# Lab-11-1: SQL Injection Attacks

In today’s lab, we are following on from the injection attacks we performed last week. We will start with a refresher exercise - using a web SQL injection exercise. After that we will use the DVWA platform to further learn about SQL injection.

**COPY OVER VIRTUAL MACHINES BEFORE STARTING THE FIRST EXERCISE**

Run the following script: L:\Virtual Machines\IN618\Lab-11-1.bat

**Exercise-11-1-1: SQL Injection Exercise 1**

Like the exercise in lab-10-1, we will start by using an online teaching tool to perform another SQL injection attack.

Open Firefox (on your Windows 10 system) and navigate to the following URL:

<https://www.hacksplaining.com/exercises/sql-injection#/start>

Take your time to complete the exercise and carefully read each step, look at the server logs and code box. This exercise is very like the other SQL injection attack we have performed and should provide another example of how this type of attack works.

**Q1.** Document the password you used to perform the SQL injection attack:

‘ or 1=1 --

**Q2.** Document the SQL query that resulted from the SQL injection attack.

SELECT \*

FROM users

WHERE email = 'user@email.com'

AND pass = '' or 1=1 --' LIMIT 1

**Q3.** Describe in your own words how the attack works.

I changed the logic of the sql query. I said if the password = ‘’ or if 1=1, which can always be true because 1 can only ever = 1.

**Q4.** What is the “Available Balance” in the “Savings” account for the [user@email.com](mailto:user@email.com) account.

$16,100.44

**Exercise-11-1-2: SQL Injection Exercise 2 (using DVWA)**

Following on from last weeks class (10-2) we are going to use the DVWA learning platform to perform an SQL injection attack.

**Make sure that Kali Linux and Ubuntu-WebApps are both running.**

Log into you Kali Linux machine.

Open the Firefox web browser.

When in Firefox: enter the following URL:

http://192.168.19.123/DVWA/

Login using the following credentials:

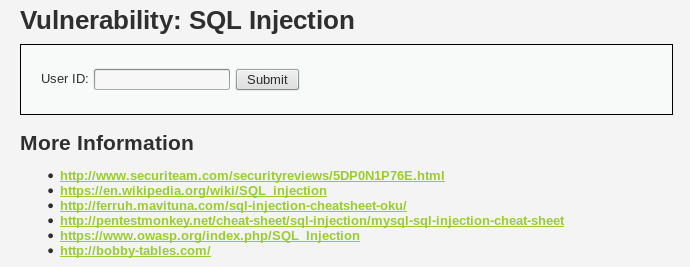
* Username: admin
* Password: password

Make sure to set the “security level” to low:

* On the left menu, select “DVWA Security”
* Using the drop-down menu, select “Low” security level
* Click “Submit” to save the changes
* On the left menu, select “Home”
* Check in the bottom left corner that “Security Level” is set to “low”

Excellent! We are now ready to start the SQL injection exercise.

Using the left menu, select “SQL Injection” to start the exercise. You will be greeted by a vulnerable web application and a “User ID” input box, as displayed below:



This web application has some severe vulnerabilities to SQL injection attacks.

Start this exercise by reviewing the help menu for this specific exercise. If you remember from last week, we can view the help menu by clicking the “View Help” button in the bottom right hand corner of the browser window.

**Q5.** Based on information from the exercise help menu, how many users are in the database? Also, try to list all the user IDs.

There are 5 users in the database, with id's from 1 to 5.

OK, so now we know that there is a collection of users, and they all have a unique user identification number (you can think of this user ID number as like you student ID number at OP). Try entering in each of the user ID numbers and reviewing the output. If successful the database should print out the “First name” and “Surname” of the user.

NOTE: If you are having problems finding the user ID numbers, ask a fellow student, or the lecturer for some help.

**Q6.** List all the “First names” and “Surnames” of all the users in the database.

First name: admin

Surname: admin

First name: Gordon

Surname: Brown

First name: Hack

Surname: Me

First name: Pablo

Surname: Picasso

First name: Bob

Surname: Smith

Well done! The information we have extracted is quite interesting and we have started to gain an understanding of how this web application works (specifically, how the userID button works).

We should probably check out the source code of how the web application works.

Find the “View Source” button, it should be in the bottom right corner of the browser window. Please do not use the built-in “View Source” button that is available in Firefox (which is found in the menu when you right-click the web page).

When the “View Source” button is clicked, a new window will appear. We can see from the first line that this script is written in the PHP language…. EEEEEK!

We have never covered PHP so far in the BIT, but we do not need to know the intricacies of the PHP language to understand what is being performed by the script. Locate the line that reads:

// Check database

$query = "SELECT first\_name, last\_name FROM users WHERE user\_id = '$id';";

To provide a better understanding of this code, here is a summary of each component.

* $query is a variable that stores an SQL query
* The remainder of the line is a string that forms an SQL query
* $id is the variable provided by the user (e.g., when a user enters “1” into the web application, the $id variable is transformed to “1”)

For example, when you enter “1” into the userID box, the following SQL query is created:

SELECT first\_name, last\_name FROM users WHERE user\_id = '1'

Given your experience in SQL classes with Krissi, this SQL query should make perfect sense to you! Based on the information from the query above, answer the following questions:

**Q7.** What is the database table called?

users

**Q8.** What field (rows) are available in the database table?

First\_name

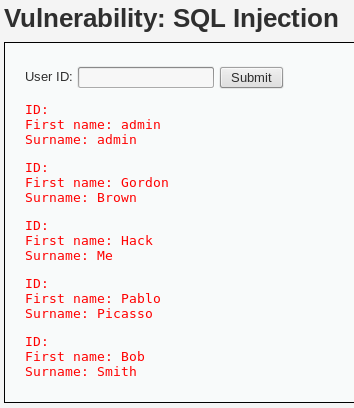
Last\_name

So, now we have a lot of information about the database, what tables are available (users), an idea of some of the rows (first\_name and last\_name), and how SQL queries are constructed from user input (aka the userID box on the web application).

Close the “View Source” window, so we can continue with the SQL injection exercise.

The next goal is to try to construct an SQL query that will print all the user’s at once. We can accomplish this using an SQL injection attack that is very similar to the answer for question 1. Basically, we need to enter some code into the userID box that will for the SQL query to always return true.

This task may be difficult. Try different inputs to the userID box and review the output, or SQL error that you get. FYI – when you get the right input, the output from the web application will be the same as the figure below:



**Q9.** What did you enter in the userID box to list all the user details?

1' or 1=1 ;#

OK, so we have exploited the database to supply information to us. But we have barely touched the surface of what information we can extract. The final part of the lab will involve using the UNION command to extract information from the database (that we are not supposed to access!).

I am hoping that you know all about the UNION operator, which basically combines the results of multiple SELECT statements. Here is a link for more information: <https://www.tutorialspoint.com/sql/sql-unions-clause.htm>

Before going any further, it would be very useful to determine what type of SQL server is running. Using the power of the UNION operator, we can issue command against the SQL Server. NOTE: we can only do this because we have the ability to inject SQL.

' UNION SELECT null,@@version#

Here is a brief overview to summarise what is happening. The two at symbols (@@) refer to a global variable available in SQL and the “version” command will dump the SQL database version for us. We must put two values in the SELECT statement (null, @@version) because the PHP script wants two variables to print out. You could easily change “null” to 1, or 2, @@hostname, or anything really!

**Q10.** What SQL database version is the web application using?

5.7.18-0ubuntu0.16.04.1

Final part of today’s lab! We are almost done!

OK, so we have figured out we can use UNION to execute two SELECT statements. We should use this for our advantage. There is a way to find out the other rows or fields that are available in the “users” table. We know there is “first\_name” and “last\_name”, and there is probably another row that contained passwords. We can guess that this row is called “password”

To get the user’s passwords we will have to write another SELECT statement. Say, for example, we wanted to dump the password for the user with the ID or ‘1’ we could write the following SQL query.

SELECT first\_name, password FROM users WHERE user\_id = '1'

But remember, we must include the SQL injection portion for this to work. This will be the answer to question 9. After that, we can use the UNION command to add another SELECT statement. Try the following:

1' OR 1=1 UNION SELECT first\_name, password FROM users#

**BONUS Q11.** What hashing algorithm is used to secure the user’s passwords?

SHA1

**BONUS Q12.** Crack the user’s passwords using hashcat or an online website. Document the user’s hashed passwords and the decrypted passwords below:

First\_name: Hack

Password: Charley

First\_name: Gordon

Password: abc123

First\_name: admin

Password: password

First\_name: Pablo

Password: letmein

First\_name: Bob

Password: password

Well done for getting this far!

As further reference for this lab, there are many tutorial on the information we covered today for the specific SQL injection exercise on DVWA:

* <http://www.securityclown.com/dvwa-sql-injection/>
* <https://pentestlab.blog/2012/09/18/sql-injection-exploitation-dvwa/>
* <http://www.hackingarticles.in/sql-injection-exploitation-dvwa-bypass-security/> (more advanced than the above two).