Laboratory Exercise E3 - Web Application Security SQL Injection

Author: Dr. David Raymond
First Edition: August 2017
Last Updated: July 2026

Due Date:

Points Possible:

Overview

This laboratory exercise will provide hands-on experience with a particular web application vulnerability known as SQL injection. SQL injection takes advantage of web application flaws that might let a user maliciously manipulate a web application's back-end database.

Resources required

This exercise requires the latest Cyber Range Cyber Basics environment in the Cyber Range.

Initial Setup

Log in to your Cyber Range account and select the latest Cyber Basics environment. Start your environment and, if necessary, log in using the username **student** and password **student**.

Tasks

Task 1: Log in to DVWA

On your Cyber Range Kali Linux system, open a web browser and enter the following into the url bar: **http://dvwa.example.com/**

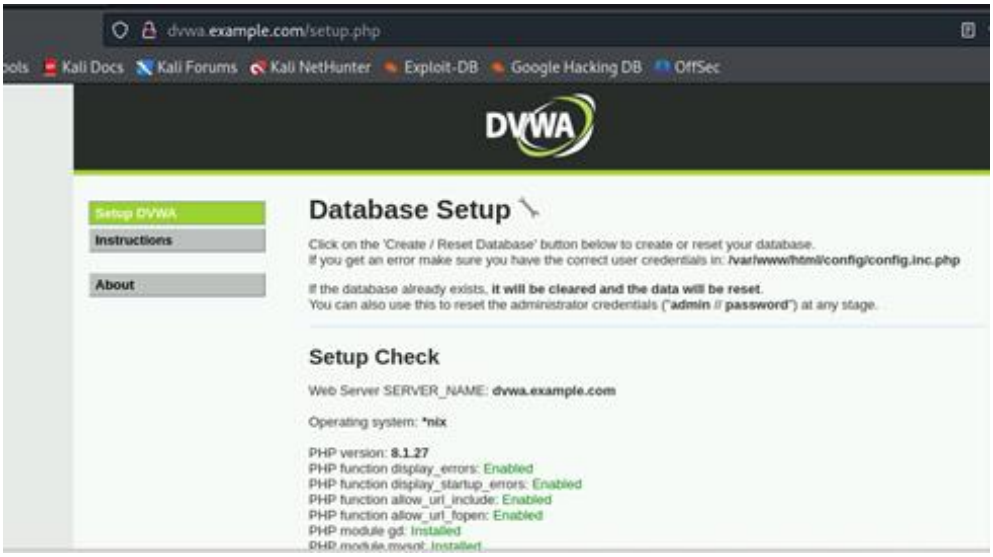
You should end up at the DVWA login screen. Log in to DVWA with these credentials:

Username: **admin**

Password: **password**

You will be prompted to 'Create/Reset Database' by clicking on the button at the bottom of the page. You will then be required to log back in using the same credentials.

Once logged in, click the **DVWA Security** button on the left side of the page and check that it is set to '**low**', then click '**submit**'. DVWA provides a range of security levels so users can test their skills and try different techniques to bypass increasingly secure web application implementations.



Task 2: SQL Primer

SQL, or Simple Query Language, is a language used to interact with relational databases. Most modern websites use databases to store their content. If the website has a page where a visitor enters search terms or other interactive content, they are often interacting with a database running on the same network as the web server. Any time a user can enter data, there is potential for abuse.

The most basic SQL command is to look up values in a database table that match a specific criterion. The format for this lookup is here:

```
SELECT [columns(s)] FROM [table] WHERE [search_criteria]
```

Here is a sample lookup using the "**users**" table below as an example:

userID	user	first_name	last_name	password
1	admin	admin	admin	a48dd378e15s5f3d16e31x1f1bb1ae91
2	gordonb	Gordon	Brown	15e3b2c347791d6187ab88719cd3cc19
3	pablo	Pablo	Picasso	12e3f178259e151967c5df13789d1548

```
SELECT first_name FROM users WHERE userid = 2
```

For this query, the SQL database will evaluate each row of the database and display the results for which the search criteria are met (in other words, are evaluated by the system as 'True'). The result of the above SQL query is **Gordon**.

Below is another similar query. Use the data in the table above to determine the answer to this query and write it in the space below.

```
SELECT last_name FROM users WHERE first_name = 'Pablo'
```

Questions

1. What is the result of the above SQL query?

Note:
It is important to understand that the *search criteria* will sometimes be met multiple times, in which case data from multiple rows in the database will be displayed. In the first example, the search criteria of "userid < 3" would return the first names **admin** and **Gordon**. In some cases, the "search_criteria" will be true for every row. This is the case when the search criteria always evaluate to "True". For example, 1==1 is ALWAYS "True".

Task 3: Hands-on with SQL Injection

Click on the 'SQL Injection' button on your DVWA screen The input box on the SQL Injection page asks for a 'User ID'. If you enter a '1' in this field, the web page constructs the following SQL query:

```
SELECT first_name, last_name FROM users WHERE user_id = '1'
```

If you were to enter something that would always evaluate to 'True', the database would return every first and last name in the database. For example, entering the following: **1' OR user_id=2 #** in the 'User ID' field results in the following query being constructed:

```
SELECT first_name, last_name FROM users WHERE user_id = '1' OR
user_id='2' # '
```

Note:
The **'** after the **1** closes the quote, the **OR** creates a Boolean expression with the **user_id='2'**. This additional information extends the Boolean expression so that it matches more rows in the database table. The **#** is an SQL comment marker that will cause the database to ignore the rest of the line.

Questions

2. What results do you get from this query?

3. Can you construct a query that will show ALL of the rows in the 'users' table? Write the query down here.

Task 4: Continued exploration

Refer to the section on the DVWA 'SQL Injection' page entitled 'More Information'. Links there will take you to resources that will provide more information about how to take advantage of SQL injection vulnerabilities.

Questions

4. Can you find out more about the field names in the 'users' table? (Take a look at some of the links on DVWA's SQL Injection page for help.)

5. Can you discover more tables in the database used by DVWA?

Set the **DVWA Security** setting to '**medium**'. Which of the above SQL injection attempts still work? To see the difference in the web application between '**low**' and '**medium'** settings, click the'**View Source'** button on the lower-right corner of the **DVWA SQL Injection** page.

6. Which of the above SQL Injection attempts still works in the 'medium' setting?

7. What can the application owner do to fix these SQL injection vulnerabilities?

References

- DVWA Releases on GitHub