Lab 5 (Milestone of Assignment 1)

Venkata Diwakar Reddy Kashireddy

***Abstract*—This report explains how to perform a blind SQL injection to find out if there are any tables in the database starting with 'A','B','C'. A blind SQL injection is a technique that attackers use to ask the database true or false questions and determines the answer based on the applications response.**

I. Introduction

In this lab, we completed a few tasks on blind SQL injection. We studied blind SQL injection and executed it on WebGoat 8.2.2.

II. TOOLS

• KVM (Kernel-based Virtual Machine): It is a full virtualization solution for Linux. [1]
• WebGoat: A deliberately insecure web application maintained by OWASP designed for teaching web application security concepts. [2]
• Overleaf: It is a collaborative cloud-based LaTeX editor that helps to create documents easily by providing standard formats. [3]
• GitHub: GitHub is an Internet hosting service for software development and version control using Git. [4]
• Red: – IU Research Desktop (RED) is a virtual desktop service for users with accounts on the Carbonate research supercomputer at IU. [5]
• ZAP: OWASP ZAP (Zed Attack Proxy) is an open-source web application security scanner. It is used by those new to application security and professional penetration testers. [6]
• Firefox DevTools: Firefox Developer Tools is a set of web developer tools built into Firefox. It can be accessed to inspect the web page. [7]

III. Blind SQL Injection

Blind SQL injection is a type of SQL injection attacks where the attacker determines database information through true/false questions, based on the application's feedback. This method is used particularly when the application hides specific error messages yet remains open to SQL vulnerabilities.

In this lab, we were tasked with identifying tables in the database that begin with the letters 'A', 'B', and 'C'. The INFORMATION SCHEMA.TABLES view provides details about all tables and views present in a database. By default, it displays data for every table and view contained in the database.[8]

First, I found out which text-box out of both Login and Register page is vulnerable to SQL injection. I found a text box that provided responses — either True or False from database based on the input I provided. For instance, when checking with the "Register" text box, it would indicate whether a user was already registered or not.

Secondly, I formed an true or false question for the database

```
(Select count(table name) from information
schema.tables where table name like 'A%')
> 0;-
```

and this was passed along with the username Tom. So eventually the input looked like

```
Tom'  and (Select count(table
name) from information schema.tables where
table name like 'A%') > 0;-
```

If the server returned that the user already exists, then it indicates that the statement is also true.

Fig. 1. Server indicating the existence of table with name starting with 'A'.

Similarly, inputting the below statement to check if there is a table starting with 'B'.

```
Tom'  and (Select count(table
name) from information schema.tables where
table name like 'B%') > 0;-
```

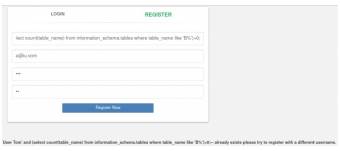

Fig. 2. Server indicating the existence of table with name starting with 'B'.

and for table 'C'.

```
Tom'  and (Select count(table
name) from information schema.tables where
table name like 'C%') > 0;-
```

Hence, we can confirm that table starting with A, B, and C exist in the database.

Fig. 3. Server indicating the existence of table with name starting with 'C'.

IV. Conclusion

SQL injection attacks represents a significant risk to websites reliant on databases. The techniques employed in such attacks are relatively simple to perform and have the potential to compromise a system's security. Notably, these threats can be initiated with ease. However, it is crucial to recognize that they can also be effectively prevented through the application of common sense and proactive measures. Blind SQL injection, a specific type of SQL Injection attack, involves asking true or false queries to the database and deducing the response from the application's feedback. Blind SQL injections are generally more challenging to exploit. Here, we confirmed the existence of tables with names starting 'A', 'B', and 'C' using information schema tables and blind sql injection.

References

[1] KVM
https://www.linux-kvm.org/page/MainPage..
[2] WebGoat
https://owasp.org/www-project-webgoat/..
[3] OverLeaf
https://www.overleaf.com/.
[4] Github
https://en.wikipedia.org/wiki/GitHub.
[5] RED
https://kb.iu.edu/d/apum.
[6] OWASP ZAP
https://www.zaproxy.org/
[7] Firefox DevTools
https://firefox-source-docs.mozilla.org/devtools-user/
[8] Information Schema Tables
https://www.mssqltips.com/sqlservertutorial/196/
information-schema-tables/.