
# SECURITY+ V4 LAB SERIES

# Lab 3: Analyzing Types of Web Application Attacks

**Document Version: 2022-04-29**

| Material in this Lab Aligns to the Following | |
|---|---|
| CompTIA Security+ (SY0-601)<br>Exam Objectives | 1.3: Given a scenario, analyze potential indicators associated with application attacks |
| All-In-One CompTIA Security+ Sixth Edition<br>ISBN-13: 978-1260464009<br>Chapters | 3: Application Attack Indicators |

# Contents

Introduction ..... 3
Objective ..... 3
Lab Topology ..... 4
Lab Settings ..... 5
1 SQL Injection Basics ..... 6
1.1 Using WebGoat for SQL Injection ..... 6
1.2 Using DVWA for SQL Injection ..... 17

## Introduction

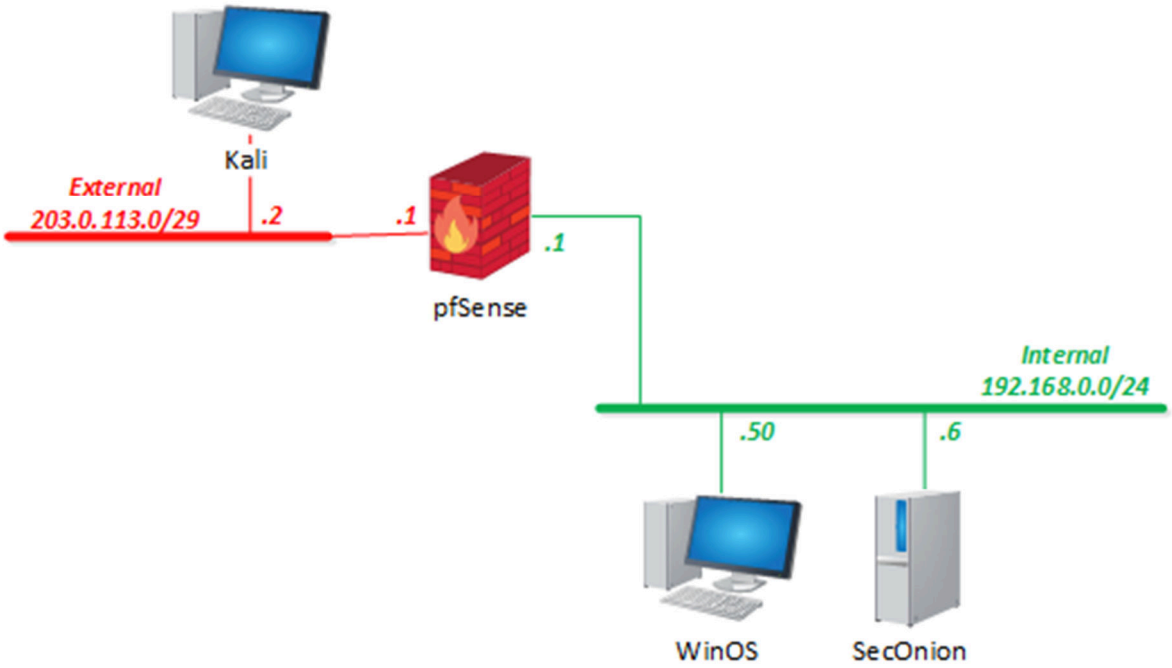
In this lab, various tools will be used to conduct web application securities practices.

## Objective

In this lab, you will perform the following tasks:

- Perform SQL injection attack
- Perform Cross Site Scripting (XSS) attack

## Lab Topology



## Lab Settings

The information in the table below will be needed in order to complete the lab. The task sections below provide details on the use of this information.

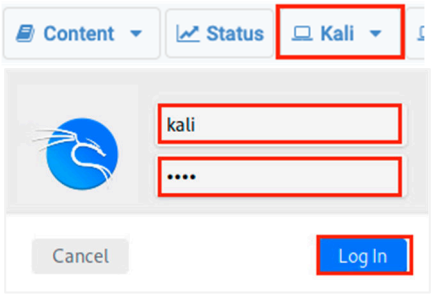
| Virtual Machine | IP Address | Account (if needed) | Password (if needed) |
|---|---|---|---|
| Kali | 203.0.113.2 | kali | kali |
| pfSense | 192.168.0.1 | sysadmin | NDGlabpass123! |
| SecOnion | 192.168.0.6 | sysadmin | NDGlabpass123! |
| WinOS | 192.168.0.50 | Administrator | NDGlabpass123! |

# 1 SQL Injection Basics

## 1.1 Using WebGoat for SQL Injection

In this section, you will be exploring how SQL Injection works and will also attack the vulnerable server.

1. Click on the **Kali** tab to access the Kali machine. Enter username `kali` and password `kali`.

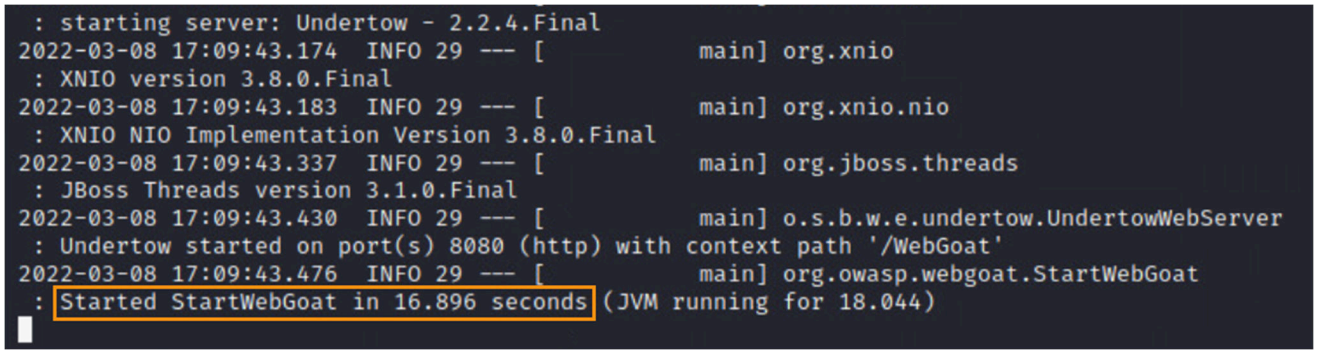



2. Once logged in, click the **Terminal** icon to start the *Terminal*.

3. In the *Terminal* window, run the following command. When prompted for a password, type `kali`.

```
kali@kali:~$ sudo docker run --rm -it webgoat/goatandwolf
```


4. Wait for a pause on the rolling message. You will see something similar to the screenshot below when everything is ready.

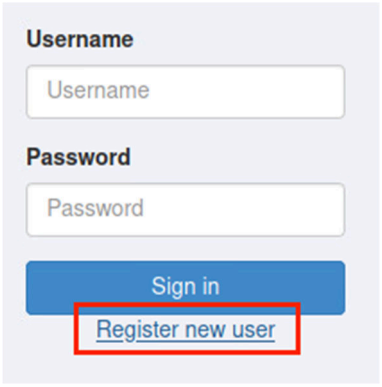


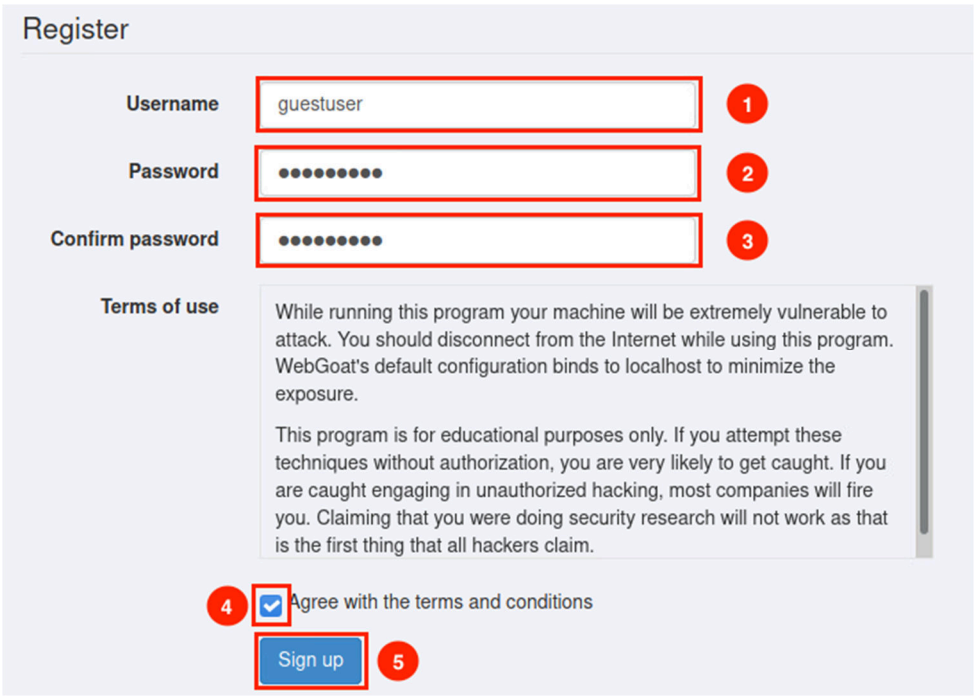

5. Start the *Web Browser* by clicking the icon.

6. In the address bar, type `http://172.17.0.2:8080/WebGoat/login` (case-sensitive) and press **Enter.**

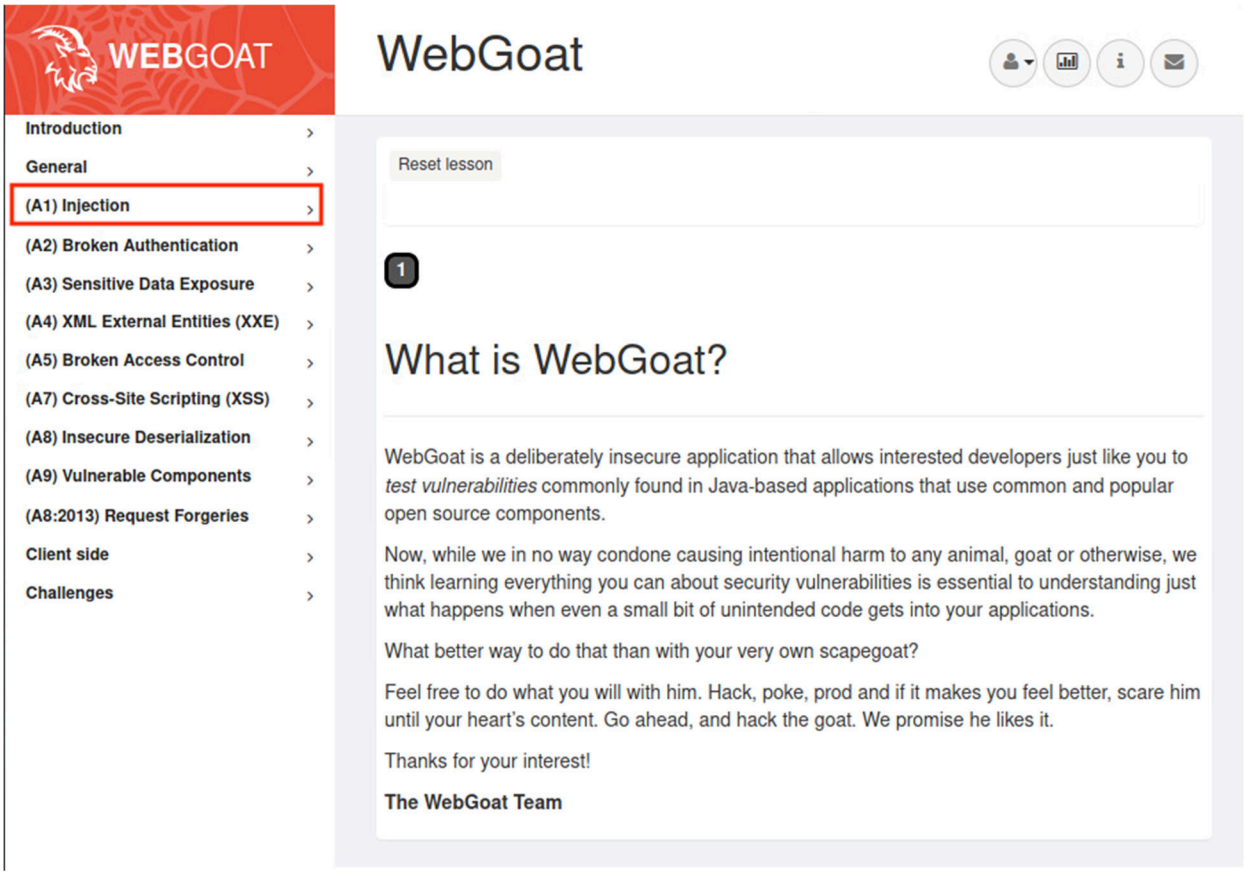

7. You will see the *WebGoat* login page, click **Register new user**, and then type `guestuser` as the username and `guestuser` as the password (you can choose your own login info, but here we are using **guestuser)**. Then, check the box, and click **Sign up.**



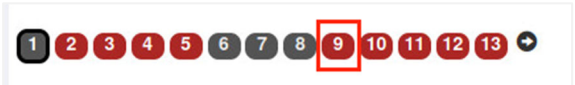

  

8. Once signed up, you will be brought to this page.

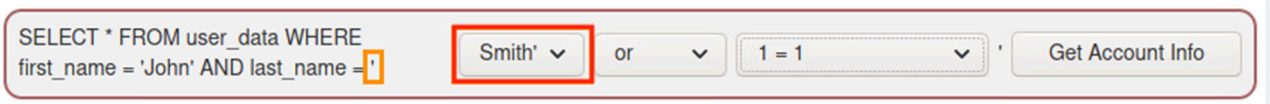



9. Click on the **(A1) Injection,** then **SQL Injection (Intro)**; you will then see the lessons in the right pane from 1 to 13.

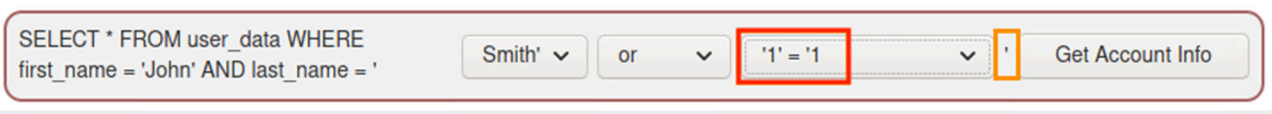

10. We will be jumping directly to lesson 9. So, click on the number 9 here. Feel free to do the lessons from 1 to 8 on your own. It will help you review the concepts mentioned in this lab.



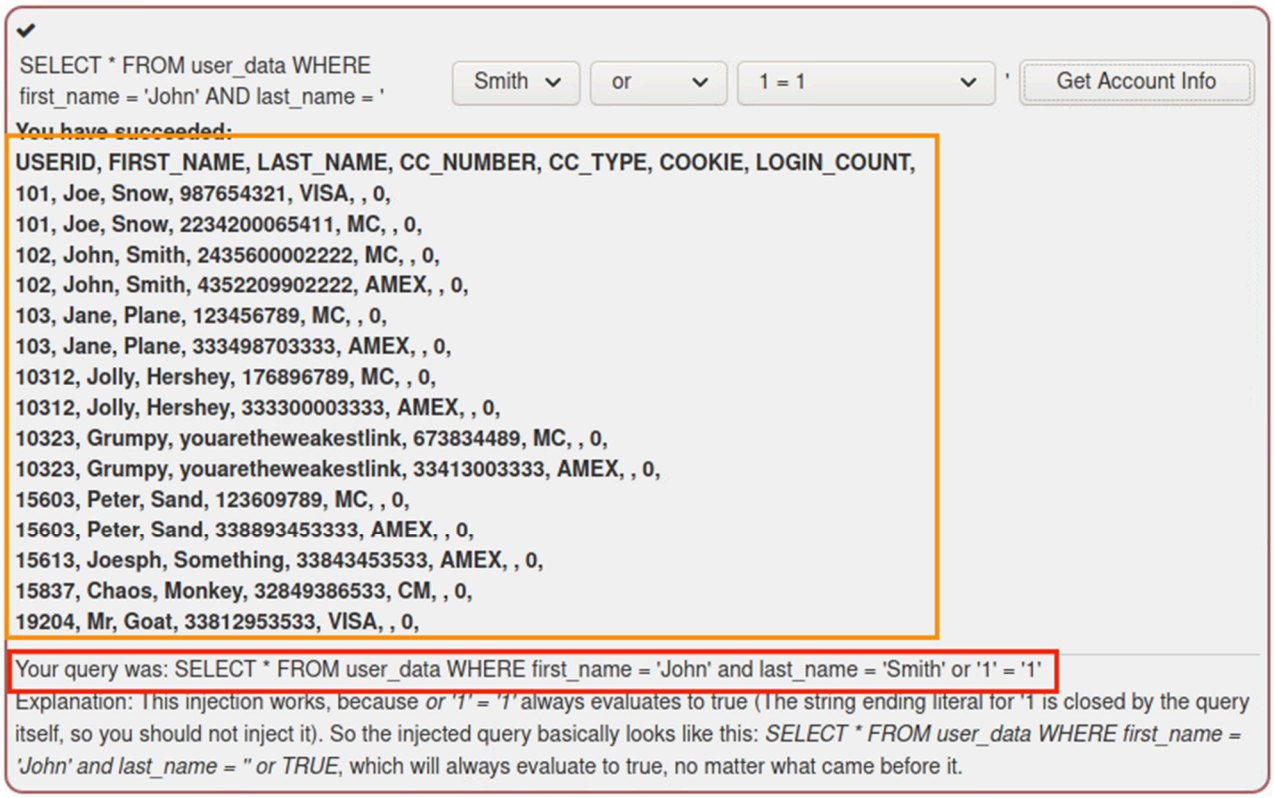
11. When in lesson 9, you will see this screen. After reading the instructions, we understand that the challenge is to select the malicious SQL command to retrieve all the users from the users' table. In the SQL query, notice the part where **last_name = '** ends with a single quote **'**. This means if we were to craft a malicious SQL query, we would want to close the single quote first. So, we select **Smith'** in the first dropdown box.



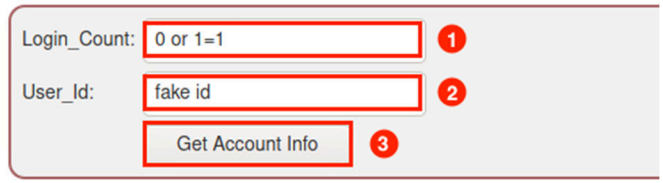

12. Next, we will leave the **or** untouched because it is the key part of the malicious query. And again, notice the single quote after **1=1**. We should correct the syntax, so we are going to select **'1' = '1** from the dropdown box.



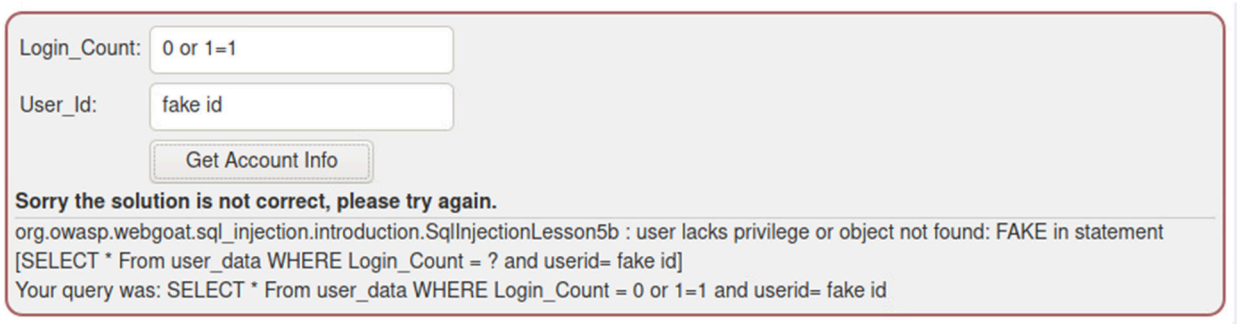

13. Clicking the **Get Account Info** button retrieves all the users from the *user_data* table. The red square shows the forged malicious SQL query.



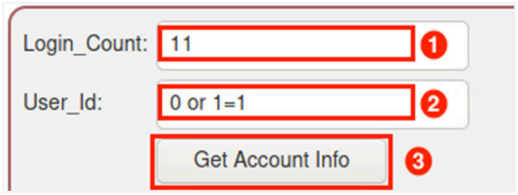

14. Now, click on **lesson 10**. In this lesson, we are asked to exploit two fields: the first one is *Login_Count*, and the second one is *User_Id***.** Since we now have two fields and we do not know which one is susceptible to SQL injection, we will have to try it in both fields. Let's begin with the *Login_Count*, type `0 or 1=1` and `fake id` as shown in the screenshot below, and then click the **Get Account Info** button.



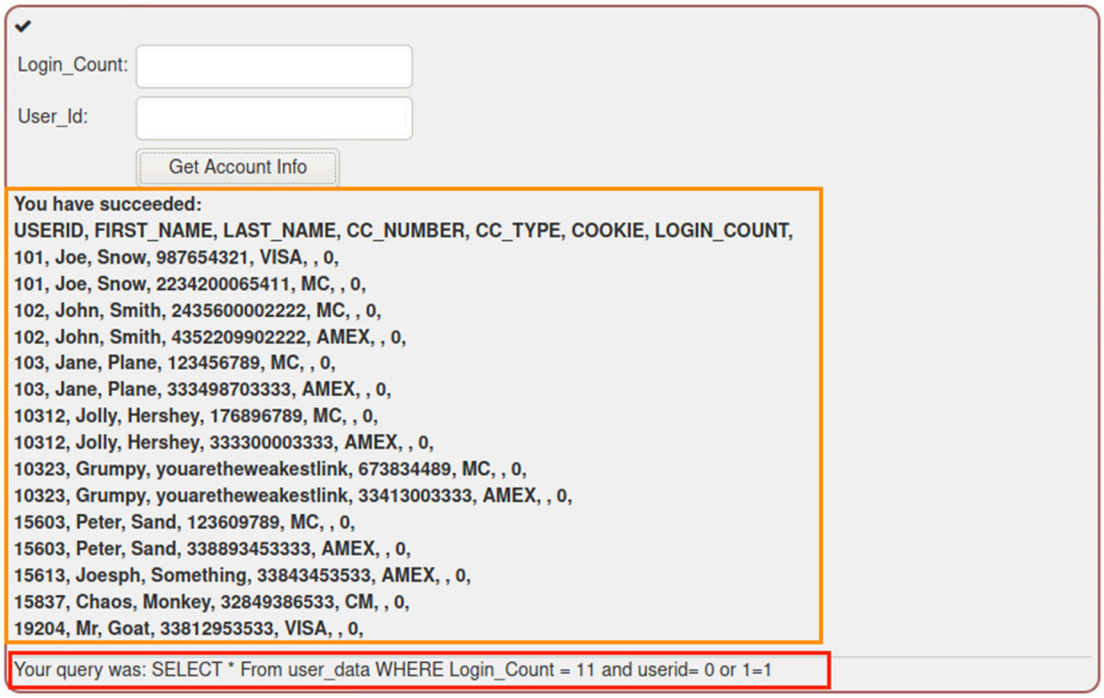

15. The result shows it is not correct**.**




16. Therefore, we will try to exploit the *User_Id*. Change *Login_count* to `11` and *User_Id* to `0 or 1=1` as shown in the screenshot below, then click **Get Account Info.**



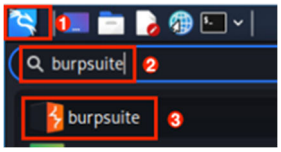

17. We now have the correct solution.



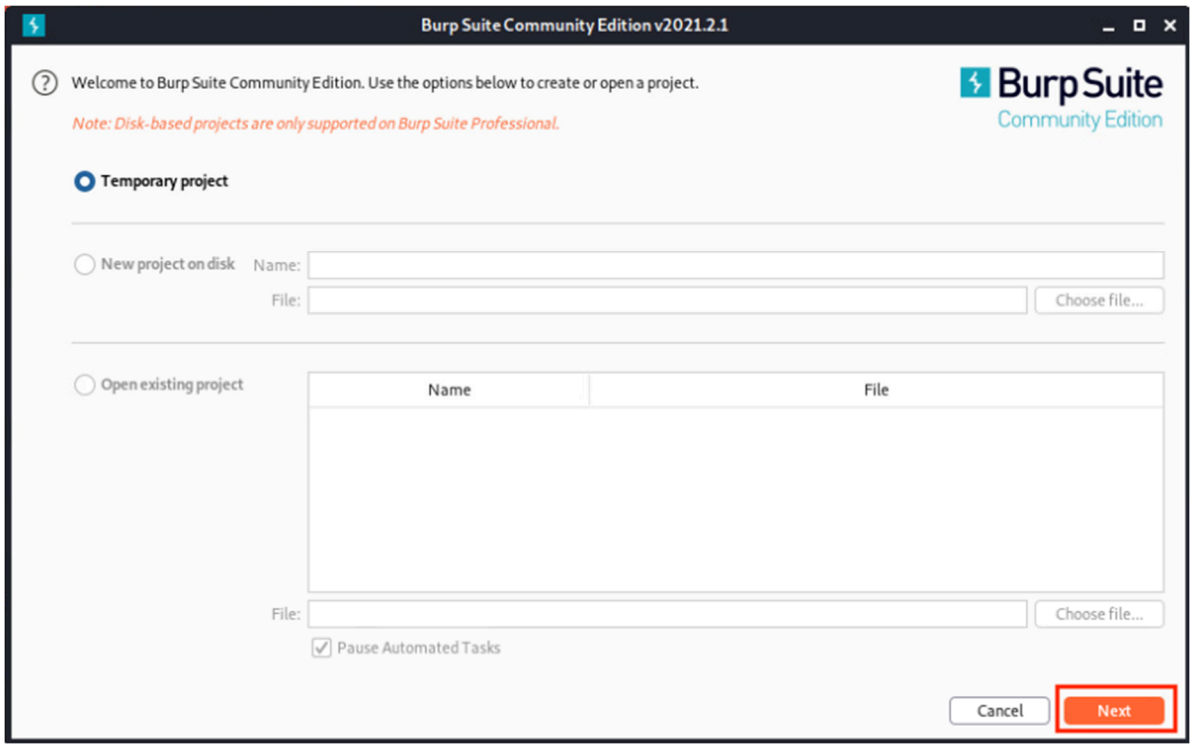

  

Keep checking the other lessons provided by the WebGoat. Use the hint if you feel stuck.

18. Let's now check a challenge and have a taste of what SQL injection could look like on a website. First, close the web browser, and click the top-left corner to bring up the Applications menu, then type `burpsuite`. In the result list, click **burpsuite** to launch the software.

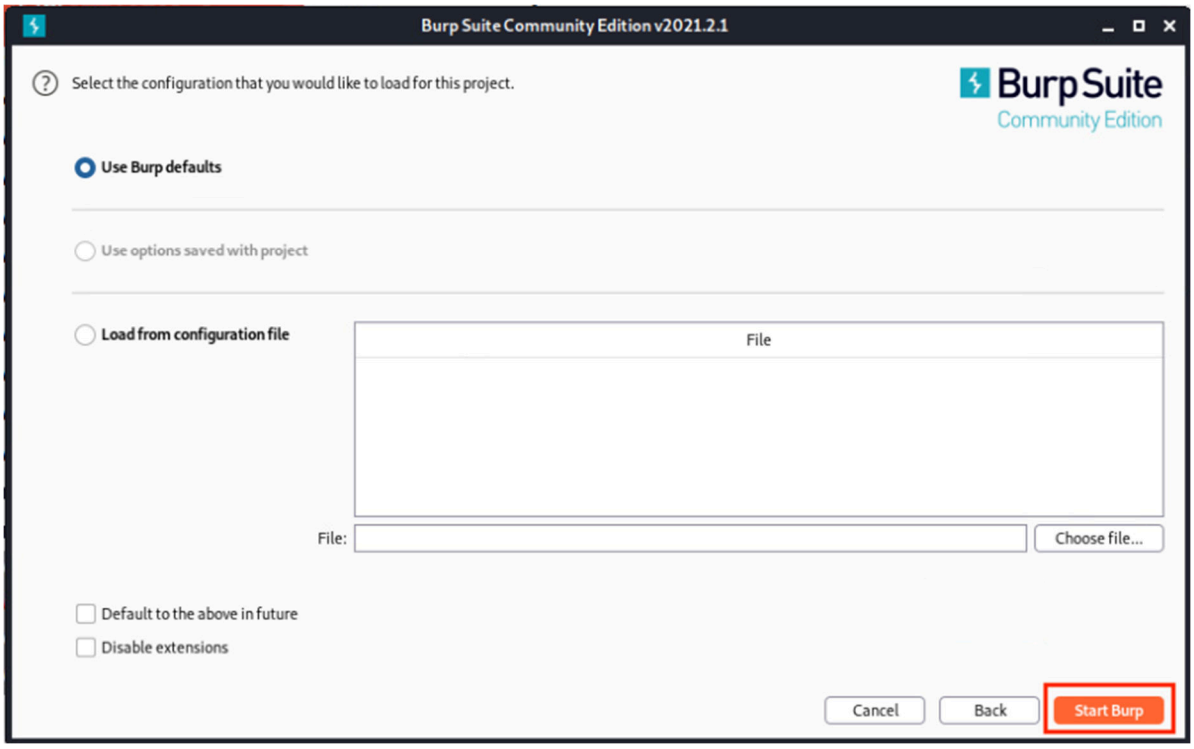



19. Click **Next** at this screen:

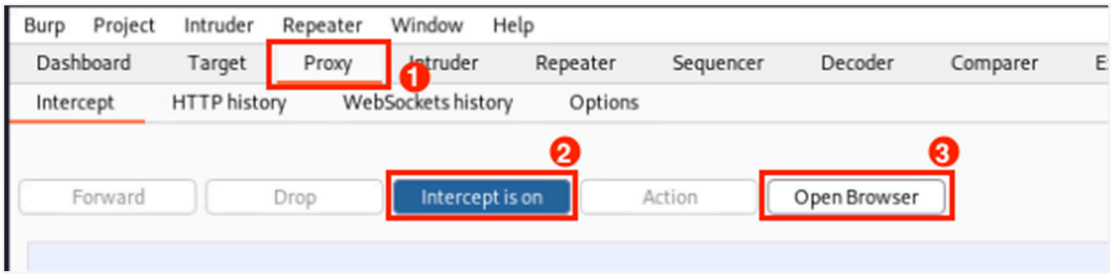


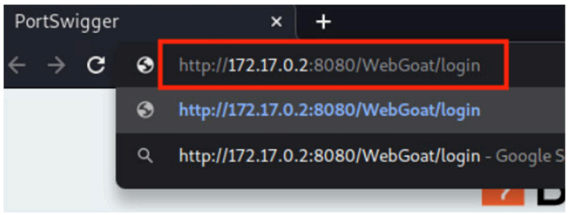
20. Then, click **Start Burp** on the next screen. If prompted for a new version, click **OK** to acknowledge it.



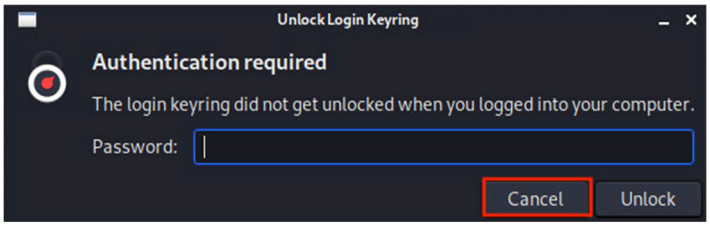

21. When *Burp Suite* starts, click **Proxy** to switch to the tab, then click **Intercept is on** to turn interception off for now (we will use it later). At last, click the **Open Browser** button.



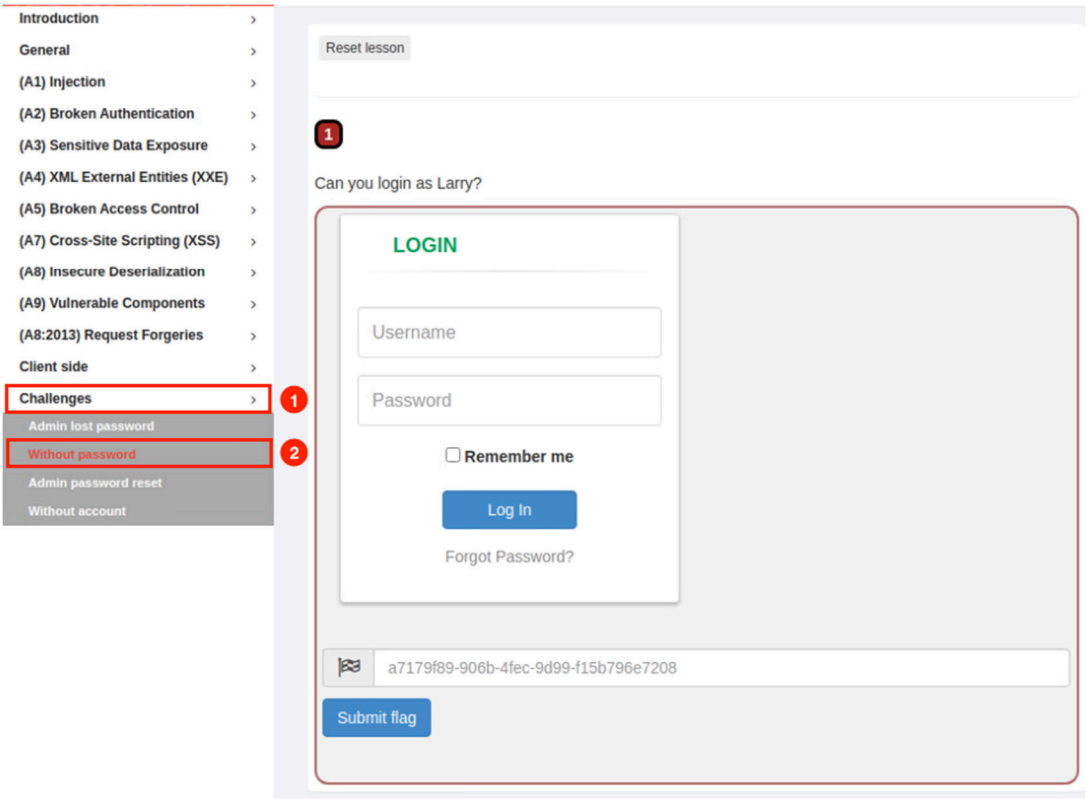

22. In the newly opened browser, go to the `http://172.17.0.2:8080/WebGoat/login` page again, and log in using the *guestuser* login you just created (username `guestuser`, password `guestuser`). When prompted to save the password, click **Never**.



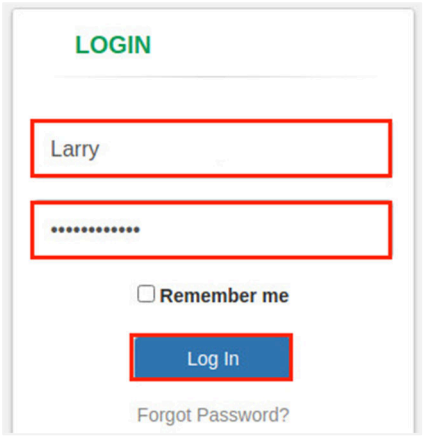
23. If prompted with *Authentication required (or choose password for new keyring)*, click **Cancel**. Click **Cancel** again when it prompts for the second time.



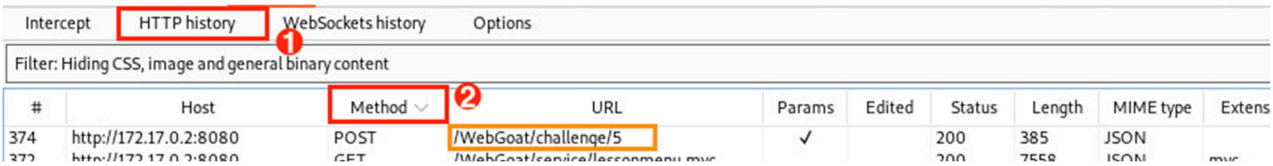

24. After you log in, click the **challenges**, then **Without password**. We will see the challenge like this:



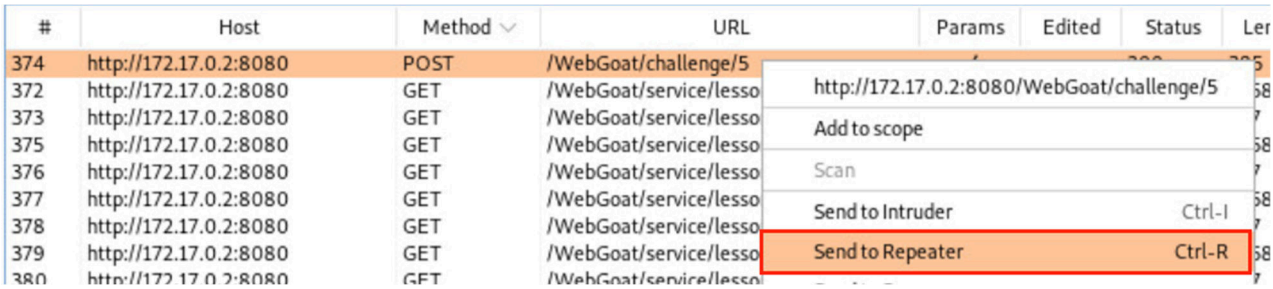

25. In the browser, fill in the username `Larry` and password `fakepassword`, and click **Log In.** This will create a POST request and send it to the WebGoat website.



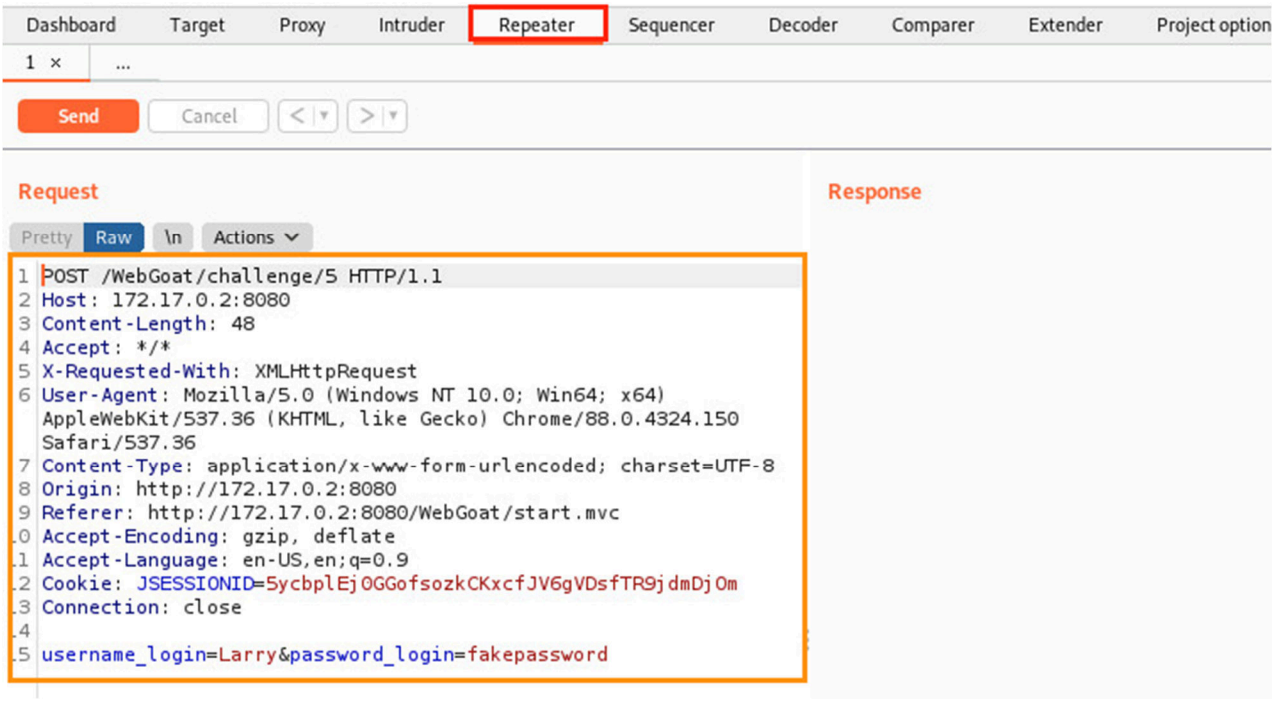
26. Switch back to the *Burp Suite* window, if you are not already, click on the HTTP history tab. In the table header, click on **Method** twice to reorder the list in descending order, find the **POST** request that has the **URL** of **/WebGoat/challenge/5**




27. Right-click on that **POST** entry, and choose to **Send to Repeater**.



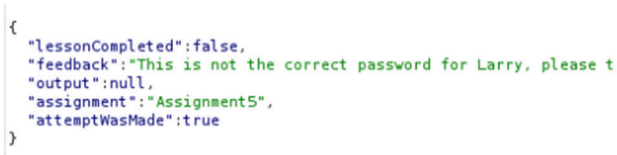

28. The **Repeater** will light up for a few seconds. Let's click the **Repeater** tab. You will see a screen like this:

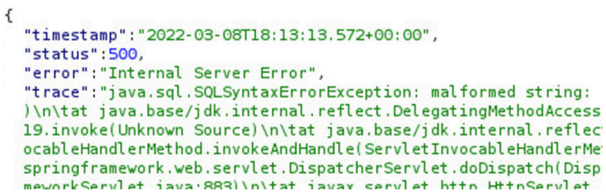



29. Repeater is a built-in function in *Burp Suite*; it functions like the *Tamper Data* add-on in Firefox you experienced in Lab 2. Here is how Repeater works. On the left side, the attacker could choose to change the content of the *POST* request, and then Send it to the server. The reply will be presented on the right side in the Response section. In our case, the *username* and the *password* we submitted earlier are shown at the bottom.


30. To exploit the SQL Injection, we just need to change our *username* or *password* fields and click **Send**. The result will show on the right side. Since the challenge is "Can you login as Larry?" we know that the username Larry must exist. So, we can manipulate the password field. Let's click in the *password_login* field and type the magic string: `0' or 1=1,` then click the **Send** button.


31. Unfortunately, our first try spawns an error (it could look like any of the following screenshots). It did not go through.



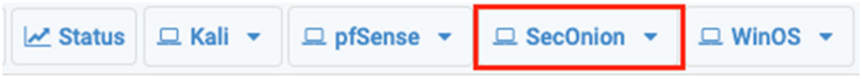
32. After a few retries, we learned that the "--" can play an important role when attacking with SQL Injection. The "--" will comment out everything after. Thus, the SQL injection query could be altered to `' or 1=1 --`

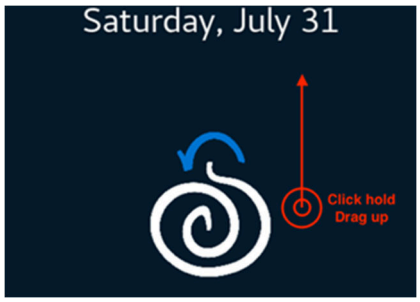

33. Feel free to submit the flag to the WebGoat website. This section ends here. You can close all windows. We will start a fresh Burp Suite and *Terminal i*n the next section.

## 1.2 Using DVWA for SQL Injection

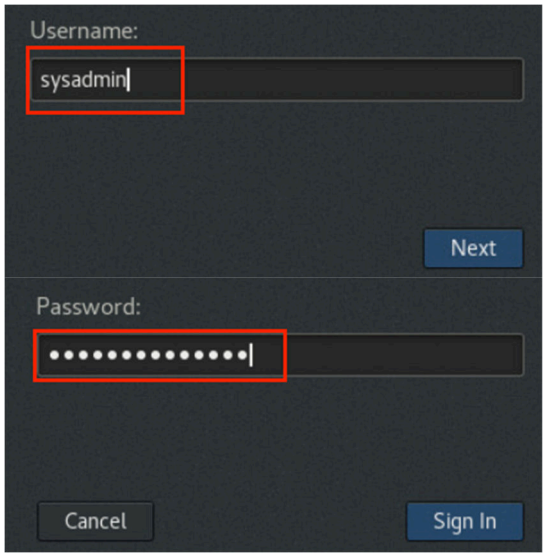
In the previous section, you learned how to do the SQL injection manually. In this section, you will learn how to do SQL injection automatically using *SQLmap*.

1. Click on the **SecOnion** tab, then click and drag up to unlock the screen for a login prompt.

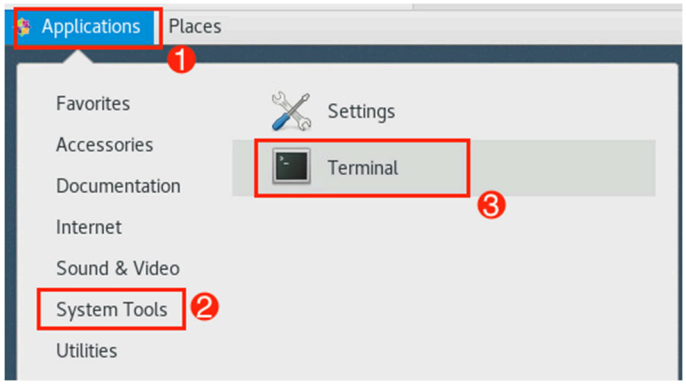



2. Type `sysadmin` as the username and `NDGlabpass123!` for the password.




3. Once logged in, click **Applications** > **System Tools** > **Terminal** to start the *Terminal.*



4. In the *Terminal,* enter the command below. When prompted, enter the password `NDGlabpass123!`:

```
sysadmin@seconion:~$ sudo docker run --rm -it -p 4444:80 vulnerables/web-dvwa
```

```
[sysadmin@seconion ~]$ sudo docker run --rm -it -p 4444:80 vulnerables/web-dvwa
[sudo] password for sysadmin:
```

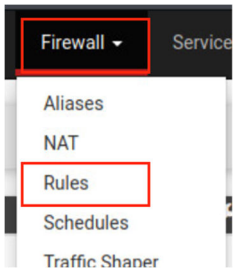
5. When you see something like this, it means the *dvwa* server is up and running.

```
[+] Starting mysql...
[ ok ] Starting MariaDB database server: mysqld.
[+] Starting apache
[....] Starting Apache httpd web server: apache2AH00558: apache2: Could not reli
ably determine the server's fully qualified domain name, using 172.17.0.31. Set
the 'ServerName' directive globally to suppress this message
. ok
==> /var/log/apache2/access.log <==

==> /var/log/apache2/error.log <==
[Tue Mar 08 20:02:35.240206 2022] [mpm_prefork:notice] [pid 316] AH00163: Apache
/2.4.25 (Debian) configured -- resuming normal operations
[Tue Mar 08 20:02:35.240301 2022] [core:notice] [pid 316] AH00094: Command line:
 '/usr/sbin/apache2'

==> /var/log/apache2/other_vhosts_access.log <==
```

6. Click back on the *Kali* tab to switch back to the Kali machine. Before we can use *SQLmap* we have to prepare the vulnerable database. Open a web browser again. In the address bar, type `http://203.0.113.1`.

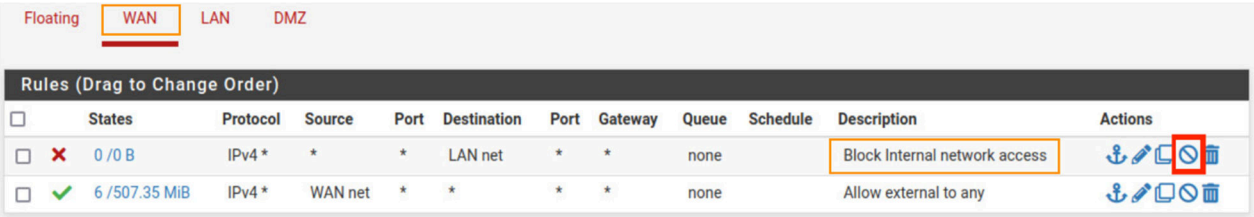

7. You will see the pfSense page; log in by entering `sysadmin` as the username and `NDGlabpass123!` as the password.


8. Once logged in, click the **Firewall** menu, and select the **Rules** option.

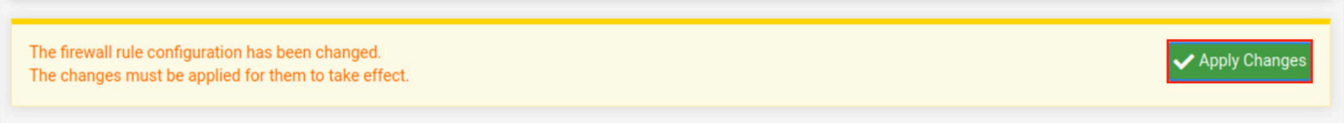


9. On the opened page, check to make sure you are on the *WAN* tab, and click the **disable** icon located on the right side of the first rule (*Block Internal network access* rule).




10. When it prompts that *rule configuration has been changed*, click **Apply Changes** to confirm the change.

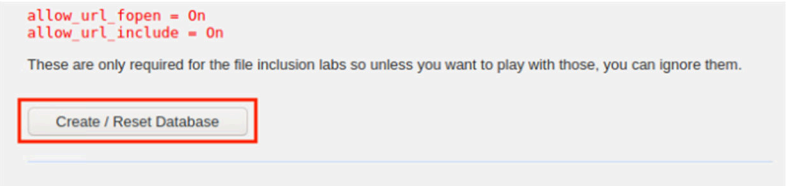



The first rule should grey out, and you can now access the internal network.

11. In the address bar, type `http://192.168.0.6:4444`.


12. Once the page opens, log in with the username `admin`, password `admin`.

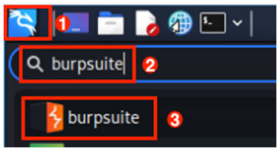

13. On the new page, scroll down to the bottom. Click the **Create/Reset Database** button. It will log you out.



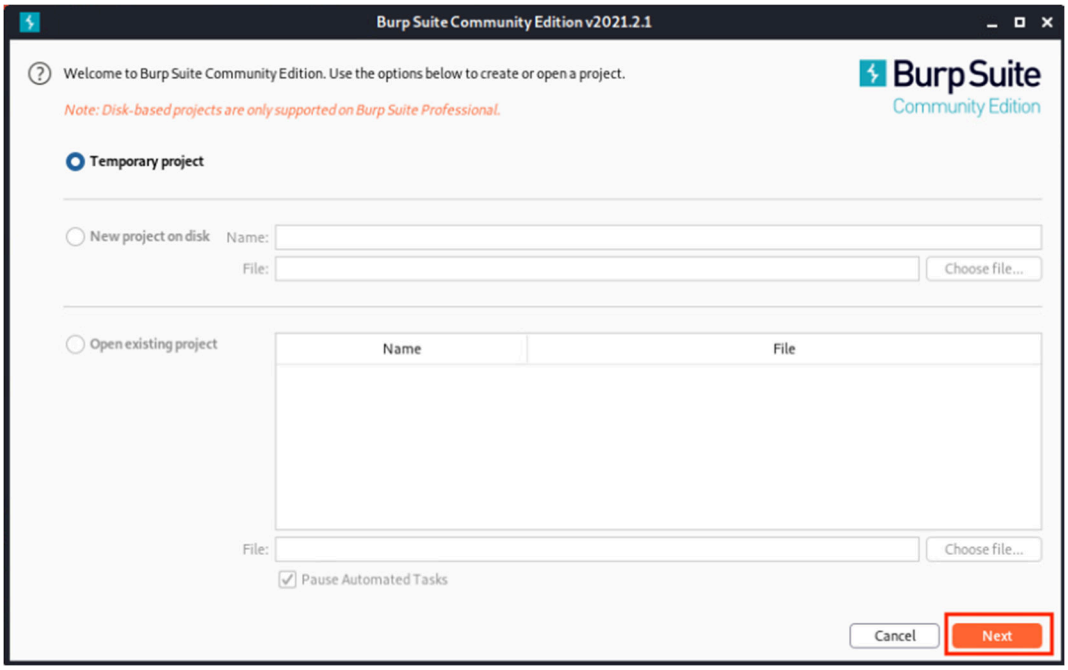

  

14. Log back in with the username **admin**, password **password**. When you see the *Welcome* page, you are all set. You can close the *Firefox* window.

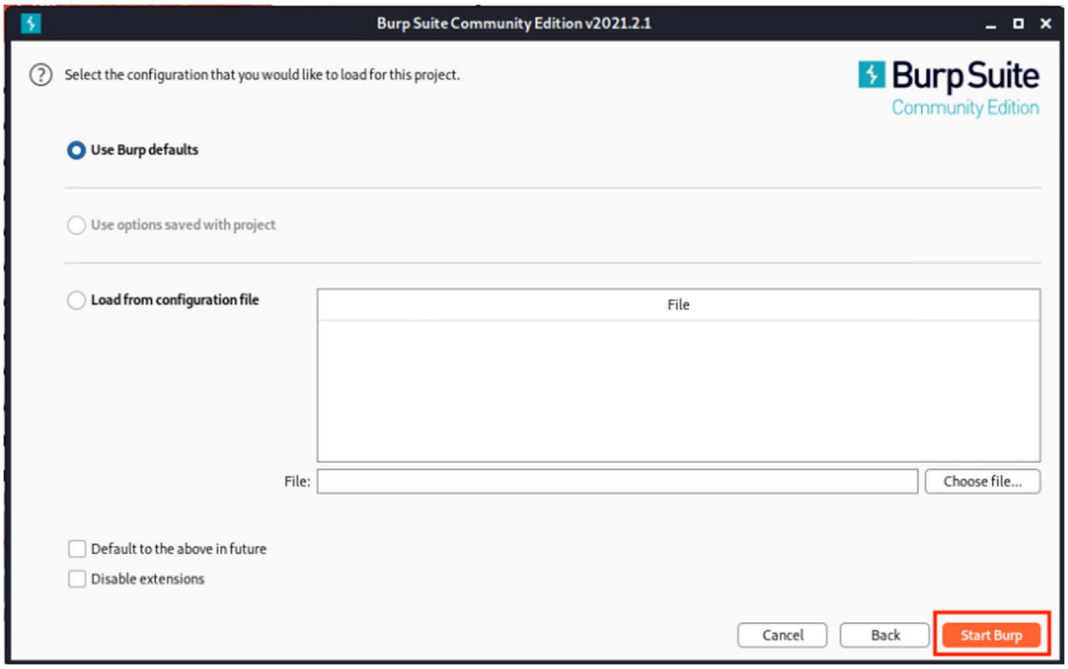

15. Now we will fire up Burp Suite once again to collect some information before we use *SQLmap*. Once again, go to the top-left corner, click the **Application** menu and type **burpsuite**, then click **burpsuite** to start the software.



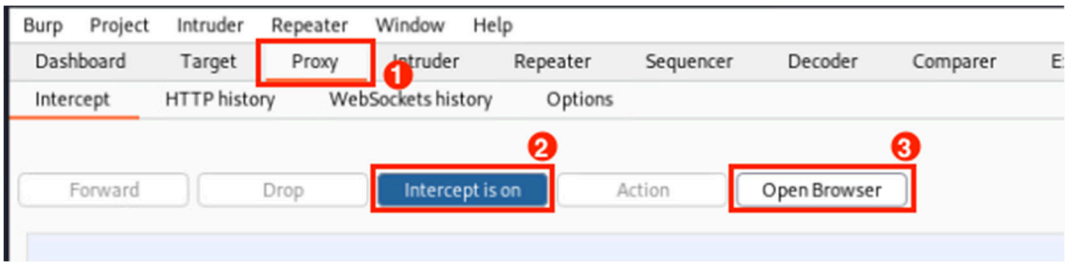

16. Click **Next** and **Start Burp** on the next two screens. (if Burp Suite prompts for an update, click **Close**, or if **Burp Suite** prompts to delete old temporary files, click **Delete**.)



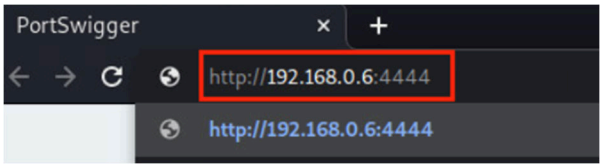

  




17. When *Burp Suite* starts, click **Proxy**, click the **Intercept is on** button to turn it off, then click **Open Browser**.

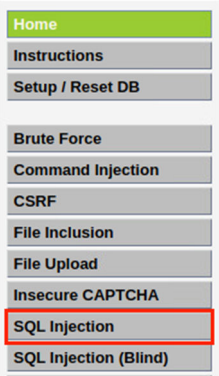



18. In the newly opened browser, go to the `http://192.168.0.6:4444` address. (if prompted to *Choose password for new keyring*, click **Cancel**).

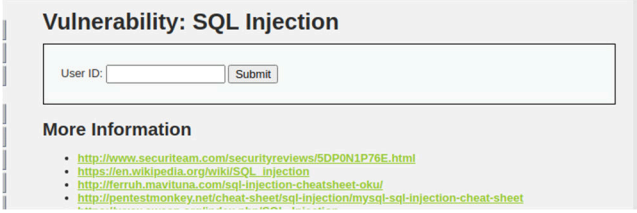



19. Log in as username `admin`, password `password`. When prompted to save the password, answer **Never**.

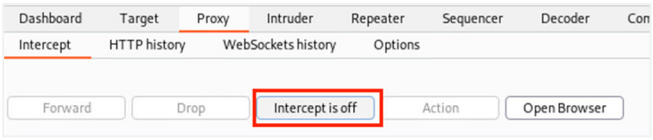
20. Click **SQL Injection** on the next page.



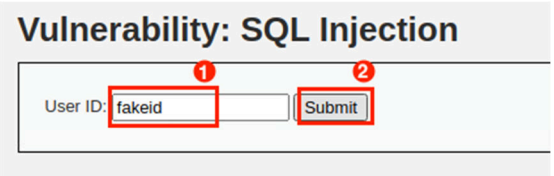

21. You will see this *Vulnerability: SQL Injection* page:




22. Go back to the *Burpsuite* window; you should still be at the *Proxy* tab. Click the **Intercept is off** button to turn it back on.




23. Then, switch back to the **Vulnerability: SQL Injection** window. We are going to type `fakeid`, and click **Submit.**

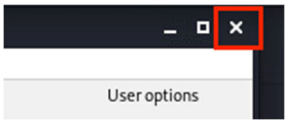



24. After you click the **Submit** button, it should bring you back to the *Burpsuite* window. We are going to need the following two pieces of information from this screen. Make a note of them on your computer or a piece of paper.

Your PHPSESSID is going to be different from the screenshot. Make sure you are recording what you see on your virtual machine.

```
/vulnerabilities/sqli/?id=fakeid&Submit=Submit
PHPSESSID=a21s0f61tb13j9333v5pecdtn5; security=low
```

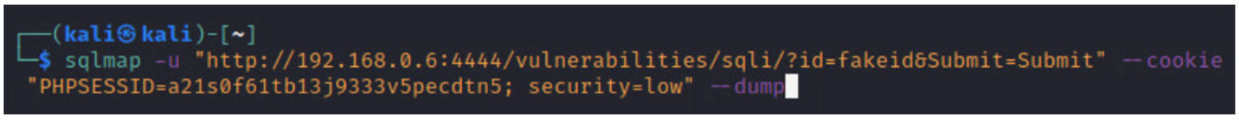

25. Once you have them written down, click the **X** button to close the *Burp Suite* window (if prompted to close all Burp Suite windows, answer **Yes**). The browser windows will close.



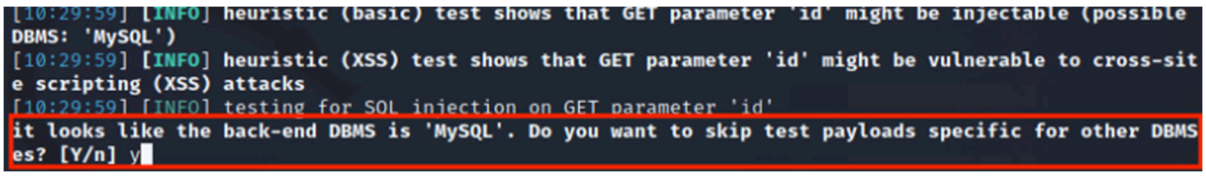

26. Click the **Terminal** icon to start the *Terminal*.

27. In the *Terminal* window, type the command below and press **Enter** to execute the command.

```
kali@kali$ sqlmap -u
“http://192.168.0.6:4444/vulnerabilities/sqli/?id=fakeid&Submit=
Submit” --cookie “PHPSESSID=buhibpdv4utvbekcvrd7ire5u0; security=low” --dump
```

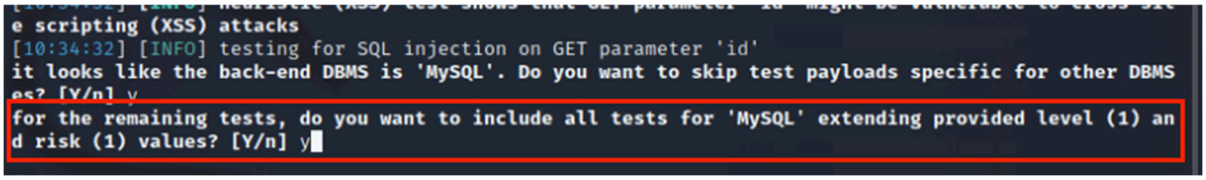



28. Type y for the question below, then press **Enter**.

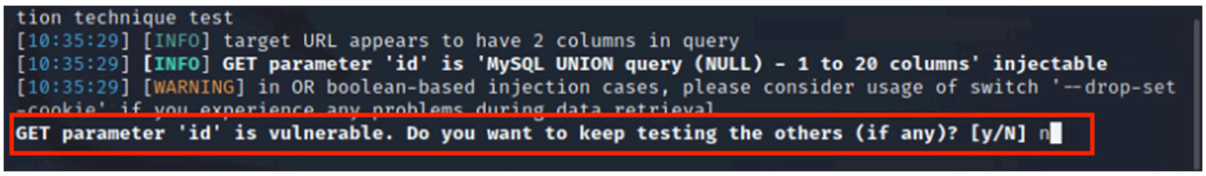


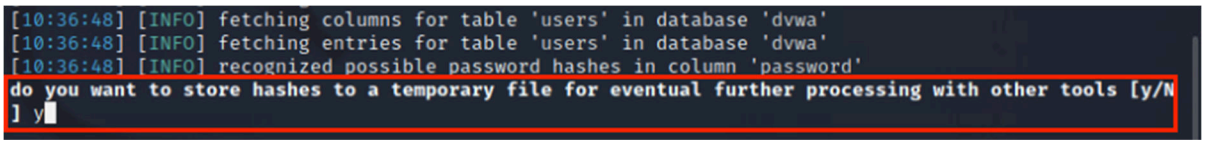
29. Type y for the question below, then press **Enter**.



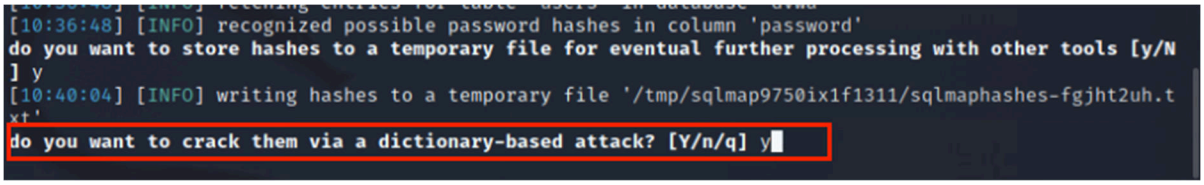

30. Type n for the question below, then press **Enter**.



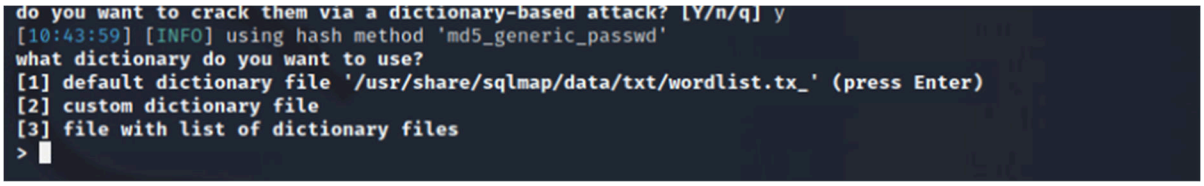

31. Type y for the question below, then press **Enter**.



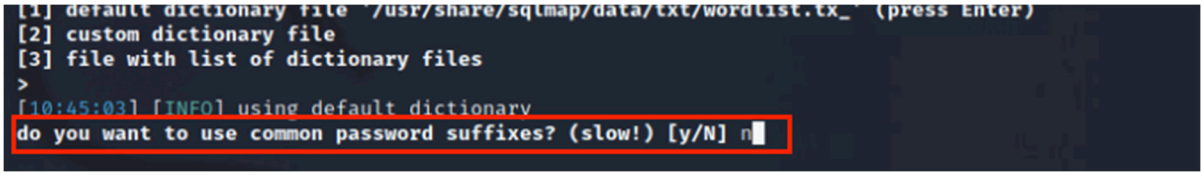

32. Type y for the question below, then press **Enter**.




33. Simply press **Enter** to use the first option.

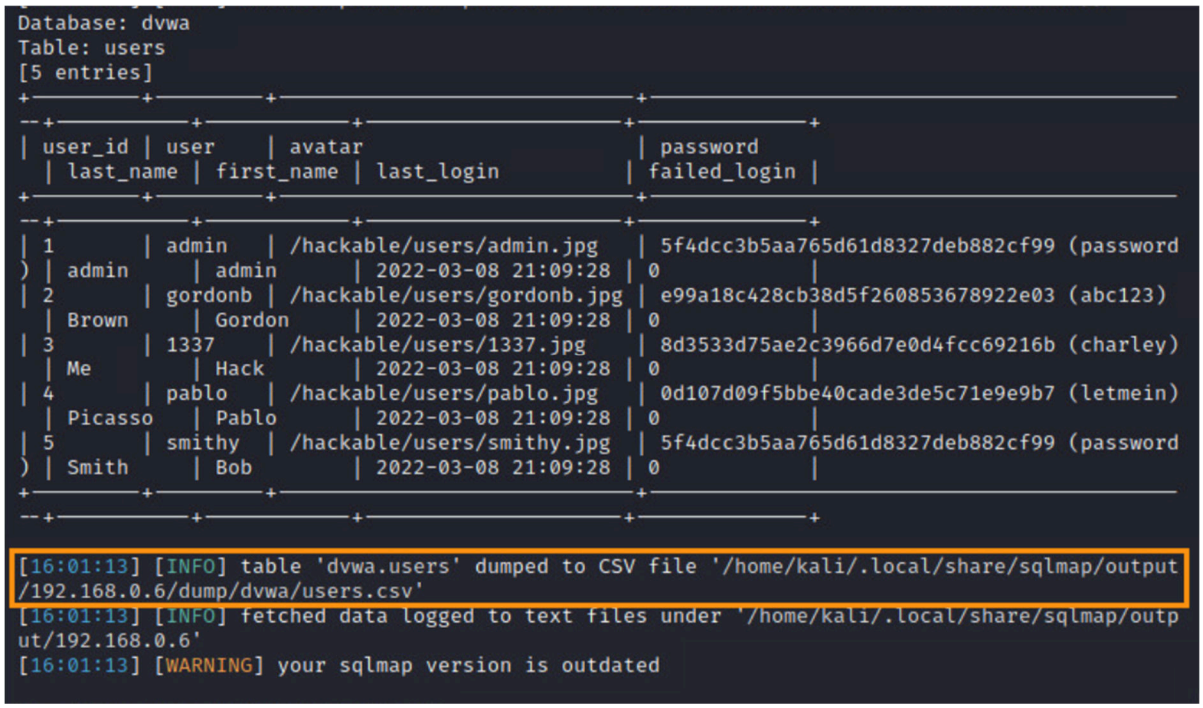



34. Type n for the question below, then press **Enter**.



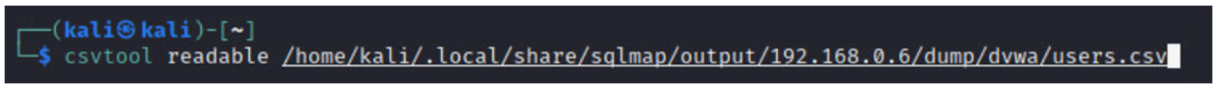

  

35. Wait till it finishes the SQL injection attack. Scroll up the *Terminal* window, until you see something like this. The rectangle indicates the place where the dumped database was stored.




36. Run the command below to have a better view of what is inside the table:

```
kali@kali$ csvtool readable
/home/kali/.local/share/sqlmap/output/192.168.0.6/dump/dvwa/users.csv
```




37. The results of the command will look like this. The passwords are stored in the database as hashes. Since the *SQLmap* decrypted all of them, you are seeing the cracked hashes in the parenthesis.


38. The lab is now complete; you may end your reservation.