Laboratory Exercise – Cyber Basics – Web Application Security: SQL Injection Lab

1. Overview

This laboratory exercise will provide hands-on experience with a particular web application vulnerabilities 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.

2. Resources required

This exercise requires the **Cyber Basics** Exercise Environment running in the Cyber Range.

[Note to instructors: This lab exercise requires an account on the Cyber Range. To sign up for an account, please visit our Sign-Up page. Your students will also require Cyber Range accounts; this will be explained upon course setup.]

For this lab, we will use an intentionally vulnerable web application called DVWA (Damn Vulnerable Web Application, available from http://www.dvwa.co.uk/). DVWA is a teaching tool to help students and system administrators understand common web application flaws that lead to compromise, as well as basic techniques that can be used to help secure these apps.

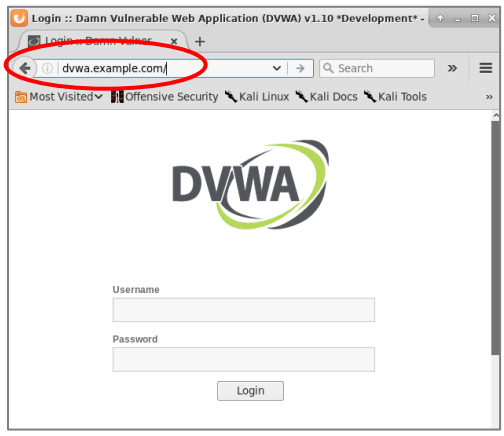
3. Initial Setup

For this exercise you will log in to your Cyber Range account and select the **Cyber Basics** environment. Click "start" to start your environment and "join" to get to your Linux desktop.

4. 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/**



This will take you to the DVWA login screen. Log in to DVWA with these credentials:

Username: **admin**
Password: **password**

Once logged in, click on the **DVWA Security** button on the left side of the page and 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 web sites use databases to store their content. If the web site has a page where a visitor enters search terms or other interactive content, they are often interacting with a database running in 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 is met (in other words, is 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.

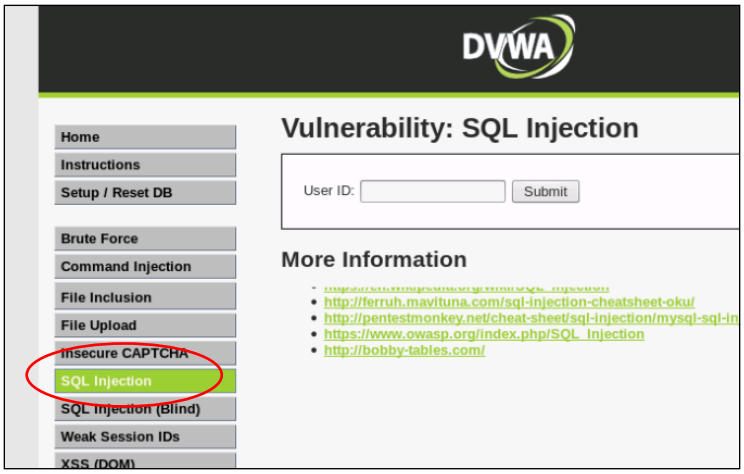
SELECT *last_name* FROM *users* WHERE *first_name = 'Pablo'*

Q1. What is the result of the above SQL query? _____

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 above example, a search criteria of "userid < 3" would return 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 evaluates 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’ # ‘

(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.)

Q2. What results do you get from this query?

Q3. 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.

Q4. 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.)

Q5. 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.

Q6. Which of the above SQL Injection attempts still work in the '**medium**' setting?

Q7. What can the application owner do to fix these SQL injection vulnerabilities?

5. References

- http://www.dvwa.co.uk/

[This portion of the lab exercise template is provided for instructors that will be using this lab in a class they are teaching.]

This exercise makes use of resources provided in the Cyber Range. It is a single Kali Linux virtual machine.

Answer Key

Task 2: SQL Primer

Q1. Picasso

Task 3. Hands on with SQL Injection
Q2. First and last name for rows 1 and 2 in the user table: admin/admin and Gordon/Brown

Q3. 1' or 1=1 #

Task 4: Continued Exploration
Q4. This requires some brute force . . . To identify field names, use the information you know to try to guess them. Example, to guess the 'user' field you might try:

a' OR username = 'admin' #
a' OR users='admin' #

[In this case, you will get an expected response from the second attempt, so you know the name of the first field is 'users', not 'username'. You would then try this with each field in turn.]

Q5. a' UNION SELECT table_schema, table_name FROM information_schema.tables WHERE table_schema = 'dvwa' #

Q6. None of these will probably work on medium setting because the special characters will be stripped (single apostrophes and number sign).

Q7. Input validation. For example, use the php function 'mysqli_real_escape_string' to strip special characters from user input.

KSAs, from NIST SP 800-181: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-181.pdf

Knowledge:

- K0069: Knowledge of query languages such as SQL (structured query language).
- K0070: Knowledge of system and application security threats and vulnerabilities (e.g., buffer overflow, mobile code, cross-site scripting, Procedural Language/Structured Query Language [PL/SQL] and injections, race conditions, covert channel, replay, return-oriented attacks, malicious code).
- K0236: Knowledge of how to utilize Hadoop, Java, Python, SQL, Hive, and PIG to explore data.

Skills:

- S0130: Skill in writing scripts using R, Python, PIG, HIVE, SQL, etc.

Knowledge Units (KUs) Addressed:
From: https://www.iad.gov/NIETP/documents/Requirements/CAE-CD_2019_Knowledge_Units.pdf
(you may need to accept an invalid iag.gov SSL certificate to reach this PDF)

- Databases (DAT)
- Database Management Systems (DMS)
- Basic Scripting and Programming (BSP)