TYPES OF XSS:

Stored XSS (Persistent XSS)

The most damaging type of XSS is Stored XSS (Persistent XSS). An attacker uses Stored XSS to inject malicious content (referred to as the payload), most often JavaScript code, into the target application. If there is no input validation, this malicious code is permanently stored (persisted) by the target application, for example within a database. For example, an attacker may enter a malicious script into a user input field such as a blog comment field or in a forum post.

When a victim opens the affected web page in a browser, the XSS attack payload is served to the victim's browser as part of the HTML code (just like a legitimate comment would). This means that victims will end up executing the malicious script once the page is viewed in their browser.

Reflected XSS (Non-persistent XSS)

The second and the most common type of XSS is Reflected XSS (Non-persistent XSS). In this case, the attacker's payload has to be a part of the request that is sent to the web server. It is then reflected back in such a way that the HTTP response includes the payload from the HTTP request. Attackers use malicious links, phishing emails, and other social engineering techniques to lure the victim into making a request to the server. The reflected XSS payload is then executed in the user's browser.

Reflected XSS is not a persistent attack, so the attacker needs to deliver the payload to each victim. These attacks are often made using social networks.

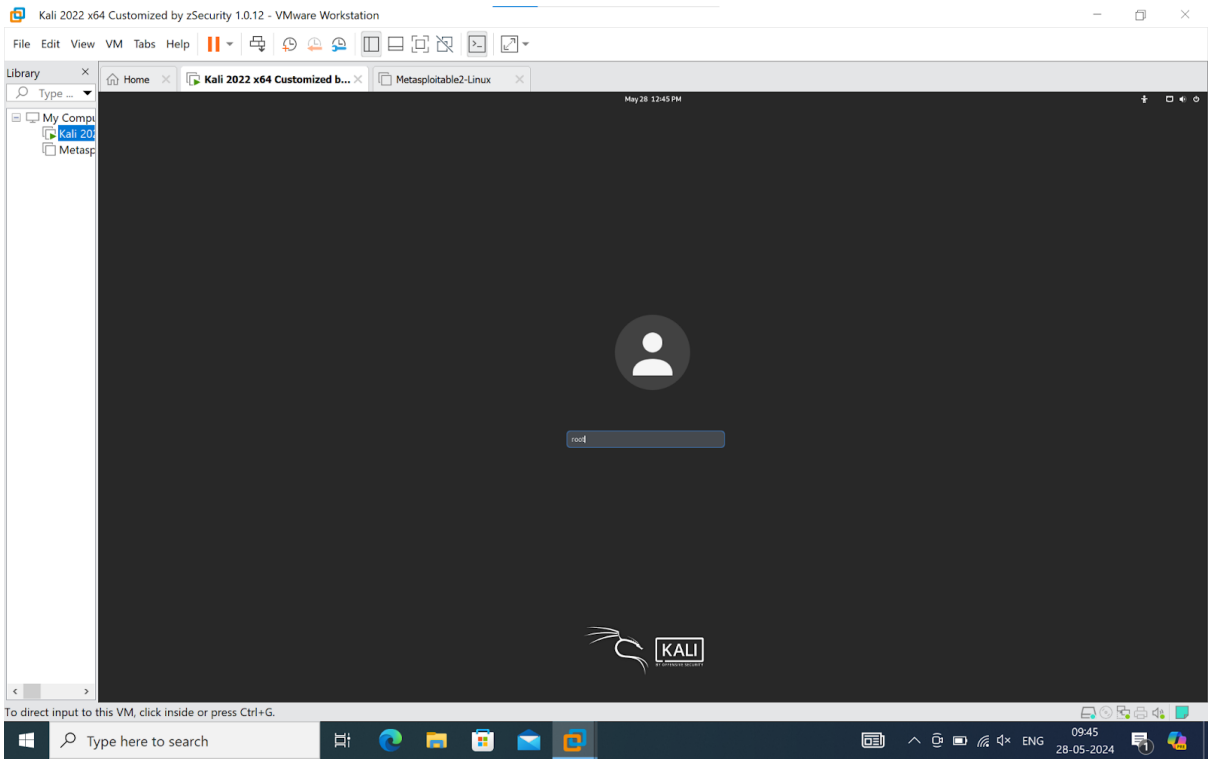
DOM-based XSS

DOM-based XSS is an advanced XSS attack. It is possible if the web application's client-side scripts write data provided by the user to the Document Object Model (DOM). The data is subsequently read from the DOM by the web application and outputted to the browser. If the data is incorrectly handled, an attacker can inject a payload, which will be stored as part of the DOM and executed when the data is read back from the DOM.

A DOM-based XSS attack is often a client-side attack and the malicious payload is never sent to the server. This makes it even more difficult to detect for Web Application Firewalls (WAFs) and security engineers who analyse server logs because they will never even see the attack. DOM objects that are most often manipulated include the URL, the anchor part of the URL, and the Referrer.

FINDING VULNERABILITIES:

1)Open the Kali Linux:



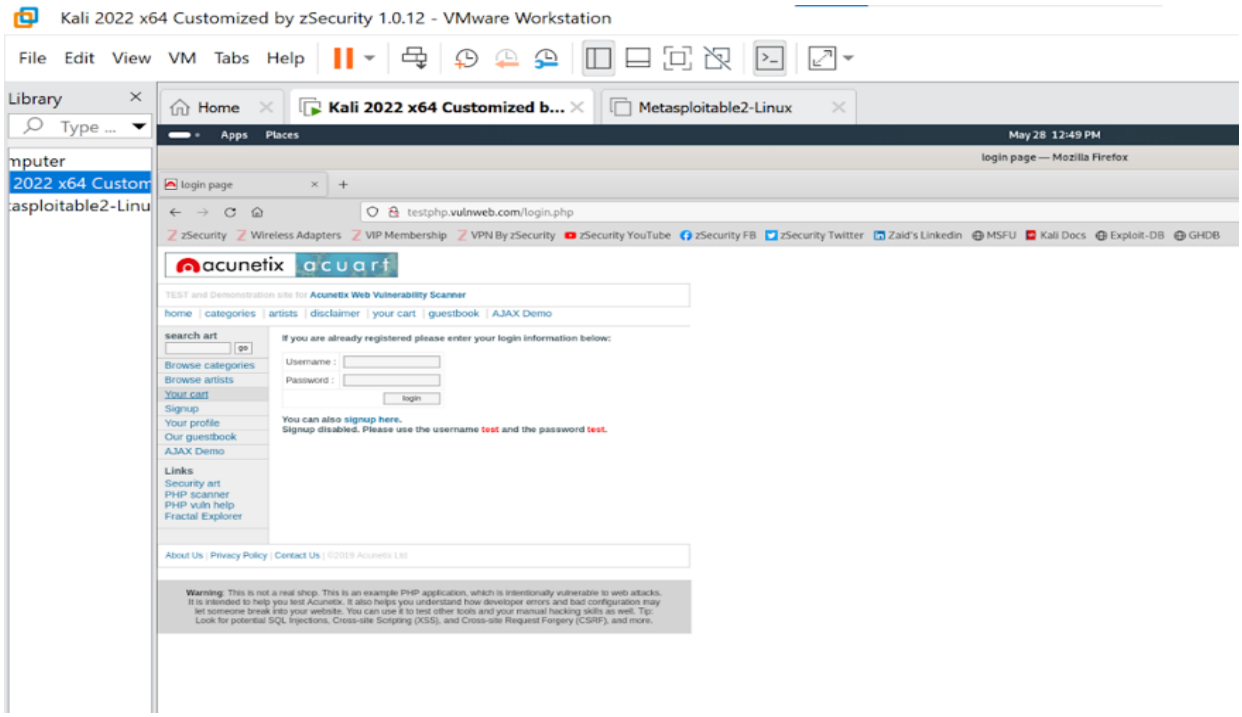
Enter your username and password.

The machine is started.

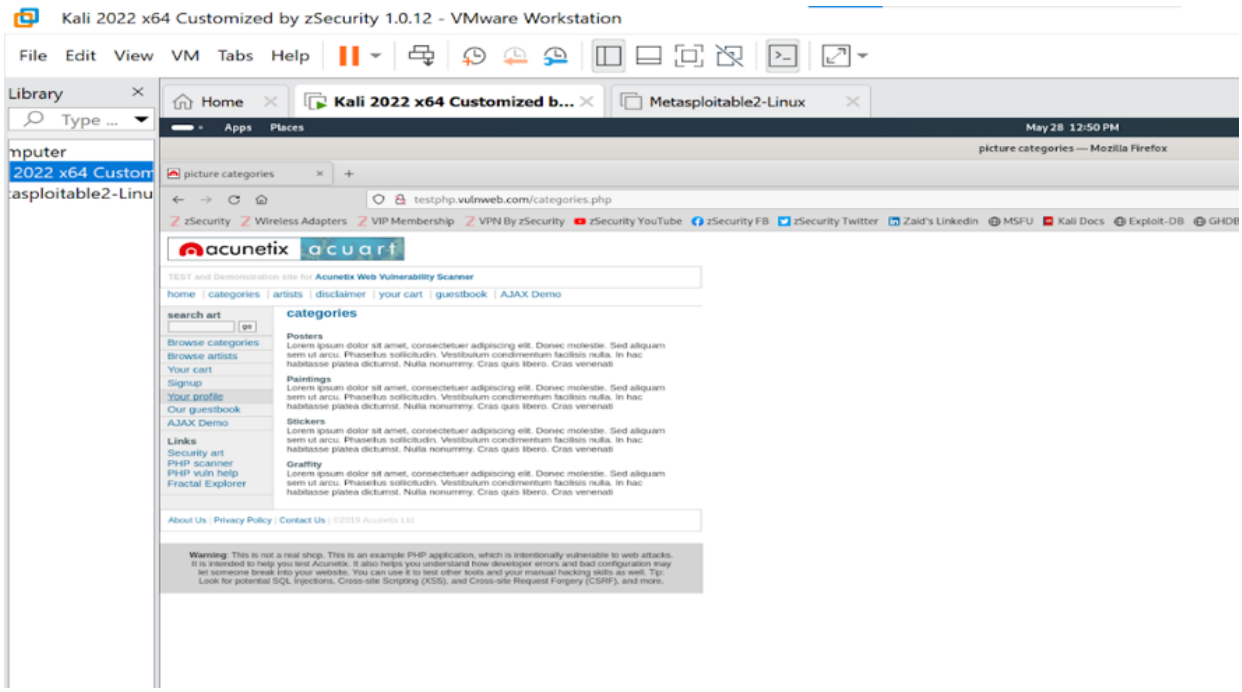
USING SQLMAP:

Our Target: http://testphp.vulnweb.com/login.php

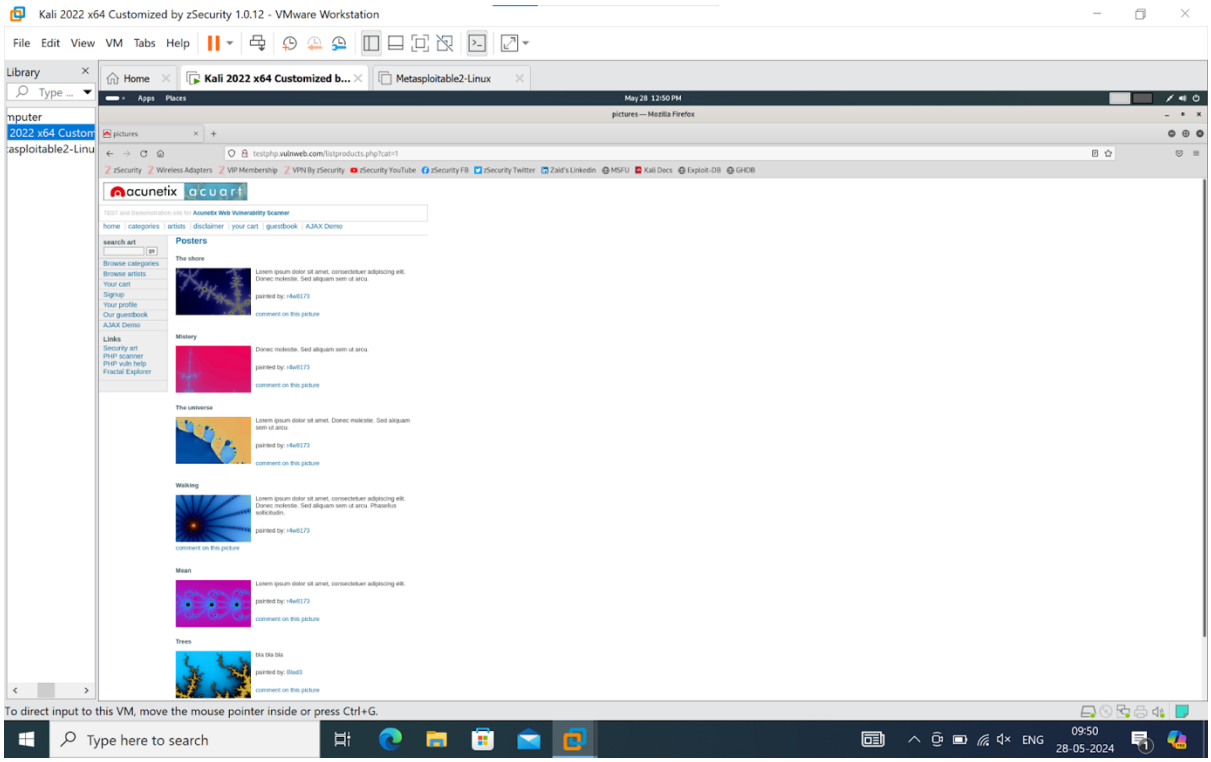
1)Go to the Target Website-



2)Go to categories -



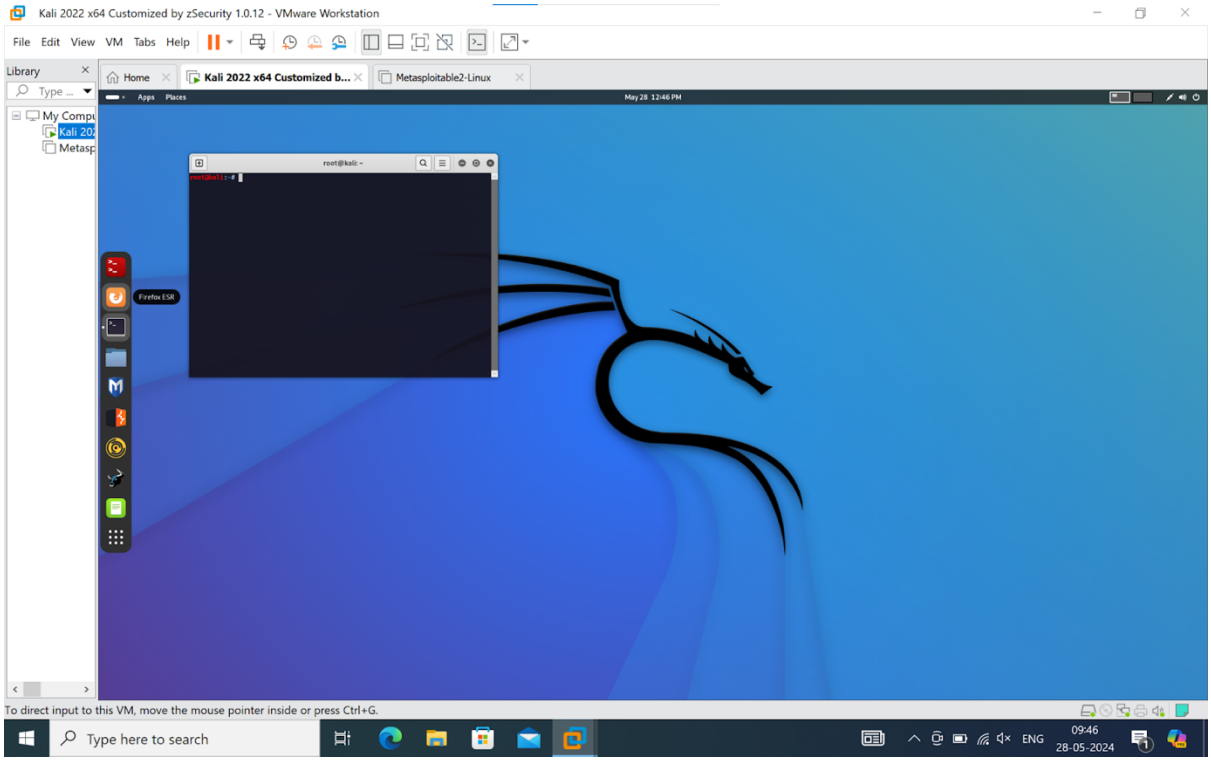
3)Go to posters –



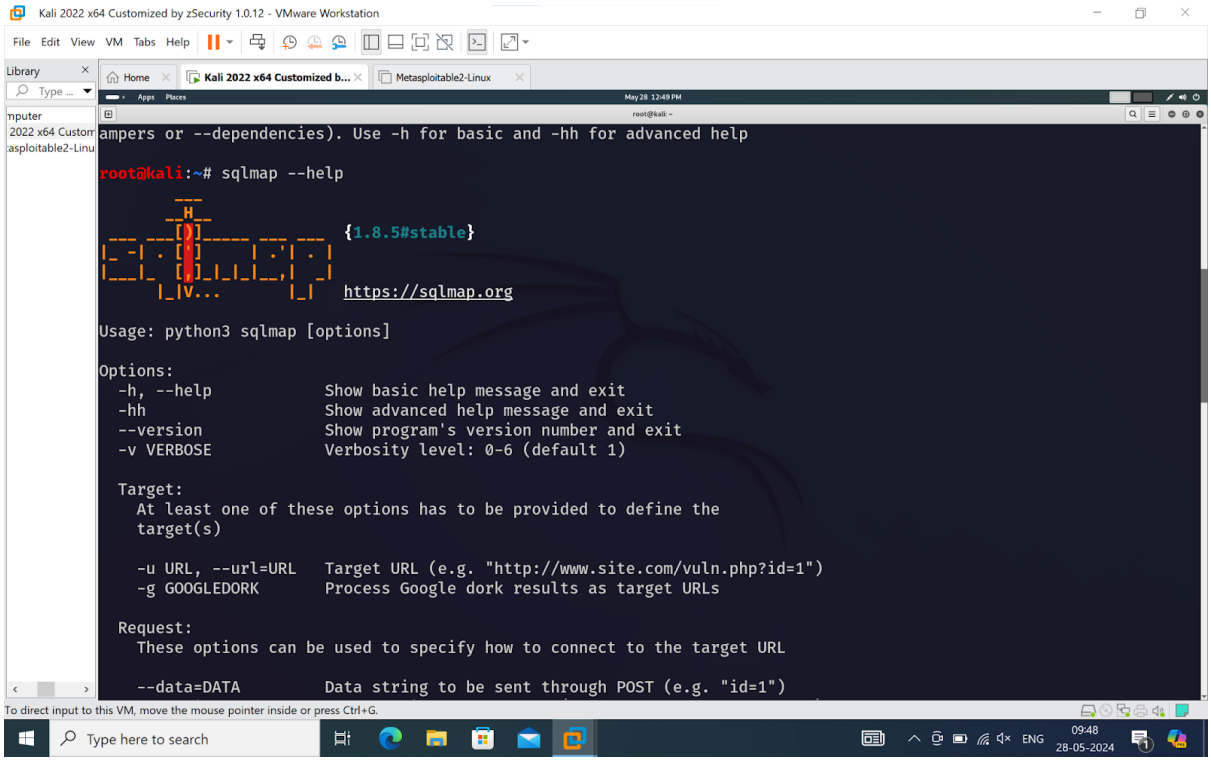
4)Now our new target link is: http://testphp.vulnweb.com/listproducts.php?cat=1

This kind of link is prone to SQL attacks and then we can catch its vulnerabilities.

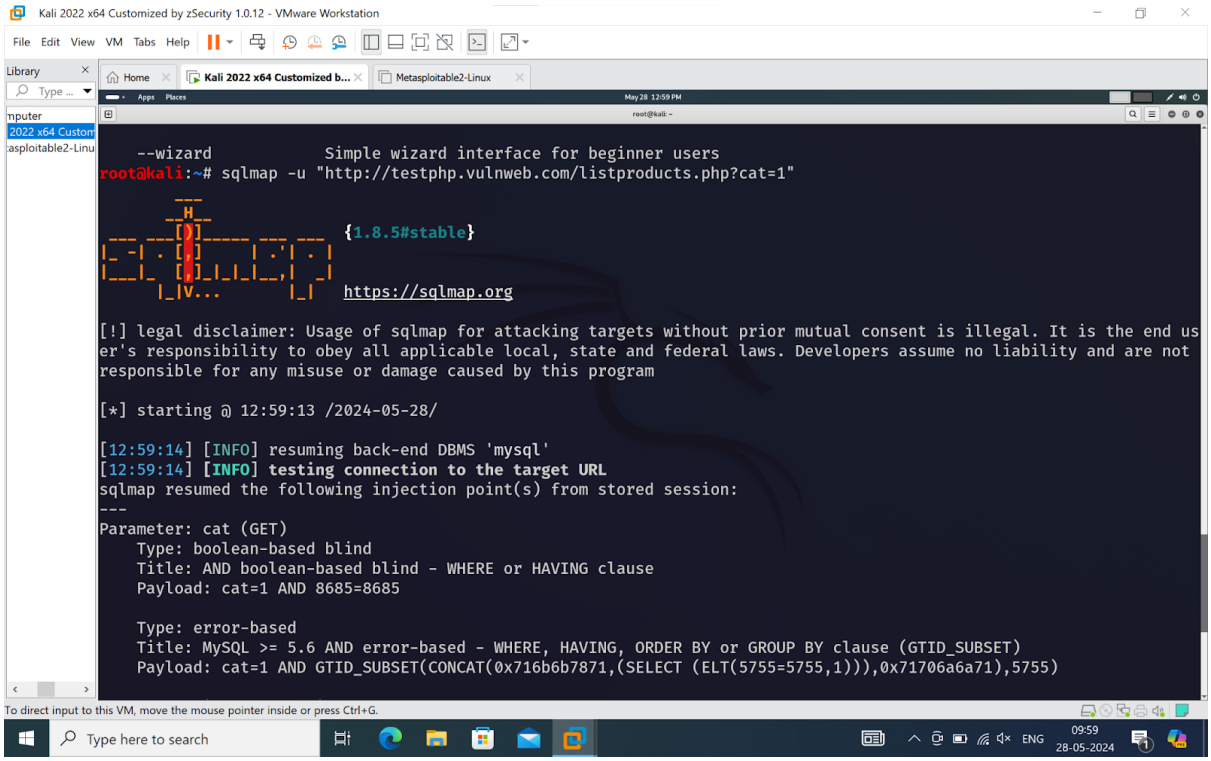
5)Open the Terminal



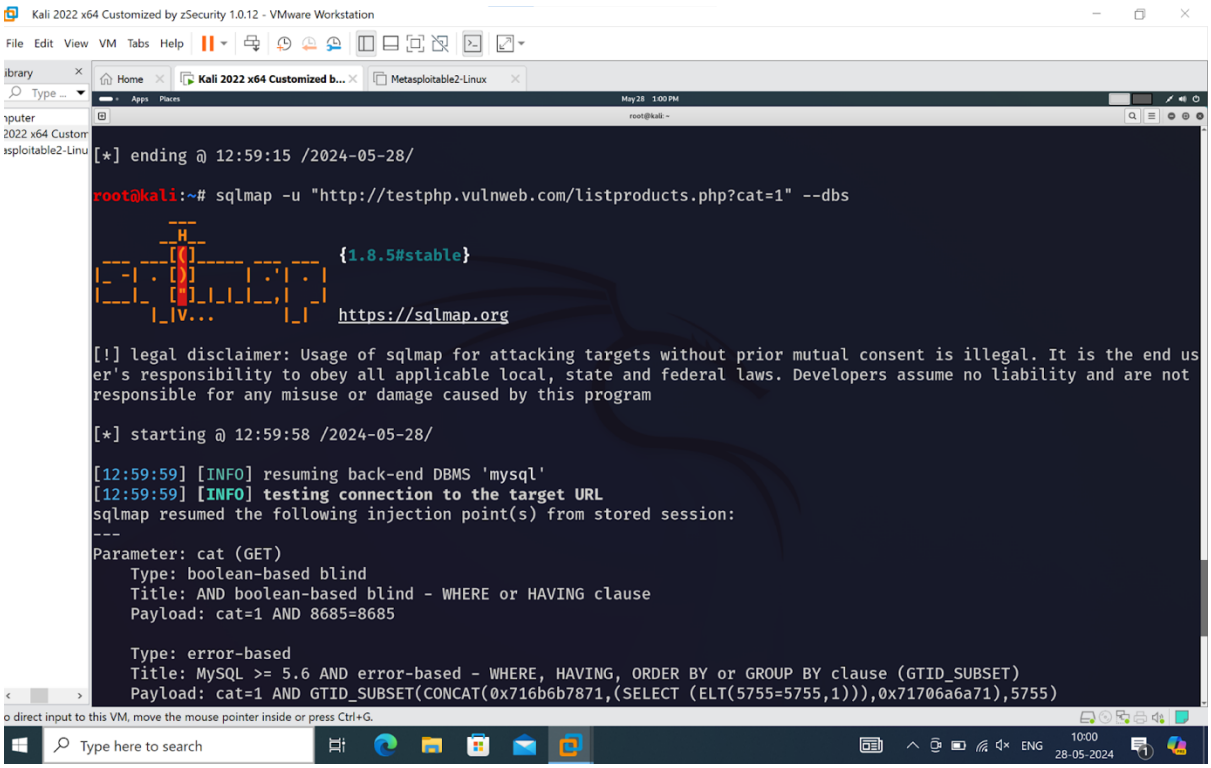
6)Open SQLMAP help options – it shows us different types of commands and their use.



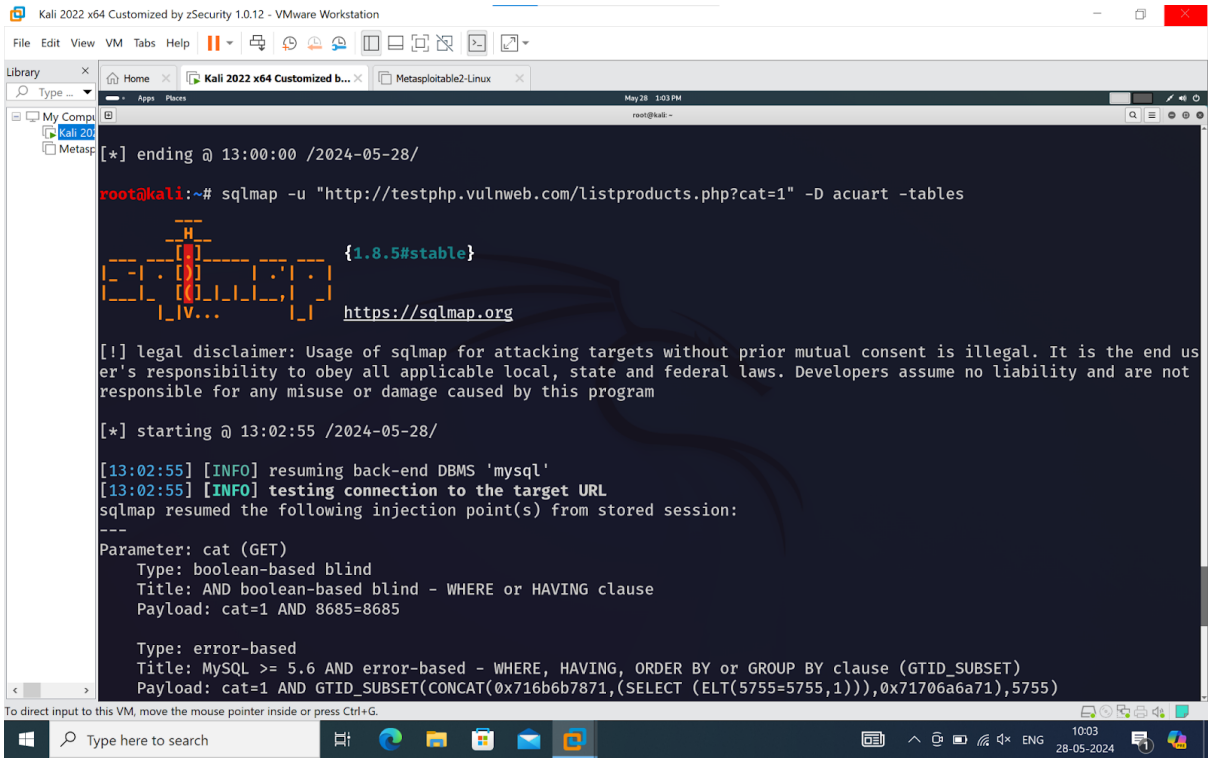
7)Scan the Target-

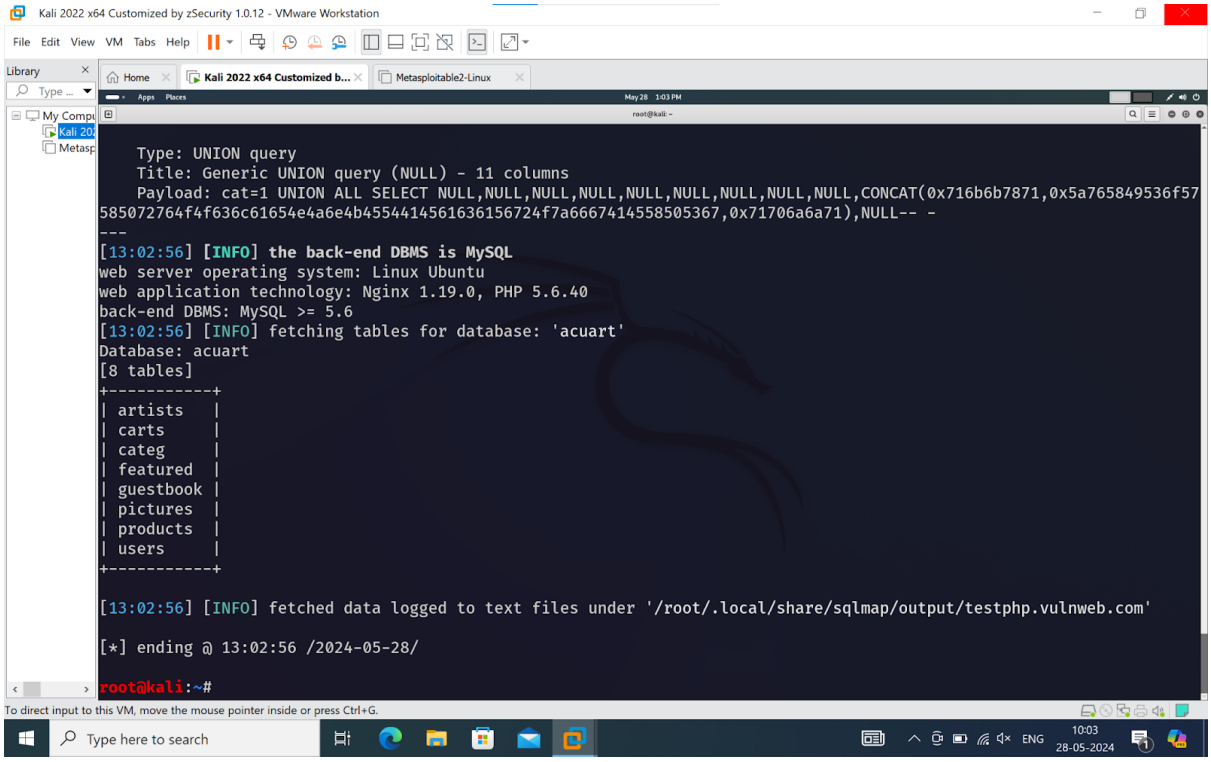


8)Get the Target's database with the help of –dbs command –

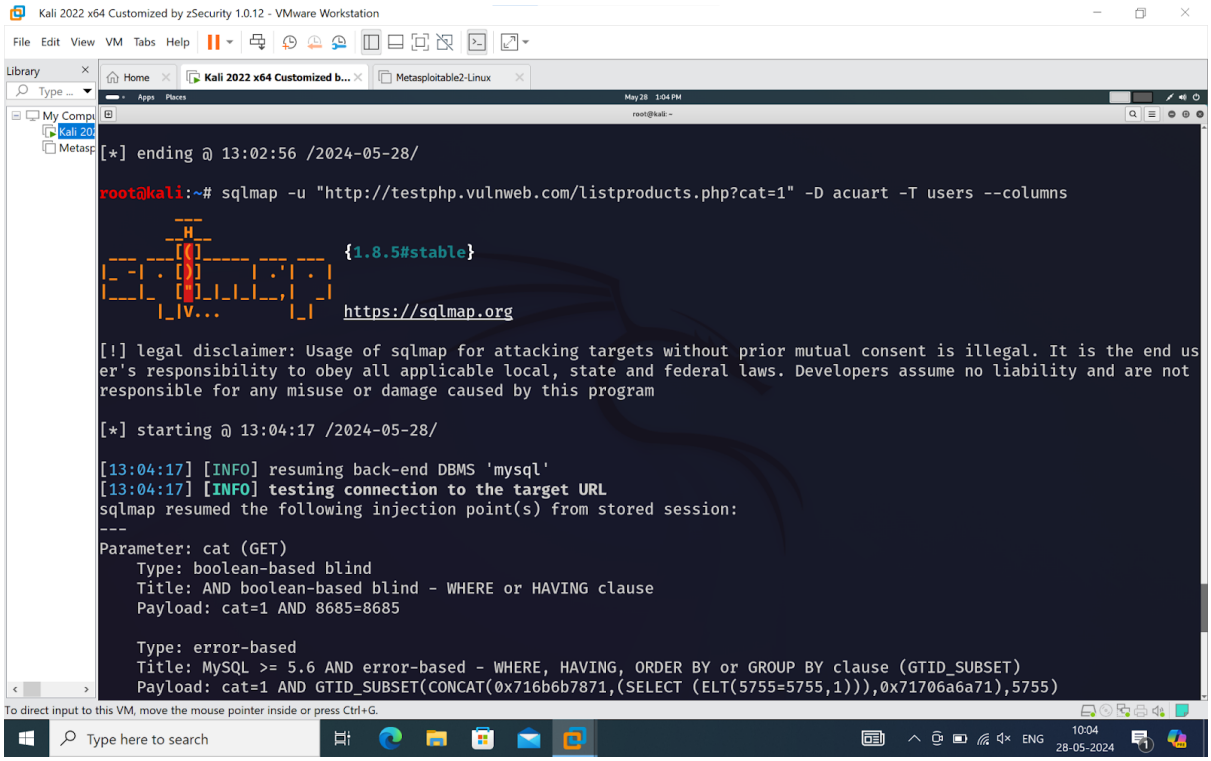


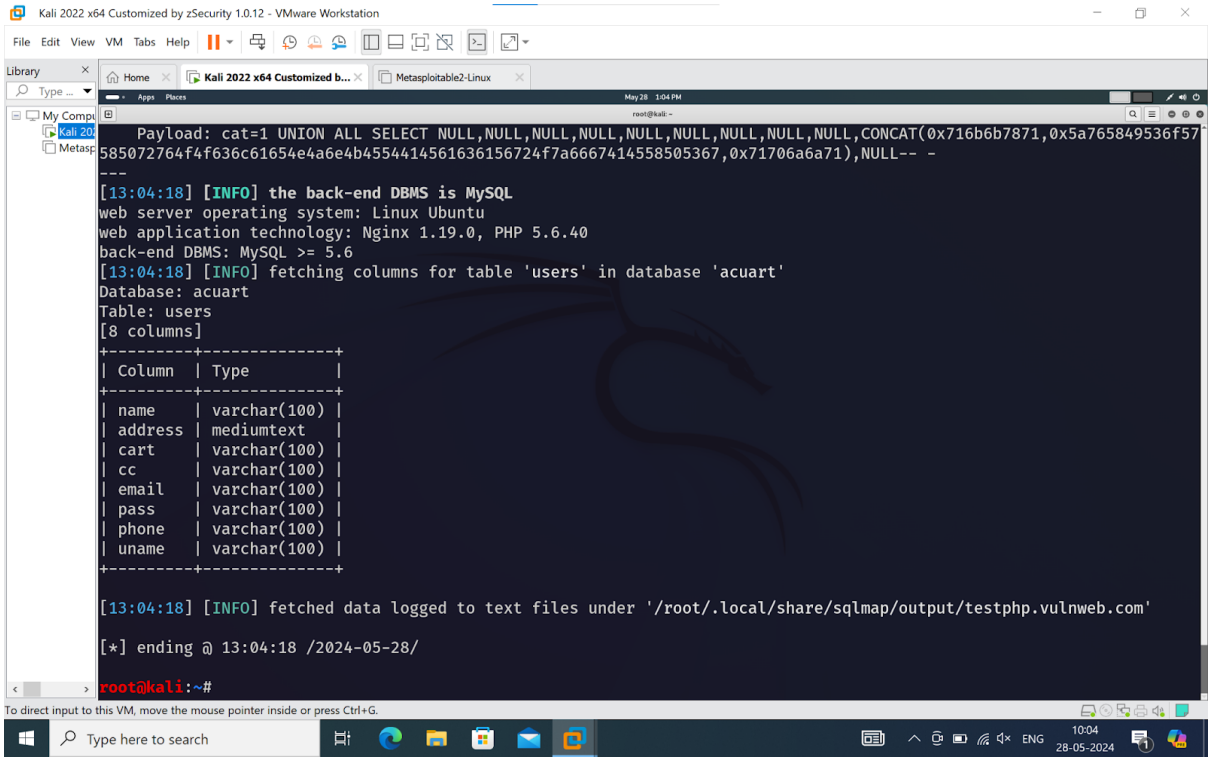
9)Get information in the form of tables-

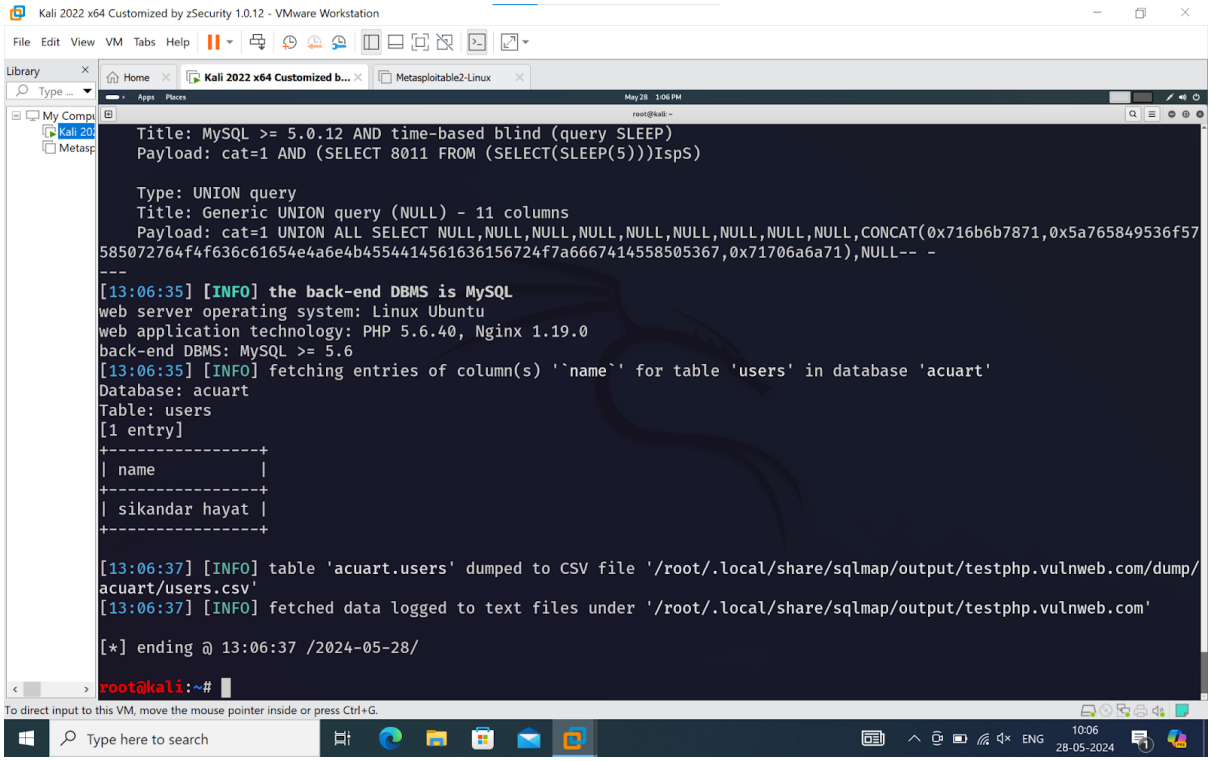




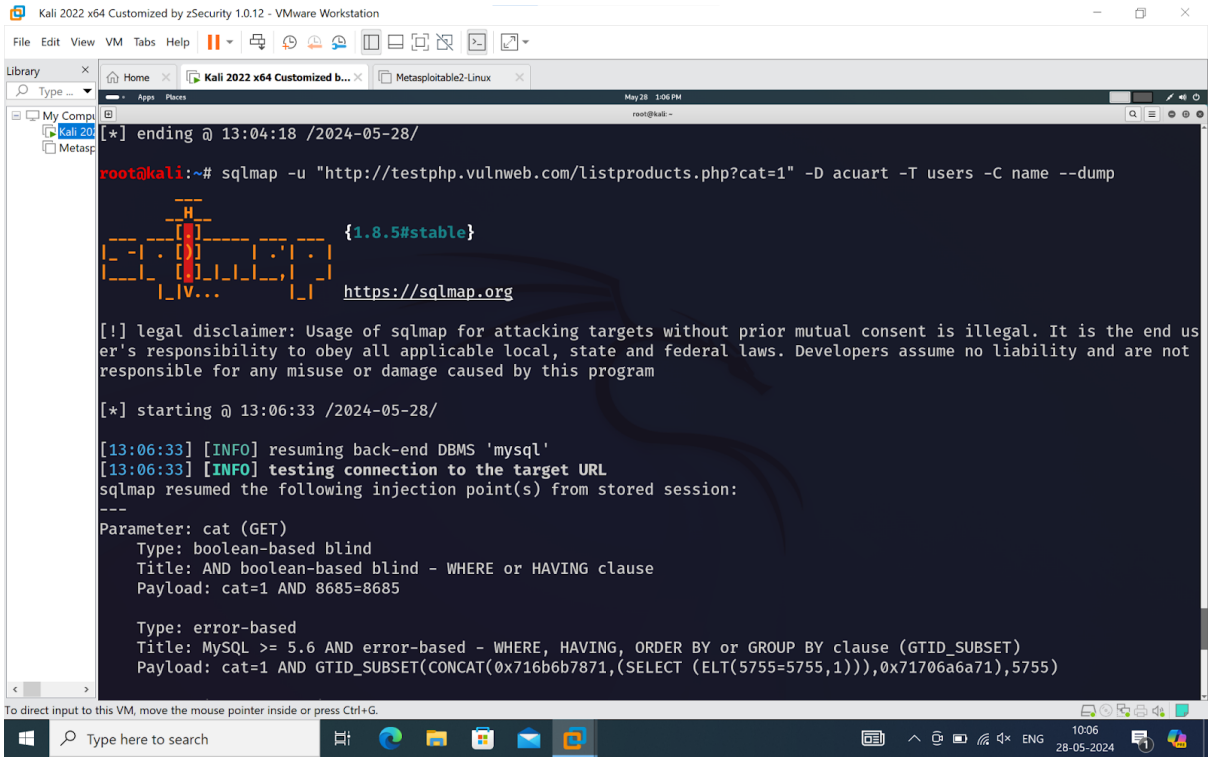
10) Search in a particular table for its columns-



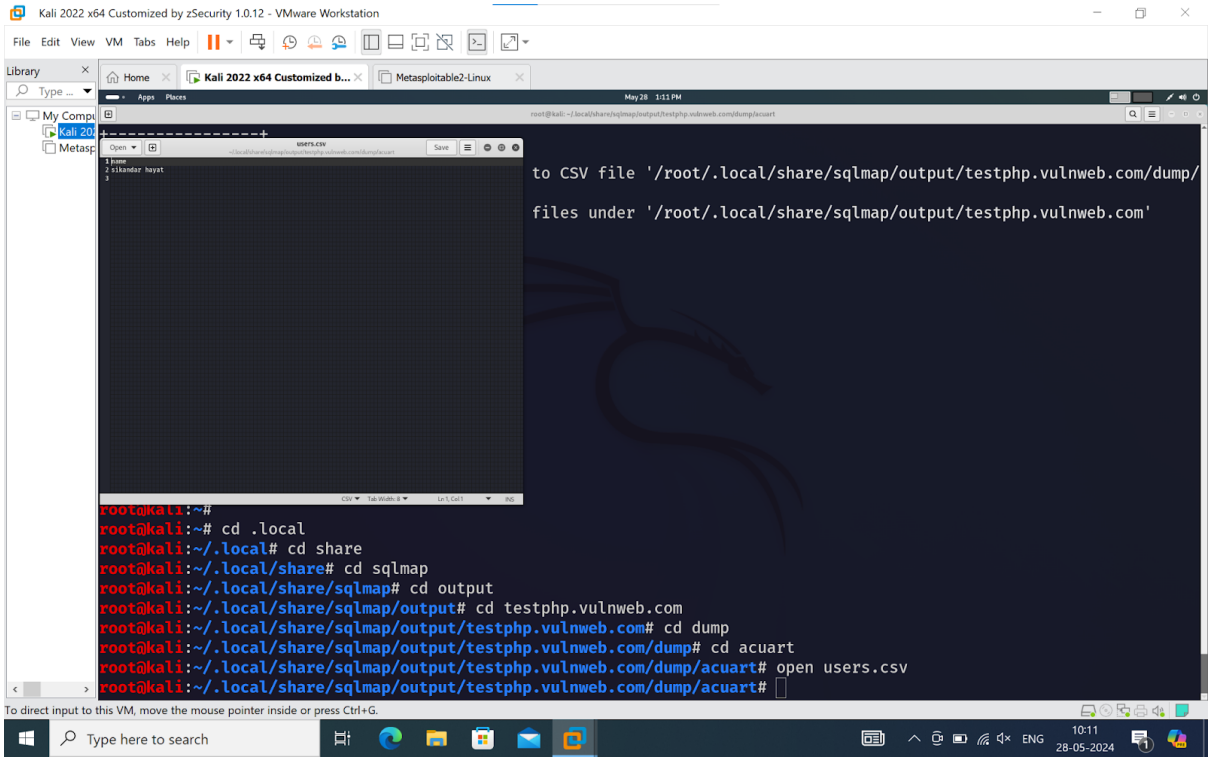




11) Save the information of a particular table and column with the help of –dump command.



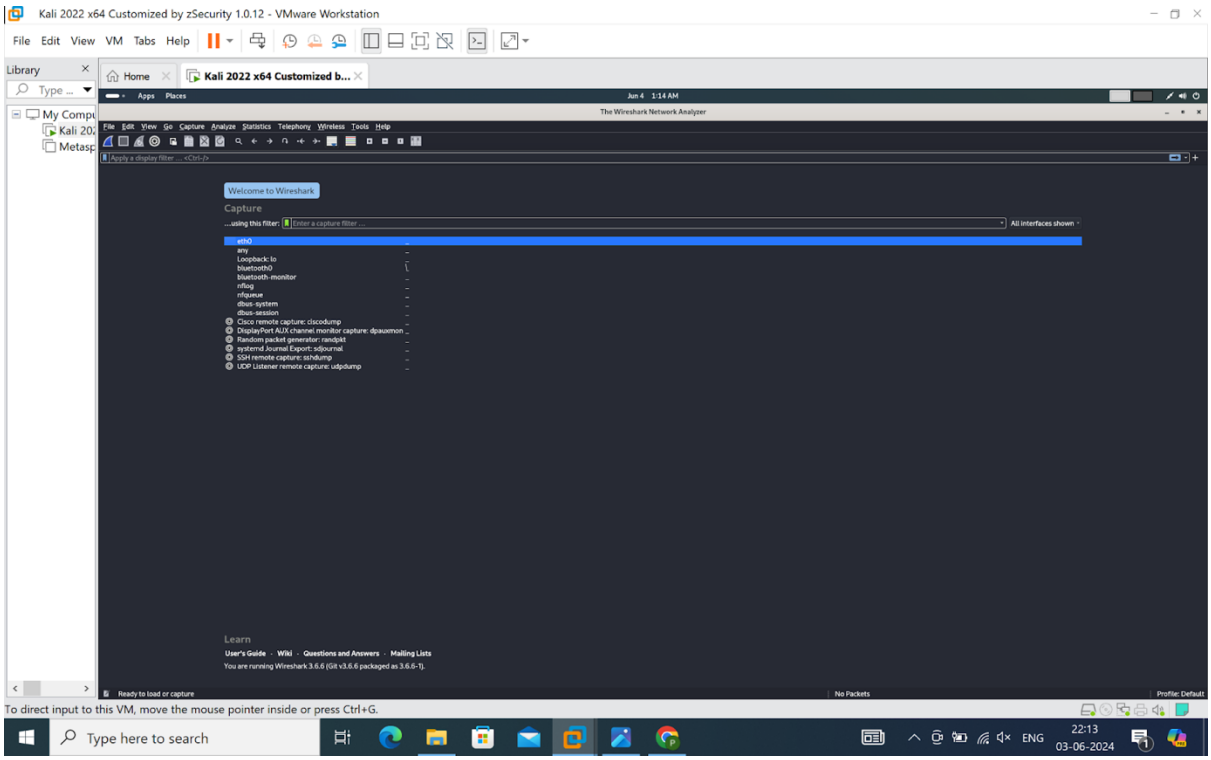
12) Access the saved file – open each directory with the command cd. At the end you can see a file with csv extension. Open it.

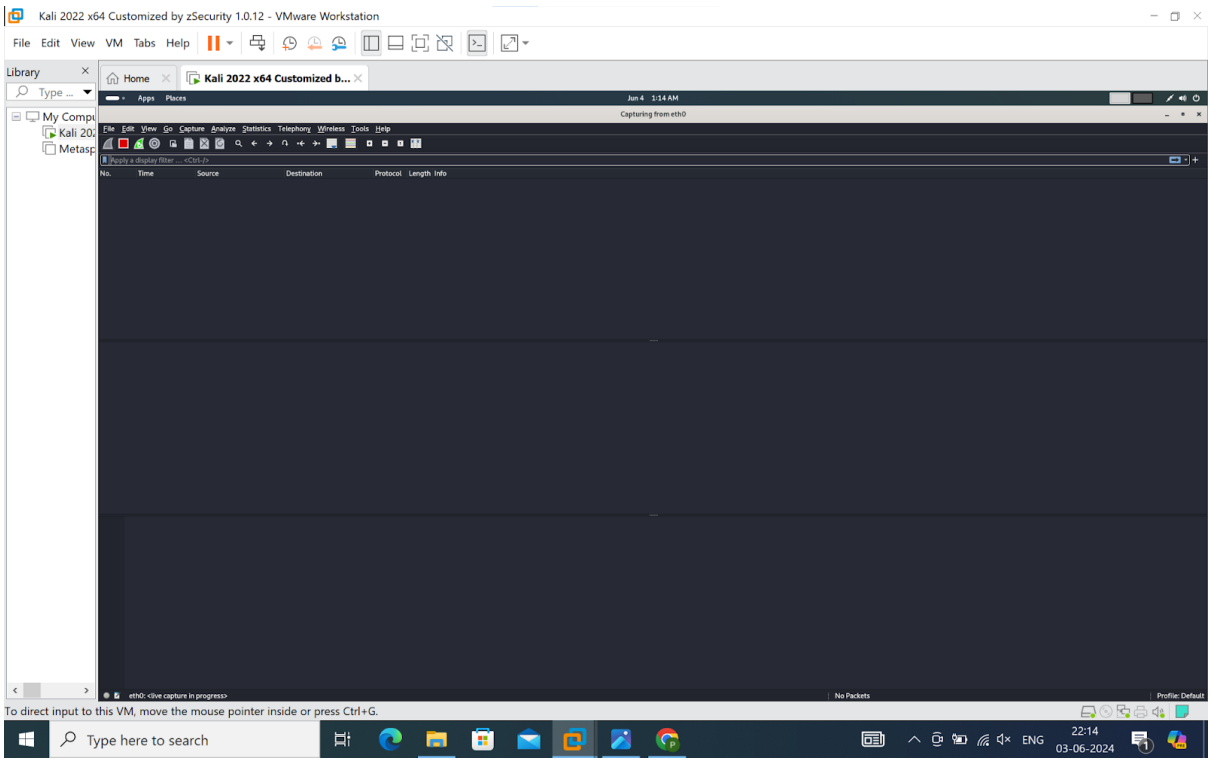


13) Thus in this way we can save all the tables and column information which can be seen and used at a later stage.

ANALYZING WIRESHARKS:

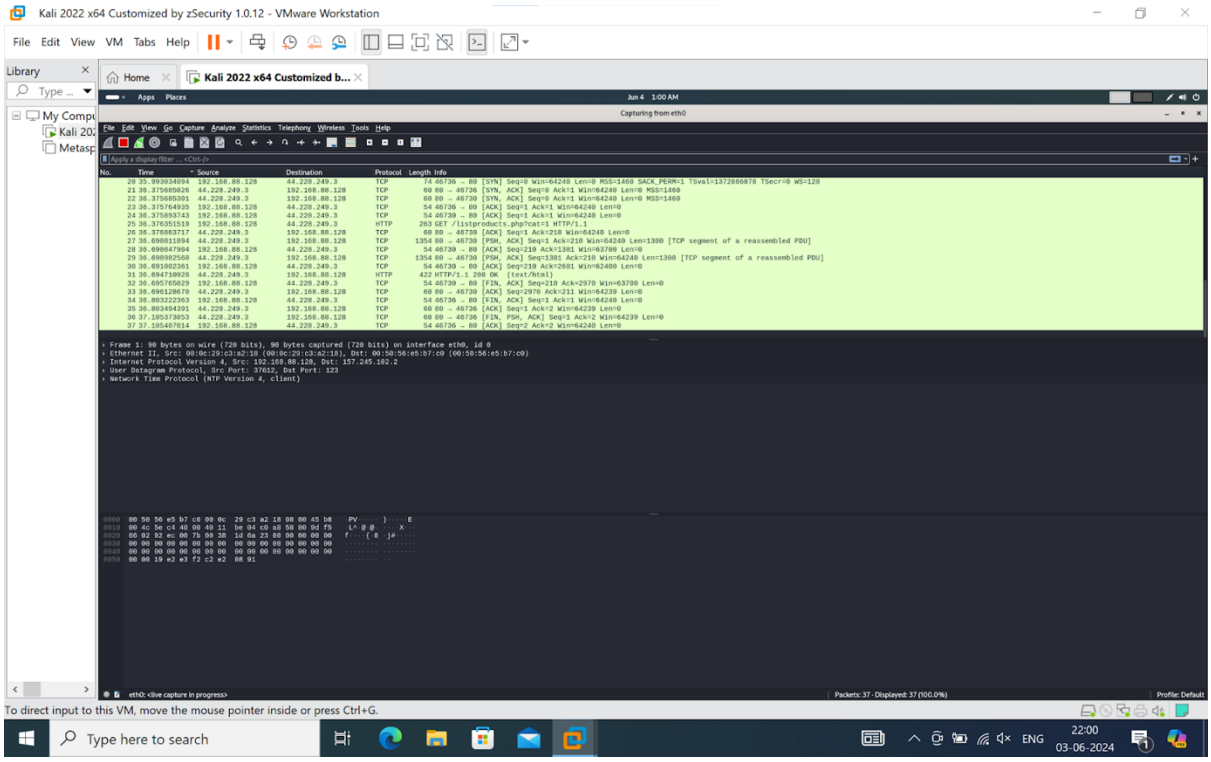
1)Open the app-

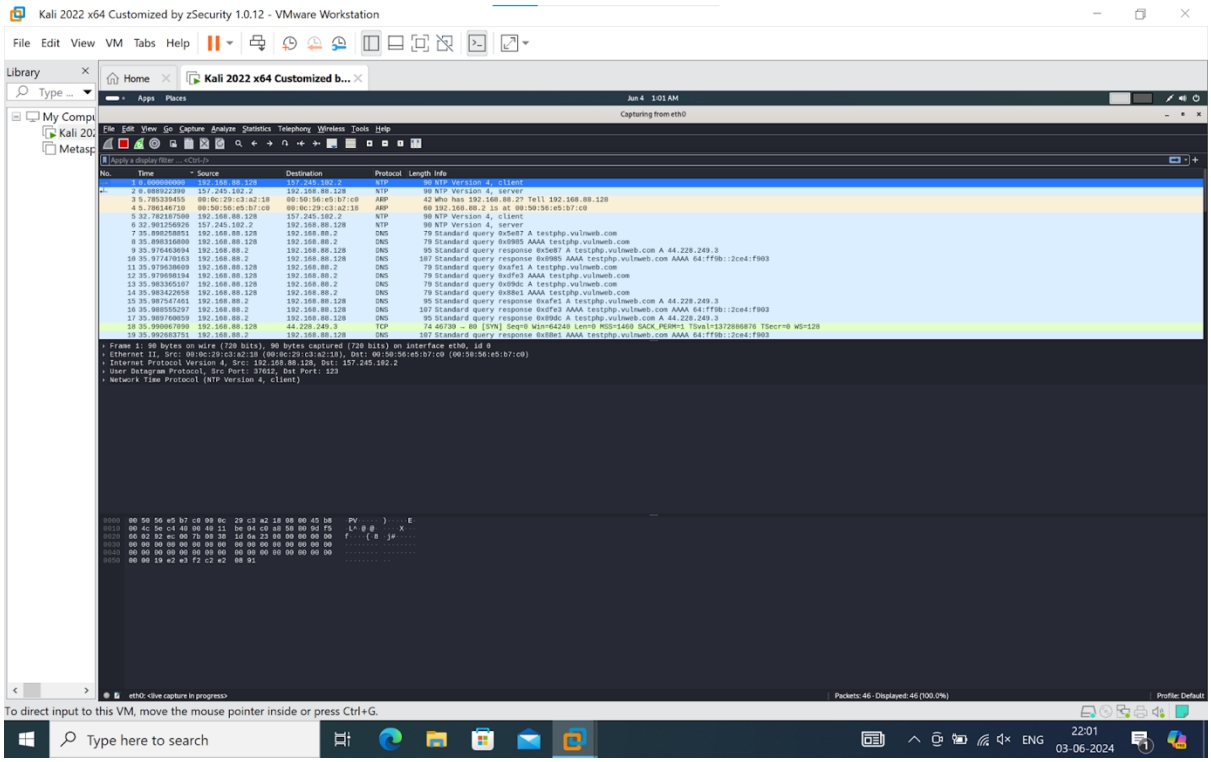




2)Go to the terminal and start SQL mapping the target.

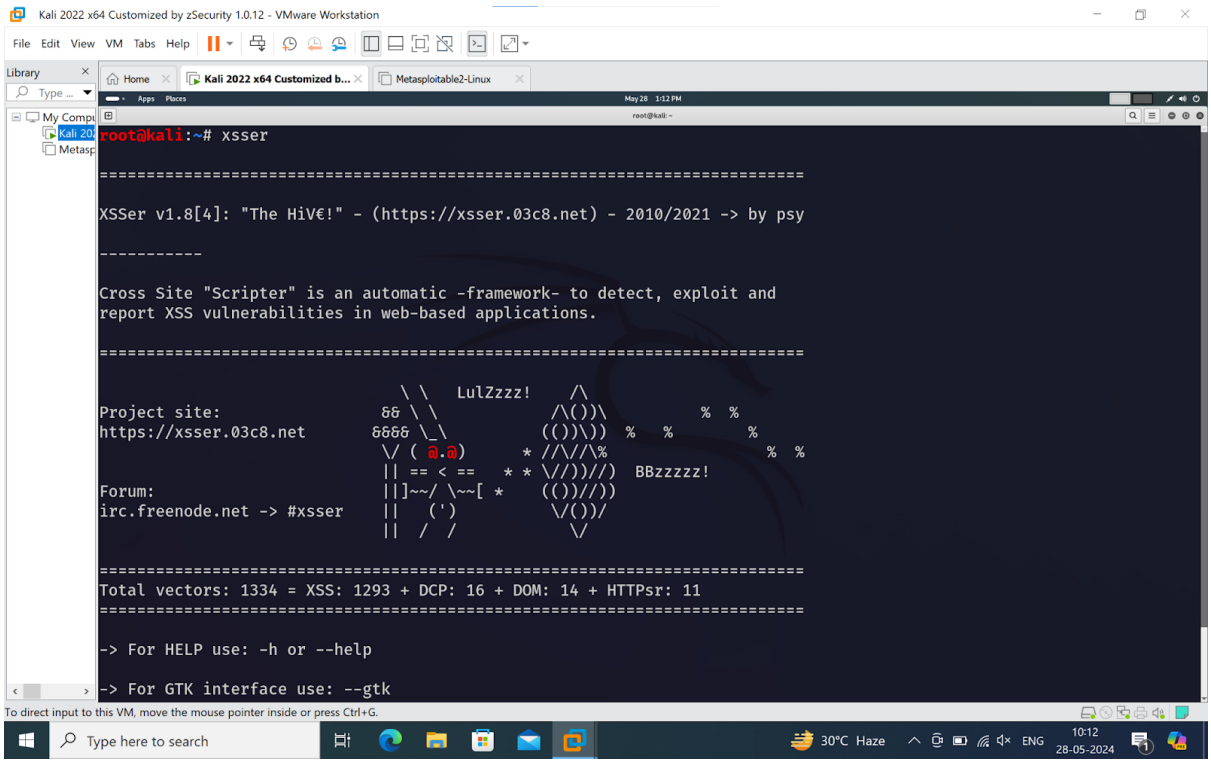
3)The packets will be caught by the wireshark



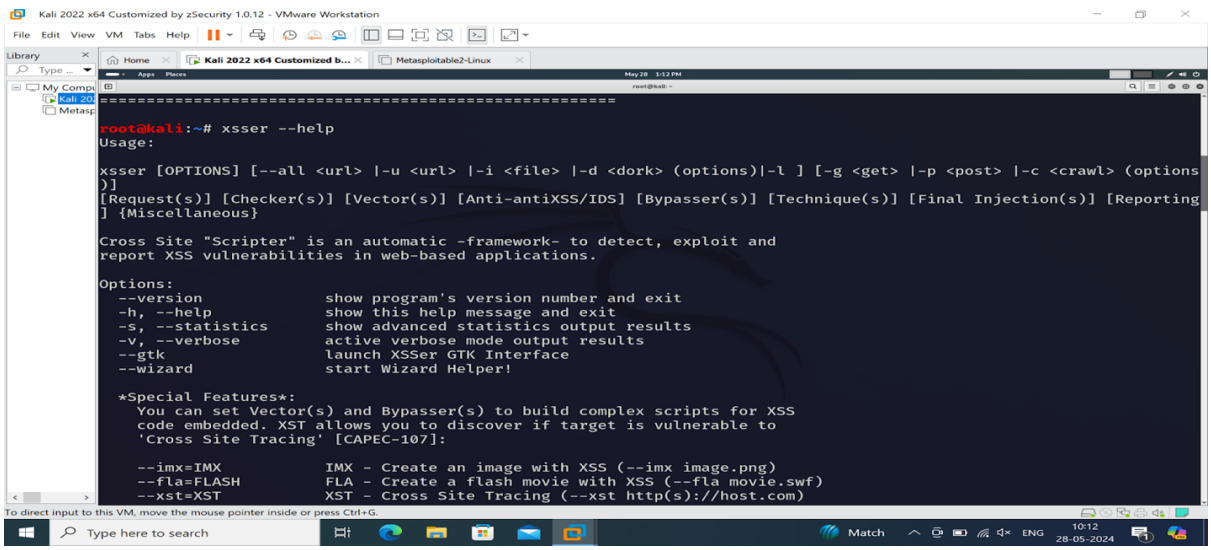


USING XSSer:

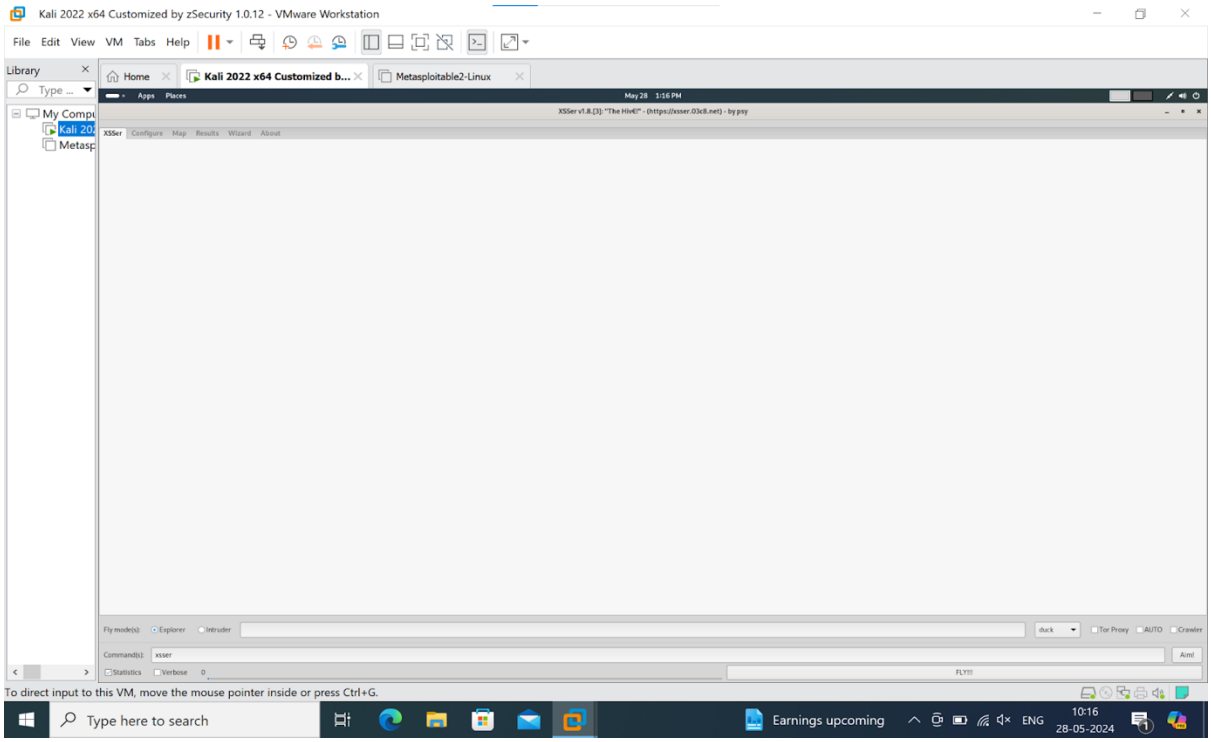
1)Install xsser in the kali linux machine and activate it-



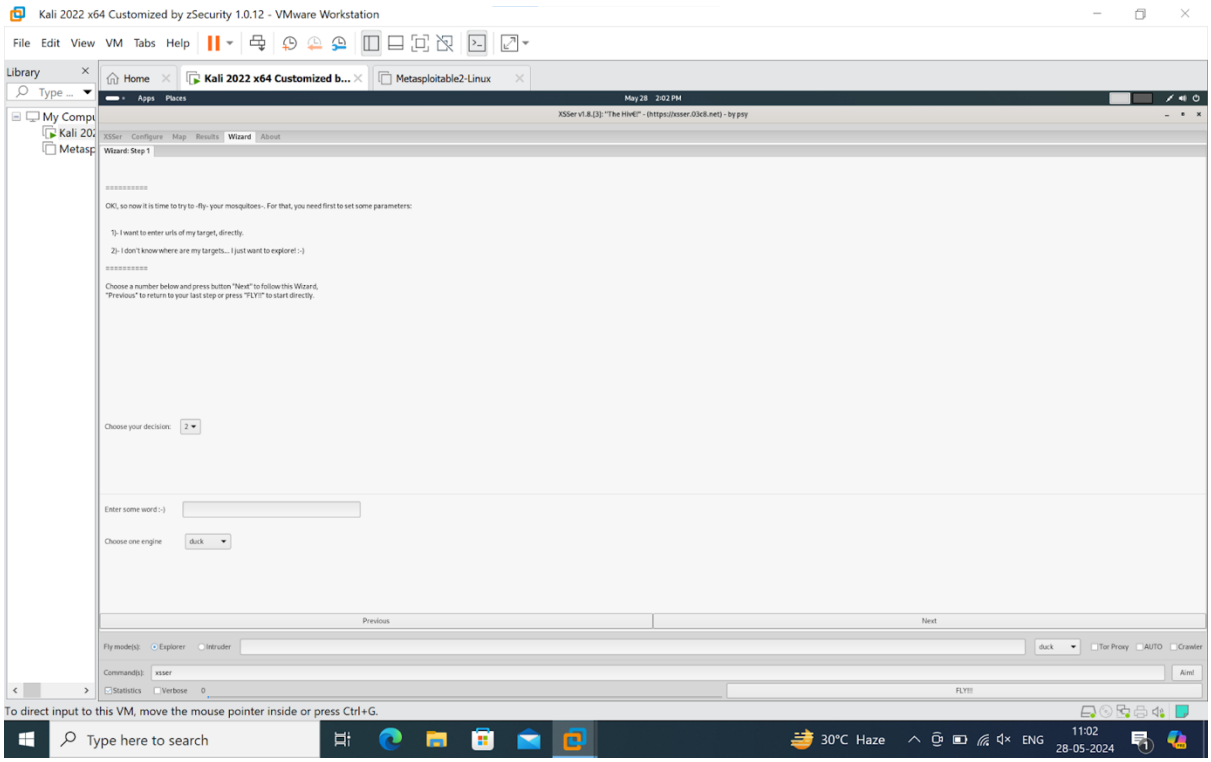
2)Open its help menu-



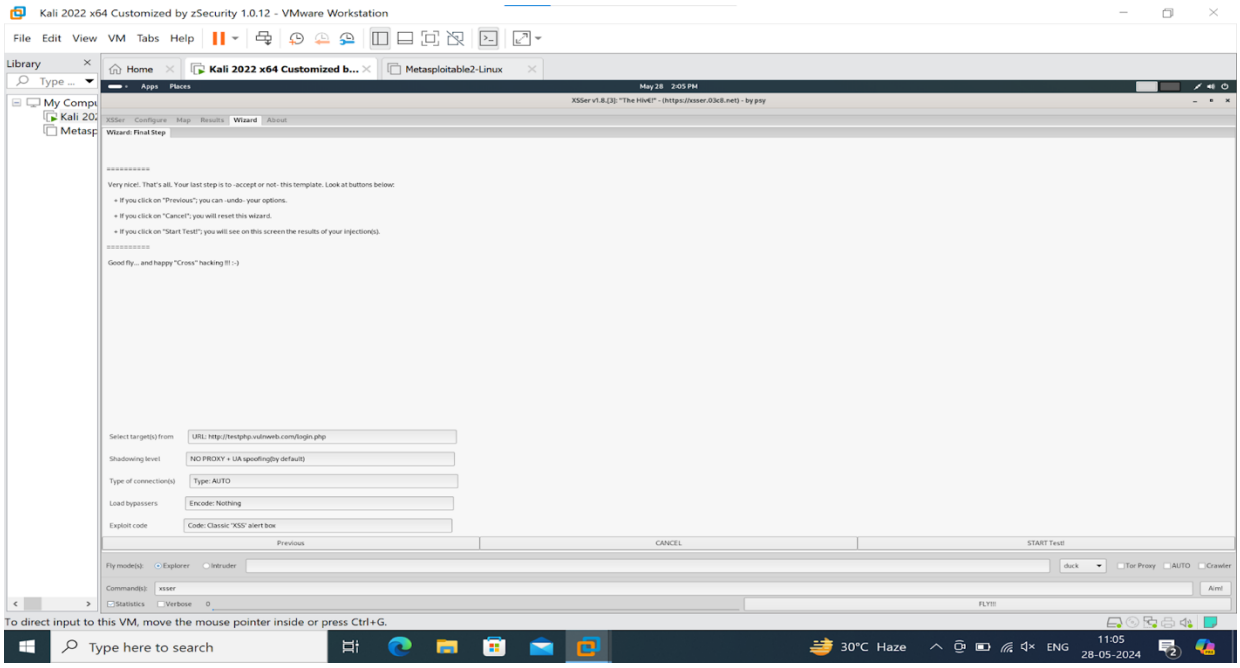
3)Open the xsser using the command –gtk :



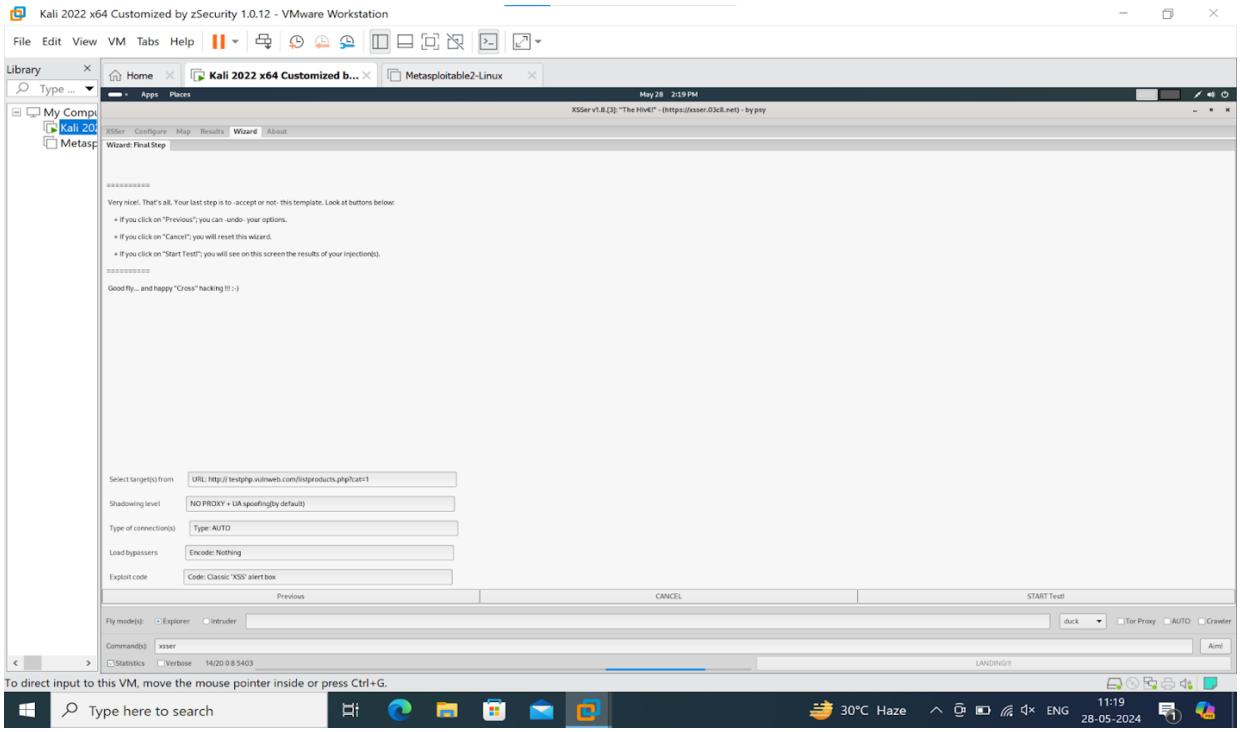
4)Open the Wizard option-



5)It will ask basic questions regarding your thoughts on site to attack, usage of proxy and techniques for the attack to scan the vulnerabilities.



6)Then press AIM and Fly at the bottom right corner of the window to start the scanning-



7)It will start the injection and give the results on the terminal.