**VISHVESHWARAIAH TECHNOLOGICAL UNIVERSITY**



**BANGALORE INSTITUTE OF TECHNOLOGY**

K.R. Road, V. V. Puram, Bengaluru-560 004

**EBTS WORKSHOP**

**CYBER SECURITY INTERNSHIP**

**PROJECT AREA:CYBER SECURITY**

**Topic: SQL Injection**

**Team name: 403 found**

**Name: ASHRITHA U**

**USN: 1BI20CS035**

**Semester: 5**

**Phone number: 9164069934**

**Mail ID:ashrithaumashankar50@gmail.com**

|  |  |  |
| --- | --- | --- |
| Project Domain | Period Covered | Period Covered |
| Ethical Hacking | 13th to 15th January, 2023 | 15th January, 2023 |

**ACKNOWLEDGEMENT**

The satisfaction and euphoria that accompanies the successful completion of any task would be incomplete without complementing those who made it possible and whose guidance and encouragement made my efforts successful. So, my sincere thanks to all those who have supported me in completing this Internship successfully.

My sincere thanks to Dr. M. U. Ashwath, Principal, BIT and Dr. Girija. J, HOD, Department of CS&E, BIT for their encouragement, support and guidance to the student community in all fields of education.

I would like to extend my heartfelt gratitude to Mr. Kamlesh Joshi, Mr. Vaishnav C V and his team for carrying out this workshop and enriching our knowledge in the field of Cyber security.

**INTRODUCTION**

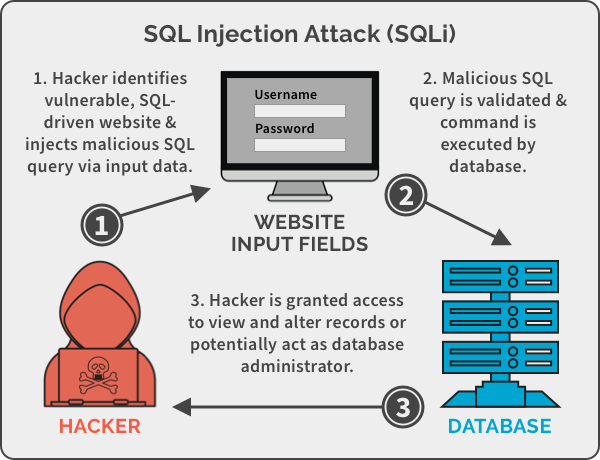
A SQL injection is a technique that attackers use to gain unauthorized access to a web application database by adding a string of malicious code to a database query.

A SQL injection (SQLi) manipulates [SQL](https://searchsqlserver.techtarget.com/definition/SQL) code to provide access to protected resources, such as sensitive data, or execute malicious SQL statements. When executed correctly, a SQL injection can expose intellectual property, customer data or the administrative credentials of a private business.

SQL injections are considered one of the most common security exploits, as evidenced by their presence on the list of [OWASP top 10 threats to web application security](https://owasp.org/www-project-top-ten/). The risk of SQLi exploits and the damage they can cause have both grown with the availability of automated tools for executing SQL injections. In the past, the likelihood of an enterprise being targeted with a SQL injection was somewhat limited because attackers had to carry out these exploits manually.

**SQL INJECTION (SQLi)**

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. A successful attack may result in the unauthorized viewing of user lists, the deletion of entire tables and, in certain cases, the [attacker](https://www.imperva.com/learn/application-security/ethical-hacking/) gaining administrative rights to a database, all of which are highly detrimental to a business.



A SQL injection (SQLi) manipulates [SQL](https://searchsqlserver.techtarget.com/definition/SQL) code to provide access to protected resources, such as sensitive data, or execute malicious SQL statements. When executed correctly, a SQL injection can expose intellectual property, customer data or the administrative credentials of a private business. SQL injection attacks can be used to [target any application](https://www.theserverside.com/feature/Application-security-vulnerabilities-are-often-known-exploits) that uses a SQL database, with websites being the most common prey. Common SQL databases include [MySQL](https://www.techtarget.com/searchoracle/definition/MySQL), [Oracle](https://www.techtarget.com/searchoracle/definition/Oracle) and [SQL Server](https://searchsqlserver.techtarget.com/definition/SQL-Server).

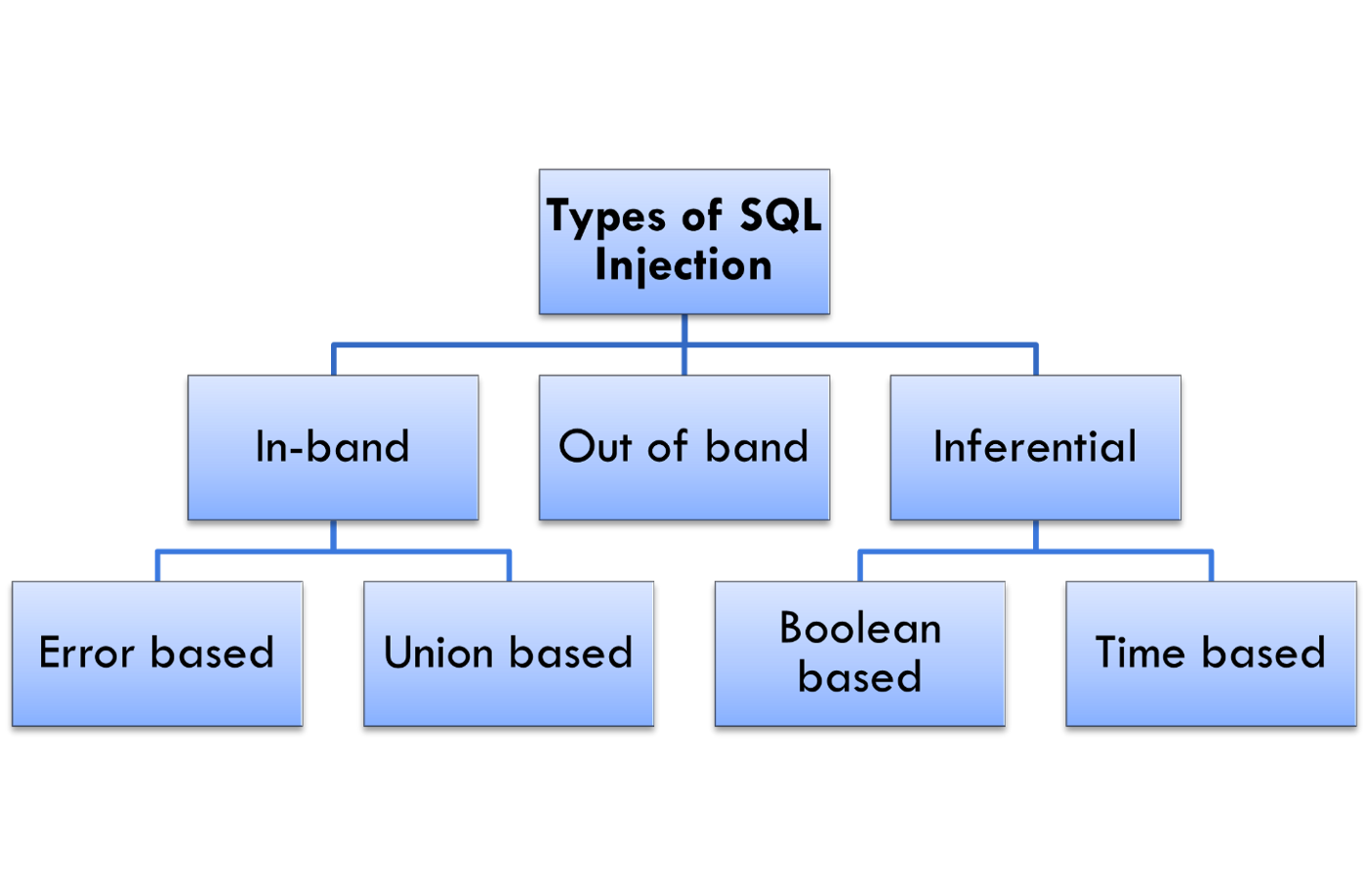
**Characteristics of an SQL injection attack**

SQL injection attacks attempt to exploit web application vulnerabilities by concatenating user input with SQL queries .SQL injection attacks can be difficult to identify because the individual steps of an attack, analyzed independently of the other steps, might be considered legitimate. Using threat detection analytics, potential SQL injection attacks can be identified by capturing the individual steps and analyzing them as part of a single complex attack.

Typical symptoms of SQL injection attacks include:

* An attacker trying to identify the structure of a dynamic SQL query, for example the number of columns queried
* An unusually large quantity of new queries, specifically queries that are uniquely or unusually structured
* Access to tables containing information about the database structure

**TYPES OF SQL INJECTION**



SQL injections typically fall under three categories: In-band SQLi (Classic), Inferential SQLi (Blind) and Out-of-band SQLi. You can classify SQL injections types based on the methods they use to access backend data and their damage potential.

**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.

### **Inferential (Blind) SQLi**

The attacker sends data payloads to the server and observes the response and behavior 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 behavioral 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.

### **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. This form of attack is primarily used as an alternative to the in-band and inferential SQLi techniques.

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.

**How to detect SQL injection vulnerabilities**

SQL injection can be detected manually by using a systematic set of tests against every entry point in the application. This typically involves:

Submitting the single quote character ' and looking for errors or other anomalies.

Submitting some SQL-specific syntax that evaluates to the base (original) value of the entry point, and to a different value, and looking for systematic differences in the resulting application responses.

Submitting Boolean conditions such as OR 1=1 and OR 1=2, and looking for differences in the application's responses.

Submitting payloads designed to trigger time delays when executed within an SQL query, and looking for differences in the time taken to respond.

Submitting OAST payloads designed to trigger an out-of-band network interaction when executed within an SQL query, and monitoring for any resulting interactions.

**Risks associated with SQL Injection**

The following are the risks associated with SQL Injection:

* **By Passing Authentication:**

It is most important to focus on By Passing Authentication during the penetration test because the attacker can access to the database just like an authorized user and he can perform his desired tasks on the data base.

* **Identifying Injectable Parameters:**

The attacker will collect the information about the structure of the back-end database of a web application and he will include the dynamic content in to the web site. This may lead the visitors to install malicious code and may redirect to the malicious site.

* **Executing Remote Commands:**

Executing these remote commands will provide attackers a tool to execute arbitrary commands on the database.For example, a remote user can execute stored database procedures and functions from a remote SQL interactive interface.

* **Denial of Service:**

The attacker can flood the server with requests so that he will the authority to stop the service to valid users, or he can delete some data.

* **Database Finger Printing:**

The attacker can determine the type of database used in backend so that he can use database-specific attacks that corresponds to weakness in a particular DBMS.

## **Consequences of a Successful SQL Injection Attack**

SQL injection attacks can have a significant negative impact on an organization. Organizations have access to sensitive company data and private customer information, and SQL injection attacks often target that confidential information. When a malicious user successfully completes an SQL injection attack, it can have any of the following impacts:

* **Exposes Sensitive Company Data**: Using SQL injection, attackers can retrieve and alter data, which risks exposing sensitive company data stored on the SQL server.
* **Compromise Users’ Privacy**: Depending on the data stored on the SQL server, an attack can expose private user data, such as credit card numbers.
* **Give an attacker administrative access to your system**: If a database user has administrative privileges, an attacker can gain access to the system using malicious code. To protect against this kind of vulnerability, create a database user with the least possible privileges.
* **Give an Attacker General Access to Your System**: If you use weak SQL commands to check user names and passwords, an attacker could gain access to your system without knowing a user’s credentials. With general access to your system, an attacker can cause additional damage accessing and manipulating sensitive information.
* **Compromise the Integrity of Your Data**: Using SQL injection, attackers can make changes to or delete information from your system.

Because the impact of a successful SQL injection attack can be severe, it’s important for businesses to practice [prevention](https://www.crowdstrike.com/blog/preventing-sophisticated-attacks-tips-from-a-real-world-incident-responder/) and limit vulnerabilities before an attack occurs.

## **How to Prevent SQL Injection Attacks?**

Preventing or mitigating SQL injection attacks is a lot about ensuring that none of the fields are vulnerable to invalid inputs and application execution. yours is manually impossible to actually to check every page and every application on the website, especially when updates are frequent and user-friendliness is the top priority.Nonetheless, security analysts and seasoned developers recommend a number of the subsequent points guarantee your database square measure well protected inside the confinement of the server.

* **Continuous Scanning and Penetration Testing:** The automated [web application scanner](https://www.indusface.com/web-application-scanning.php/)has been the best choice to point out vulnerabilities within the web applications for quite some time now. Now, with SQL injections getting smarter in exploiting logical flaws, website security professionals should explore manual testing with the help of a security vendor .They can authenticate user inputs against a set of rules for syntax, type, and length. It helps to audit application vulnerabilities discreetly so that you can patch the code before hackers exploit it to their advantage.

### **Restrict Privileges:** It is more of a database management function, but enforcing specific privileges to specific accounts helps prevent blind SQL injection attacks. Begin with no privileges account and move on to ‘read-only’, ‘edit’, ‘delete’ and similar privilege levels. Minimizing privileges to the application will ensure that the attacker, who gets into the database through the application, cannot make unauthorized use of specific data.

### **Use Query Parameters:** Dynamic queries create a lot of troubles for security professionals. They have to deal with variable vulnerabilities in each application, which only gets graver with updates and changes. It is recommended that you prepare parameterized queries. These queries are simple, easy to write, and only pass when each parameter in SQL code is clearly defined. This way, your info is supplied with weapons to differentiate between code and information inputs.

### **Instant Protection:** A majority of organizations fail the problems like outdated code, scarcity of resources to test and make changes, no knowledge of application security, and frequent updates in the application. For these, web application protection is the best solution. A managed [**web application firewall**](https://www.indusface.com/web-application-firewall.php) can be deployed for immediate mitigation of such attacks. It contains custom policies to block any suspicious input and deny information breach instantly. This way, you do not have to manually look for loopholes and mend problems afterward.

**SQL INJECTION USING SQLMAP TOOL:**

**SQLMAP:**

sqlmap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. It comes with a powerful detection engine, many niche features for the ultimate penetration tester and a broad range of switches lasting from database fingerprinting, over data fetching from the database, to accessing the underlying file system and executing commands on the operating system via out-of-band connections.

**FEATURES OF SQLMAP:**

* Full support for six SQL injection techniques: boolean-based blind, time-based blind, error-based, UNION query-based, stacked queries and out-of-band.
* Support to directly connect to the database without passing via a SQL injection, by providing DBMS credentials, IP address, port and database name.
* Support to enumerate users, password hashes, privileges, roles, databases, tables and columns.
* Automatic recognition of password hash formats and support for cracking them using a dictionary-based attack.
* Support to dump database tables entirely, a range of entries or specific columns as per user's choice. The user can also choose to dump only a range of characters from each column's entry.
* Support to search for specific database names, specific tables across all databases or specific columns across all databases' tables. This is useful, for instance, to identify tables containing custom application credentials where relevant columns' names contain string like name and pass.
* Support to download and upload any file from the database server underlying file system when the database software is MySQL, PostgreSQL or Microsoft SQL Server.
* Support to execute arbitrary commands and retrieve their standard output on the database server underlying operating system when the database software is MySQL, PostgreSQL or Microsoft SQL Server.

**Sqlmap Options**

**Mandatory Arguments**

At least one of the following is necessary for the sqlmap command to run:

* **-h:**  Basic help
* **-hh:**  Advanced help
* **--version:** Shows sqlmap version number
* **-v VERBOSE:** Set verbosity level where VERBOSE is an integer between 0 and 6 inclusive (default: 1)
* **--wizard:** Simple wizard interface for beginner users
* **--shell:** Prompt for an interactive sqlmap shell; inside the shell, omit sqlmap and enter options and arguments directly
* **--update:** Update sqlmap to the latest version --purge Safely remove all content from sqlmap data directory
* **--list-tampers:** Display list of available tamper scripts
* **--dependencies:** Check for missing (optional) sqlmap dependencies

**General Options**

Set general working parameters.

* --batch: Never ask for user input, use the default behaviour
* --answers: Set predefined answers: parameters are substring(s) of question prompt(s); join multiple answers with a comma. You may use this with
* --batch.: Usage: --answers="quit=N, follow=N"
* --flush-session: Flush session files for current target
* --crawl=CRAWL\_DEPTH: Crawl (collect links of) the website starting from the target URL –crawl-exclude=CRAWL\_EXCLUDE Regular expression to exclude pages from being crawled (e.g. --crawl-exclude="logout" to skip all pages containing the keyword “logout”)
* --csv-del=CSVDEL: Delimiting character used in CSV output (default “,”)
* --charset=CHARSET: Blind SQLi charset (e.g., "0123456789abcdef")
* --dump-format=DUMP\_FORMAT: Format of dumped data (CSV (default), HTML or SQLITE)
* --encoding=ENCODING: Character encoding used for data retrieval (e.g., GBK)
* --flush-session: Flush session files for current target
* --output-dir=OUTPUT\_DIR: Custom output directory path
* --parse-errors: Parse and display DBMS error messages from responses
* --save=SAVECONFIG: Save options to a configuration INI file
* --skip-waf: Skip heuristic detection of WAF/IPS protection
* --web-root=WEBROOT: Web server document root directory (e.g. "/var/www")

**Request Options**

* Specify how to connect to the target URL.
* --data=DATA: Data string to be sent through POST (e.g. "id=1")
* --cookie=COOKIE: HTTP Cookie header value (e.g. "PHPSESSID=77uT7KkibWPPEkSPjBd9GJjPLGj; security=low")
* --random-agent : Use randomly selected HTTP User-Agent header value
* --proxy=PROXY: Use a proxy to connect to the target URL
* -tor: Use Tor anonymity network
* --check-tor: Check to see if Tor is used properly

**DEMONSTATION:**

As an example, we will make use of a website that is designed with vulnerabilities for demonstration purposes: http://testphp.vulnweb.com/listproducts.php?cat=1

As you can see, there is a GET request parameter (cat=1) that can be changed by the user by modifying the value of cat. So, this website might be vulnerable to SQL injection of this kind. To test for this, we use SQLMAP.

Here we are trying to find the username and password of the artist r4wB173

$ sqlmap –h : To look at the set of parameters that can be passed with sqlmap

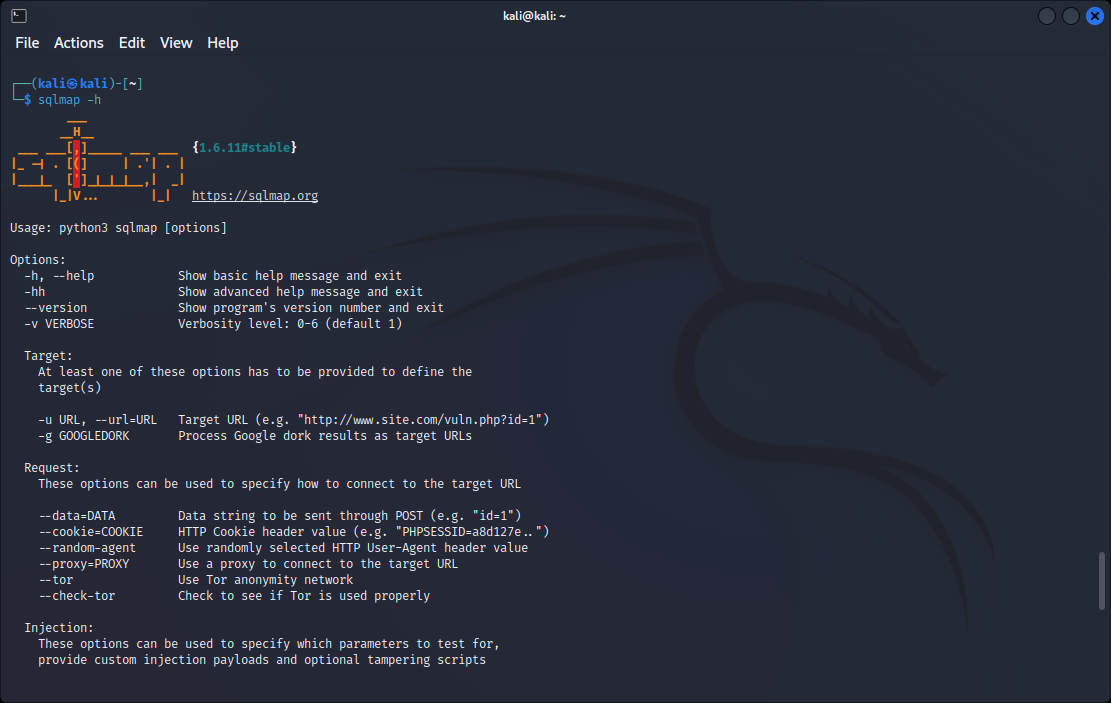


Fig: running help command in kali linux terminal

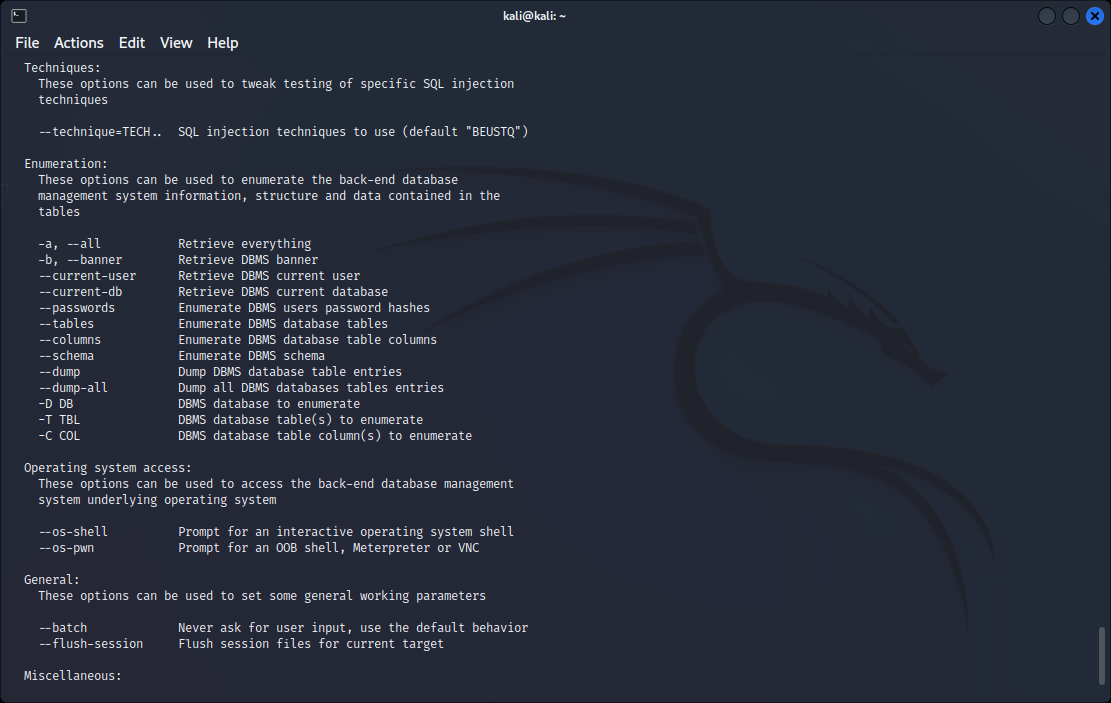


Fig: displaying options that can be used to enumerate databases

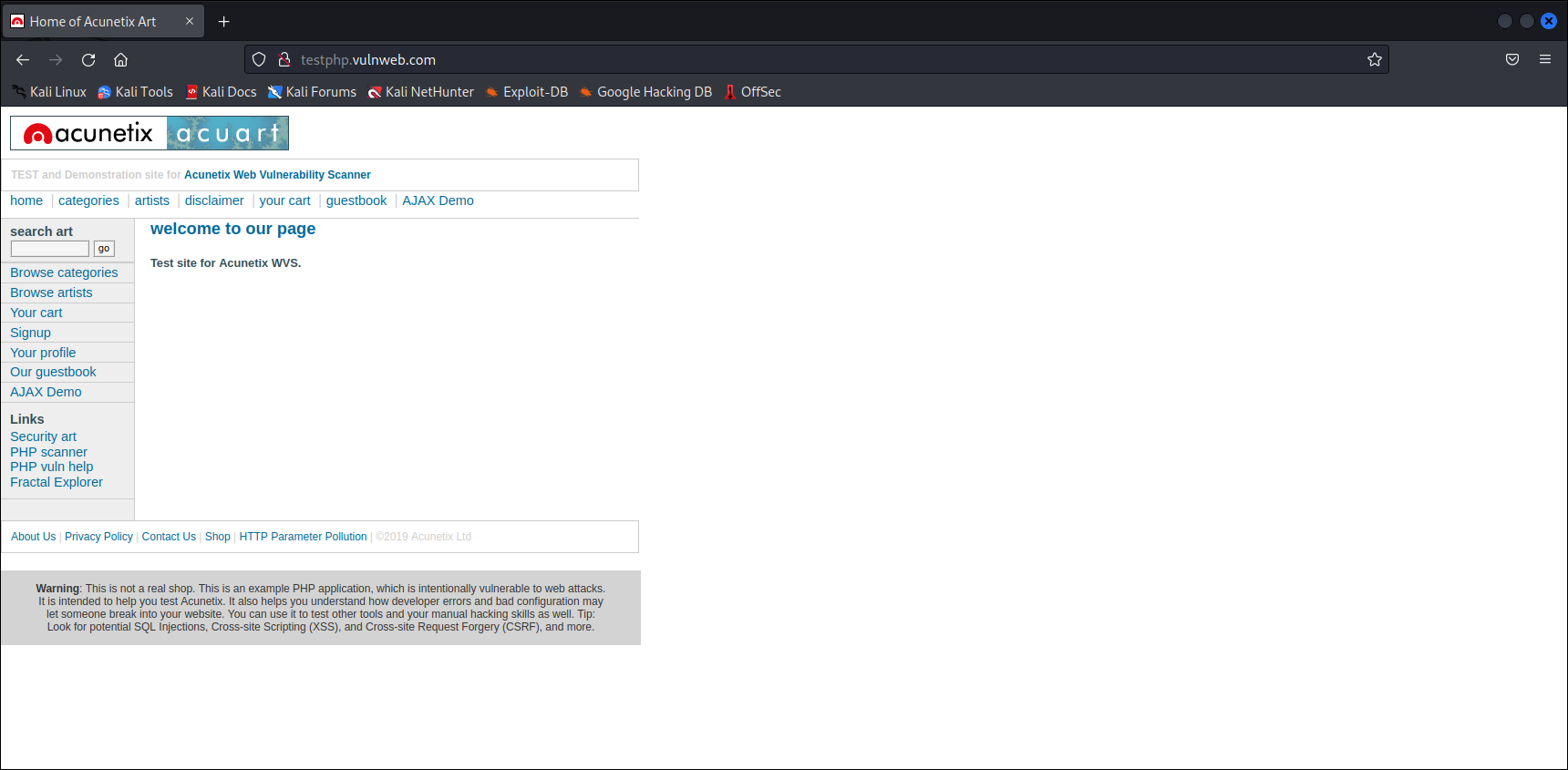
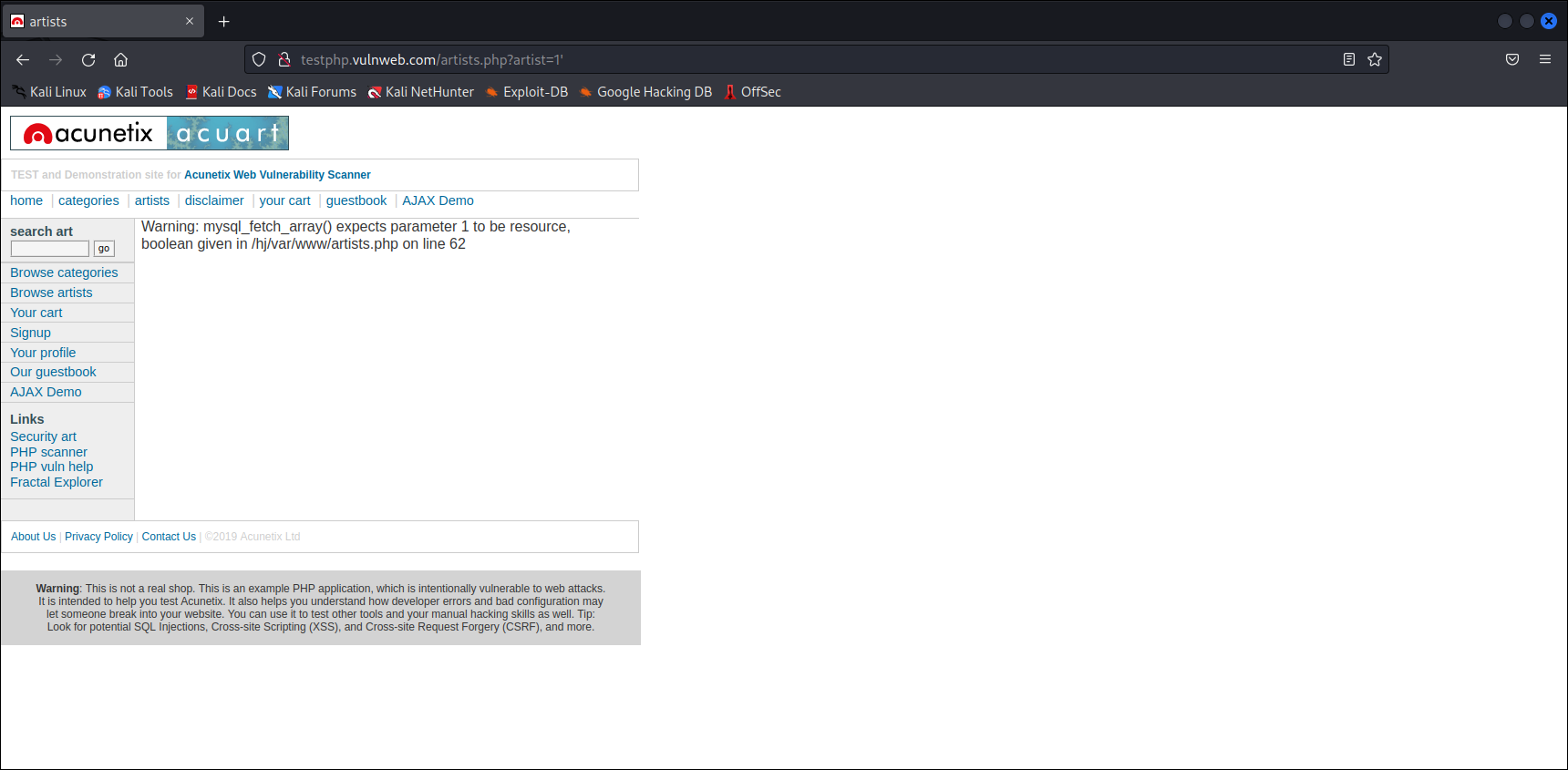


Fig: website used for sql injection

Adding “ **‘** “ to this website <http://testphp.vulnweb.com/artists.php?artist=1> gives a warning, which is an indication that we can inject this website.

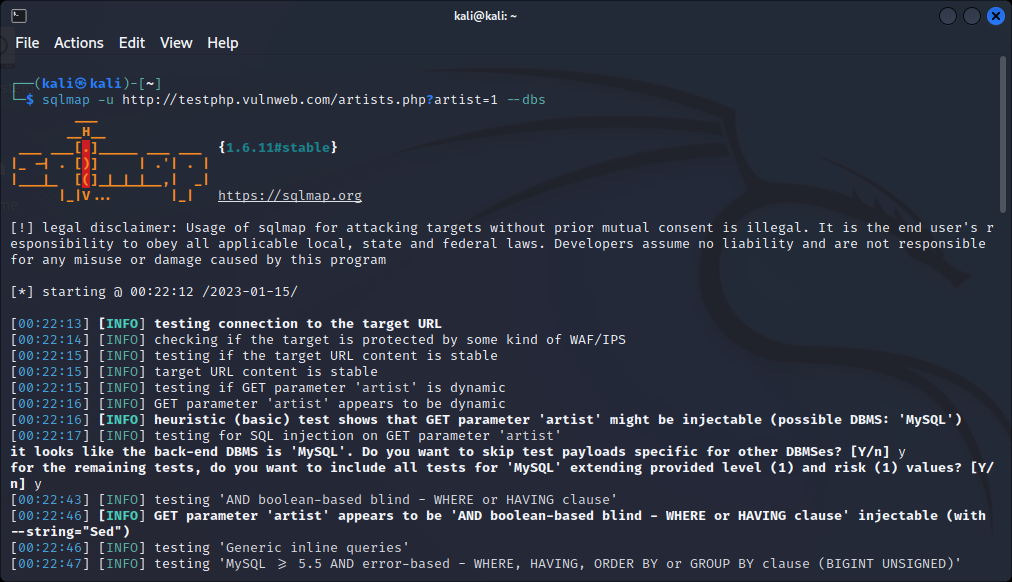


$ sqlmap –u enables us to add the target url <http://testphp.vulnweb.com/artists.php?artist=1>

**$ sqlmap –u** [**http://testphp.vulnweb.com/artists.php?artist=1**](http://testphp.vulnweb.com/artists.php?artist=1) **--dbs**

Tests the connection to target link and lists all the databases present.

--dbs: to find the databases present



Back-end DBMS is MySQL

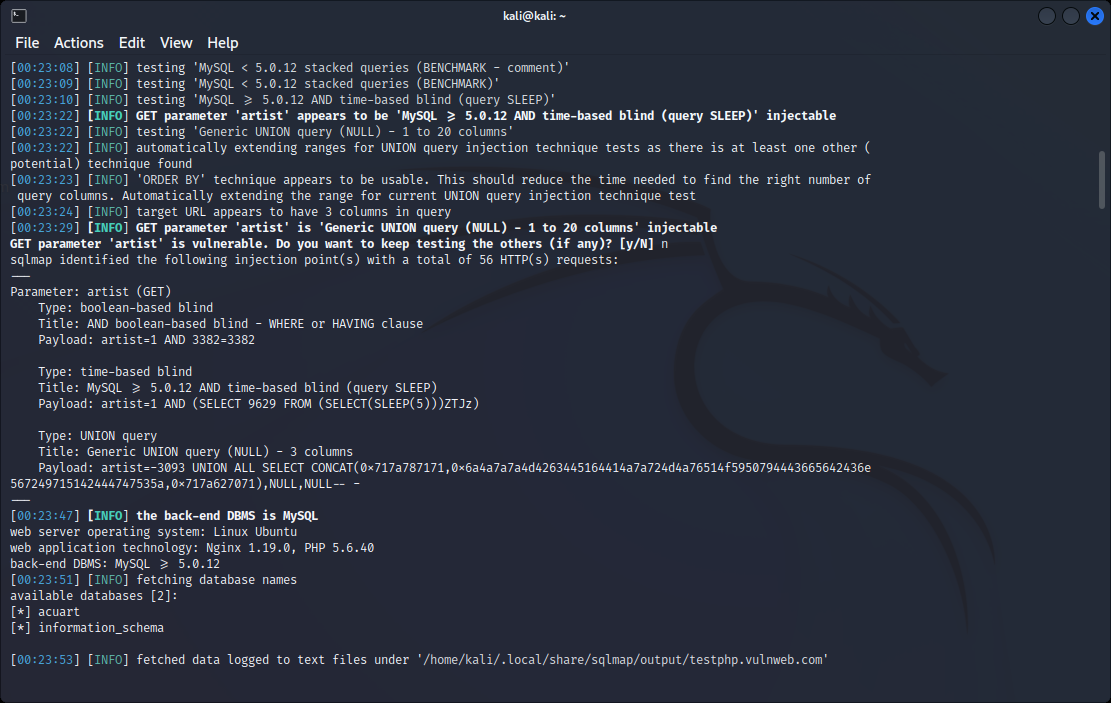


Fig: retrieving existing data and vulnerabilities of the website

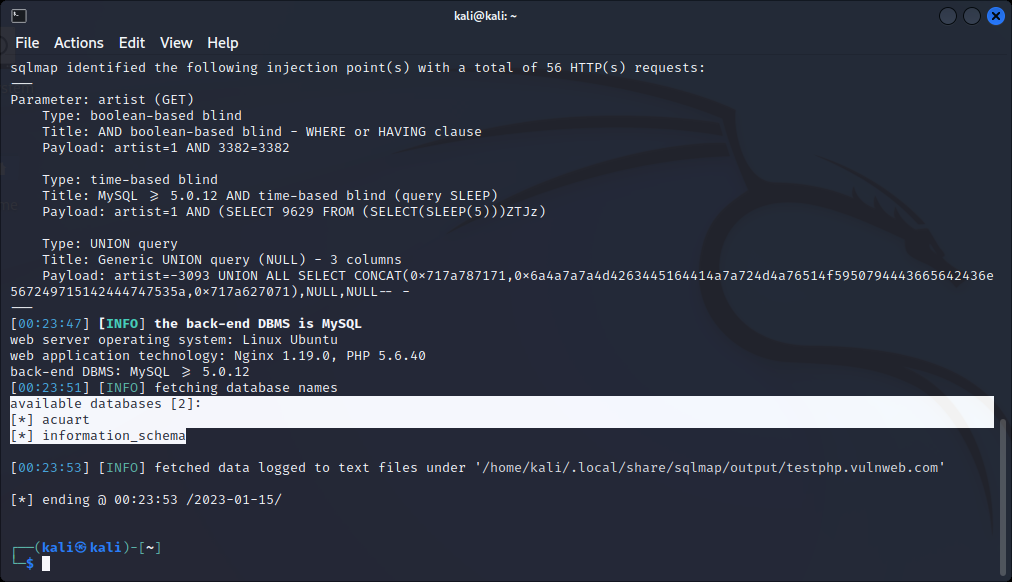


Fig: shows the available databases where injection is possible

**$ sqlmap –u** [**http://testphp.vulnweb.com/artists.php?artist=1**](http://testphp.vulnweb.com/artists.php?artist=1) **–D acuart --tables**

-D is to select the database and –tables do list tables in that database

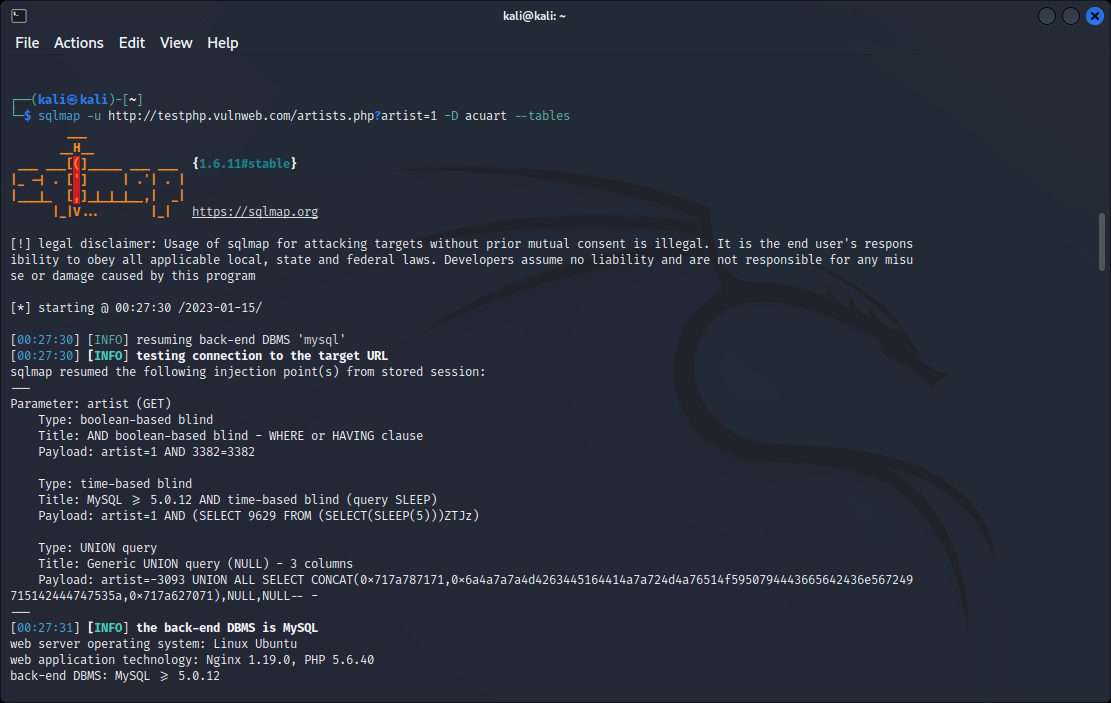


Fig: finding out tables present in the database acuart

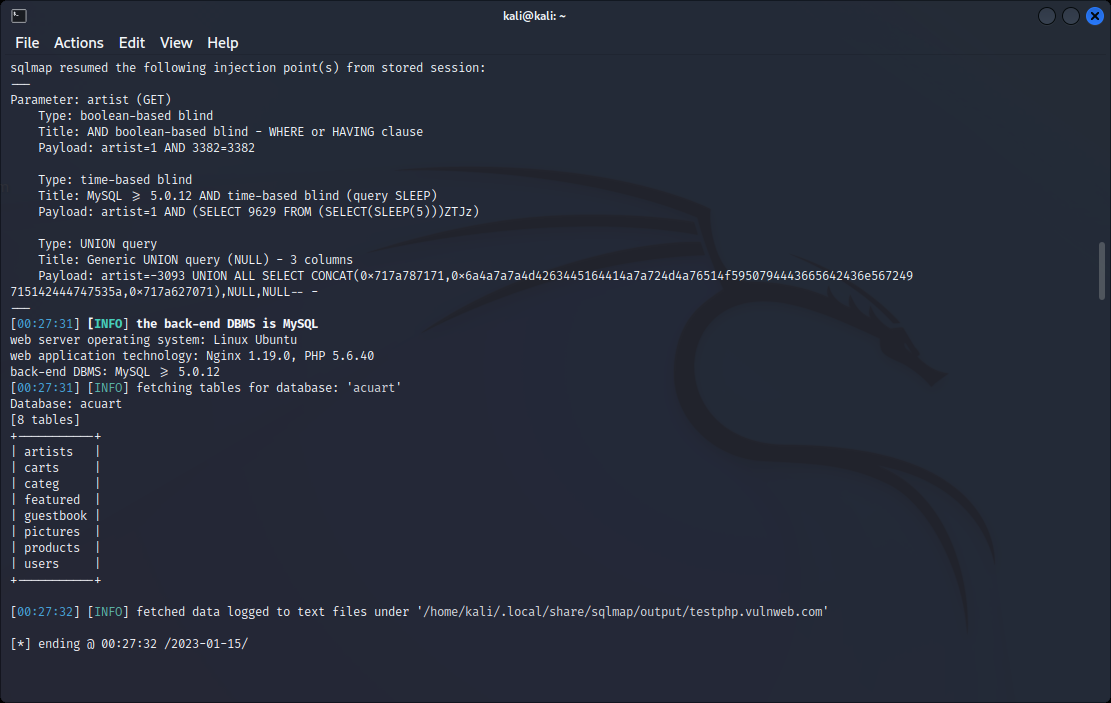
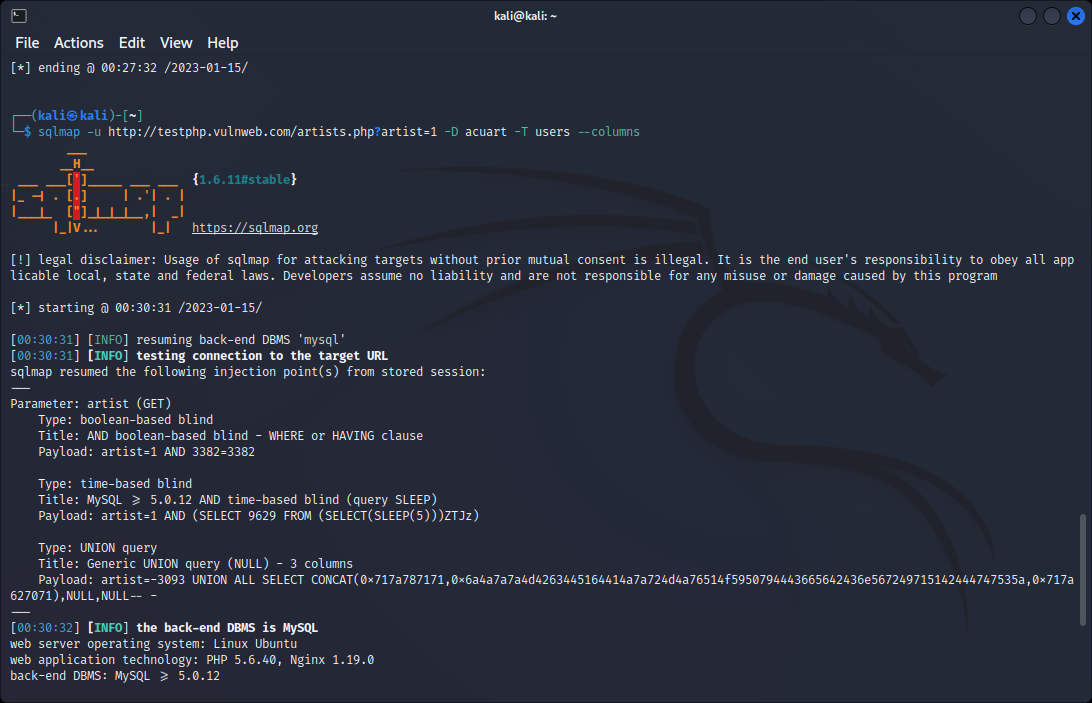


Fig: displaying all the tables present in the database acuart

**$ sqlmap –u** [**http://testphp.vulnweb.com/artists.php?artist=1**](http://testphp.vulnweb.com/artists.php?artist=1) **–D acuart –T users –columns**

-T: indicating the table

--columns: to display all the columns in the table user



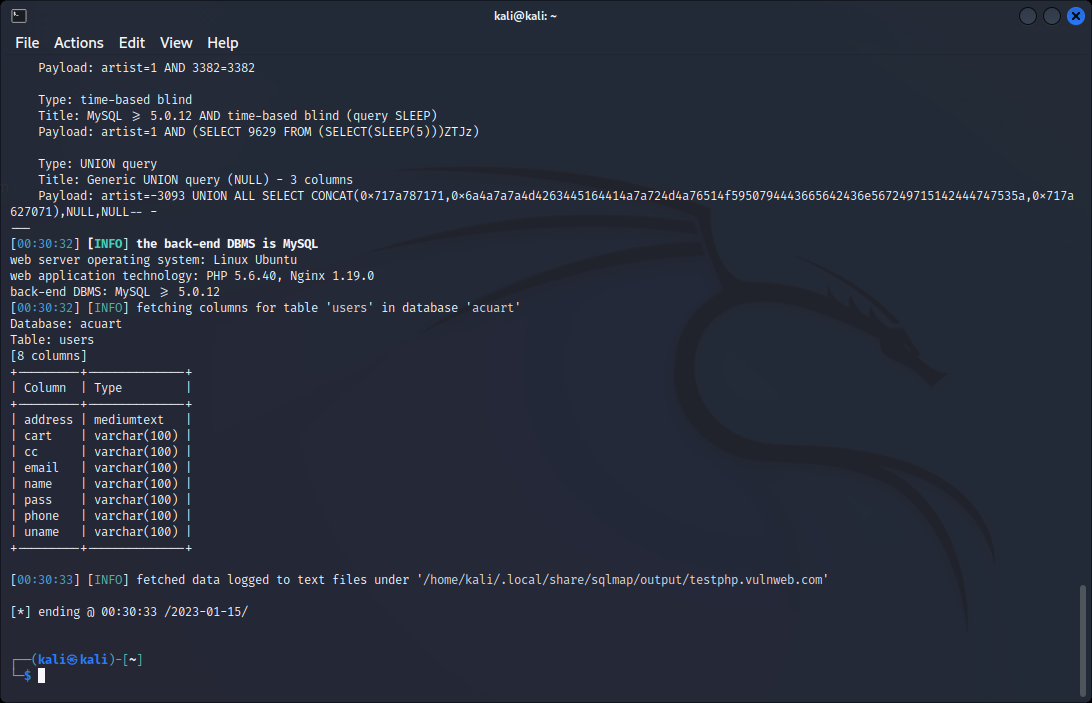
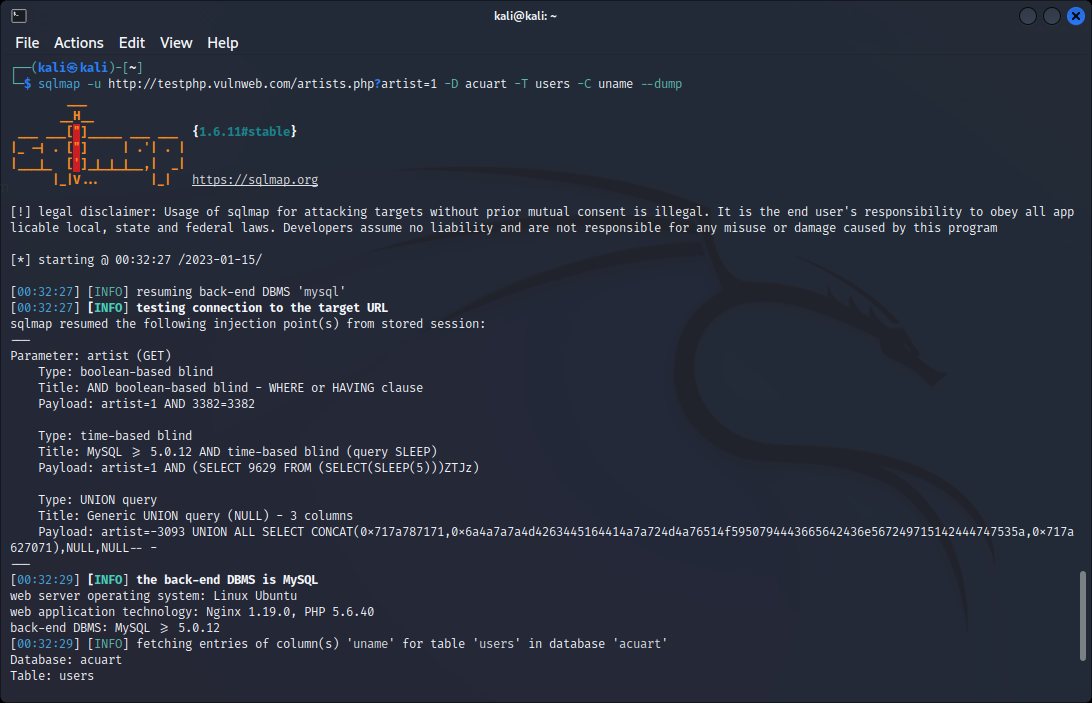


Fig: displaying all the columns in the table users in the database acuart

**$ sqlmap –u** [**http://testphp.vulnweb.com/artists.php?artist=1**](http://testphp.vulnweb.com/artists.php?artist=1) **–D acuart –T users –C uname –dump**

-C: indicates the column name

--dump: the entries of the column are fetched



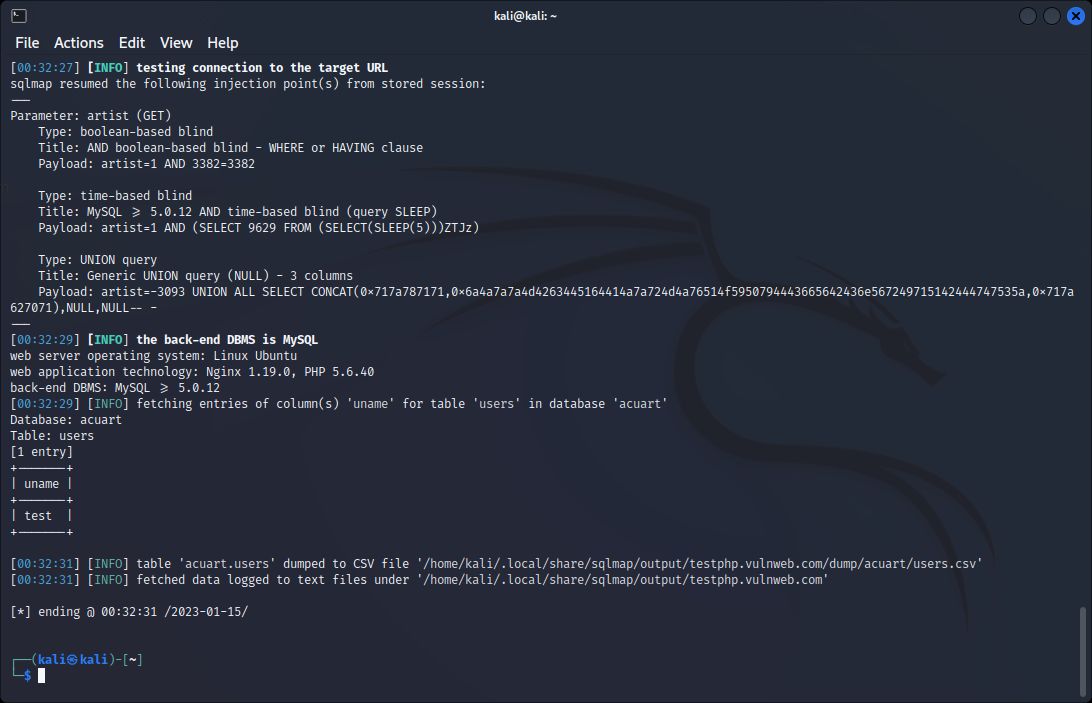
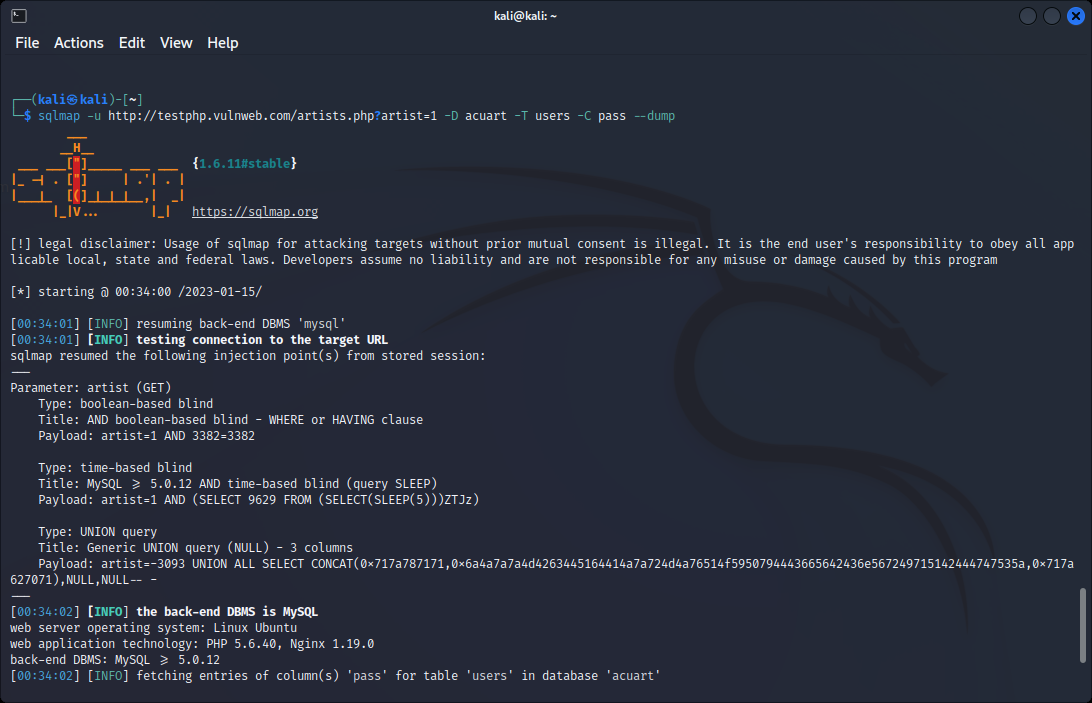


Fig: displaying the entries in the column uname

**$ sqlmap –u** [**http://testphp.vulnweb.com/artists.php?artist=1**](http://testphp.vulnweb.com/artists.php?artist=1) **–D acuart –T users –C pass --dump**

--dump: fetches all the entries in the column pass



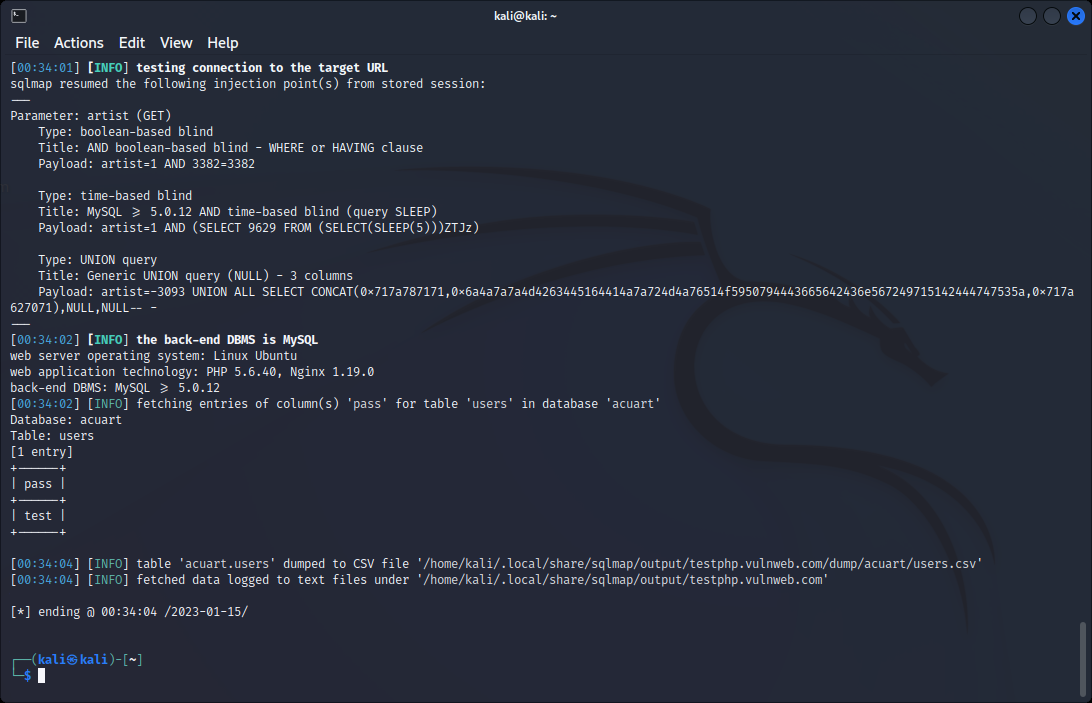


Fig: displaying the entries in the column pass

From sqlmap tool, we found out the username and password of the artist r4wB173

uname= test

pass= test

Using the username and password we can login

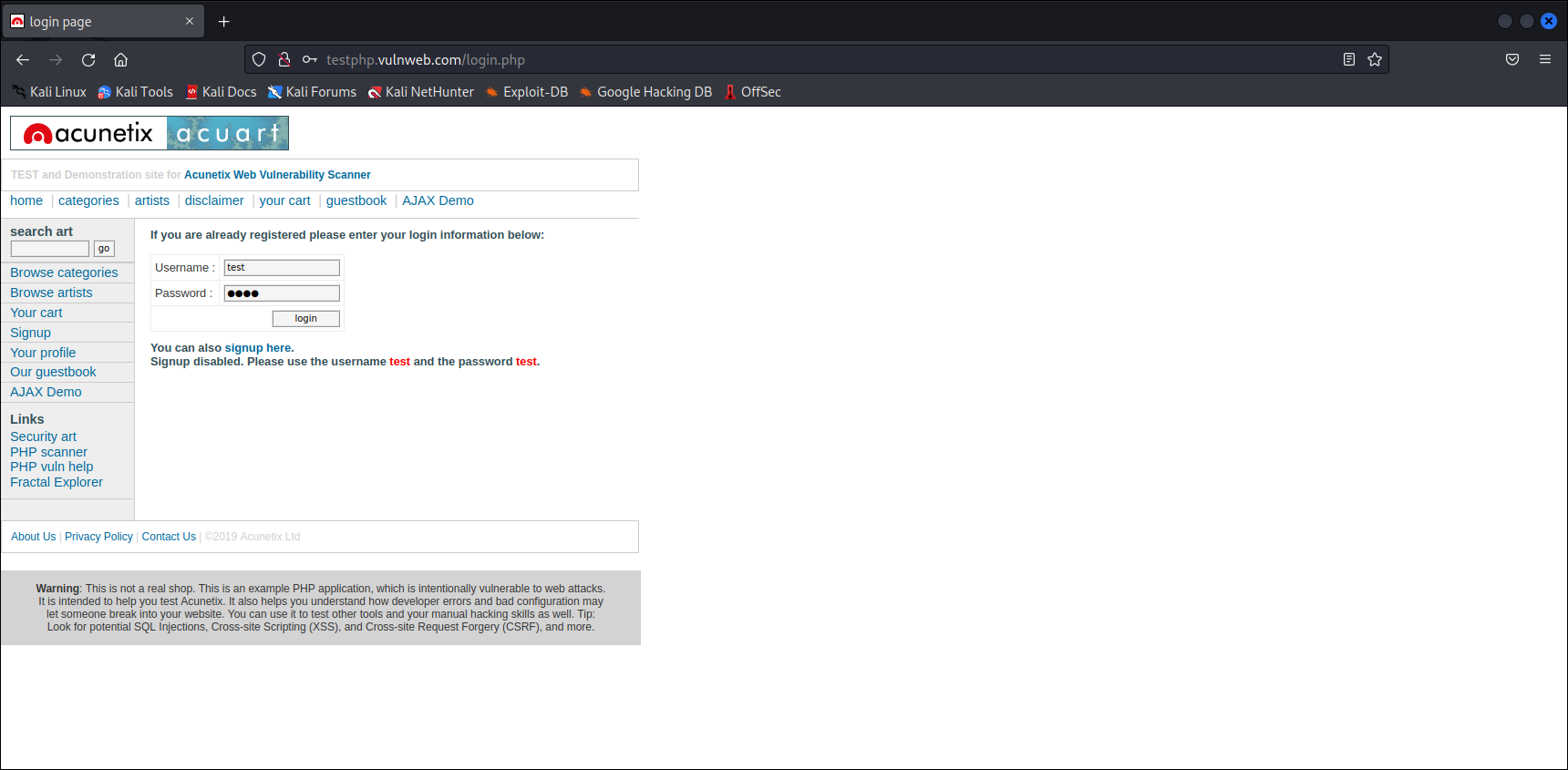


Fig: login page of the artist

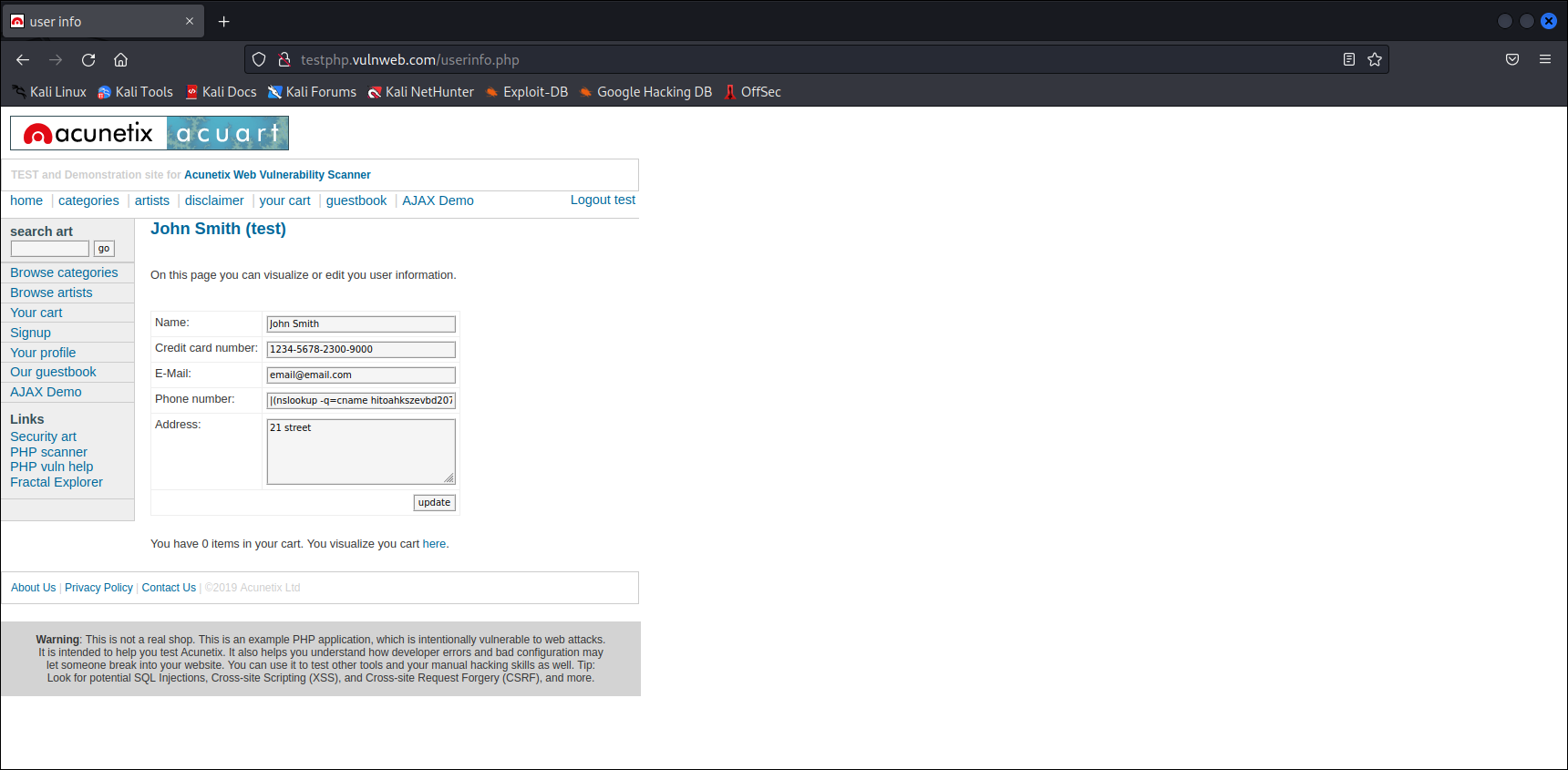


Fig: successful login to the artist’s account

**CONCLUSION**

SQL Injection attacks can exploit an organization’s database and control a database server behind a web application. Because they are relatively easy to implement, and because the potential reward is great, SQL injection attacks are not uncommon. Statistics vary, but it’s estimated that SQL injection attacks comprise the majority of attacks on software applications. According to the Open Web Application Security Project, injection attacks, which include SQL injections, were the third most serious web application security risk in 2021.

SQL Injection is a very popular attack method for Cyber Criminals. But taking proper precautions like ensuring the Data is Encrypted, Performing Security tests and by being up to date with patches, one can take meaningful steps toward keeping the data secure