Lab 4

Chetan Sahrudhai Kimidi - ckimidi

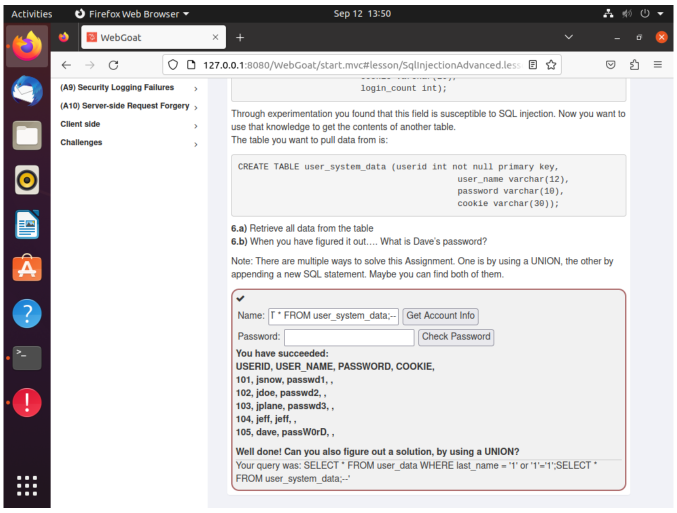
***Abstract*—Working out the SQL Injection advanced section in WebGoat 8, and Blind SQL injection in WebGoat 7.1**

I. Introduction

WEBGOAT, version 8, has a section related to advanced SQL Injection, which contains information and tasks about Blind SQL injection and regarding how to use a combination of SQL Injection techniques, using unions and joins. In WebGoat 7.1, there are tasks about Blind Numeric and String SQL Injection. I have solved these in this lab.

II. WebGoat v2023.4 - SQL Injection (Advanced)

In this lesson, there was information regarding special statements such as unions and joins, and special characters such as ;, - -, # etc.
The first task was to pull data from a table called user_system_data, by using a table called user_data. There were two fields, name and password, and the name field was susceptible to injection. So, using both the approaches of special characters and special statements (unions), I solved this. One of the queries I used is as follows -
select * from user_data where last_name = ’1’ or ’1’=’1’;
select * from user_system_data;- -
I used the double hyphen in the ending, to comment out the single quote which would get appended to the input. This can be seen in Fig. 1.



```
                              login_count int);
```

Through experimentation you found that this field is susceptible to SQL injection. Now you want to use that knowledge to get the contents of another table.
The table you want to pull data from is:

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

6.a) Retrieve all data from the table
6.b) When you have figured it out.... What is Dave's password?

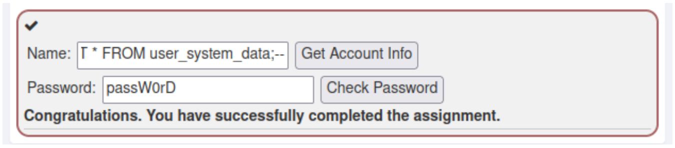
Note: There are multiple ways to solve this Assignment. One is by using a UNION, the other by appending a new SQL statement. Maybe you can find both of them.

Name: T * FROM user_system_data;--
Password:

```
You have succeeded:
USERID, USER_NAME, PASSWORD, COOKIE,
101, jsnow, passwd1, ,
102, jdoe, passwd2, ,
103, jplane, passwd3, ,
104, jeff, jeff, ,
105, dave, passW0rD, ,
Well done! Can you also figure out a solution, by using a UNION?
Your query was: SELECT * FROM user_data WHERE last_name = '1' or '1'='1';SELECT * FROM user_system_data;--'
```

Activities
Firefox Web Browser
Sep 12 13:50
WebGoat
127.0.0.1:8080/WebGoat/start.mvc#lesson/SqlInjectionAdvanced.less
(A9) Security Logging Failures
(A10) Server-side Request Forgery
Client side
Challenges
Get Account Info
Check Password

Fig. 1. SQL Injection Advanced Task



Name: T * FROM user_system_data;--
Password: passW0rD
Congratulations. You have successfully completed the assignment.

Fig. 2. Dave’s password

III. WebGoat 7.1 - Injection Flaws: Blind Injection

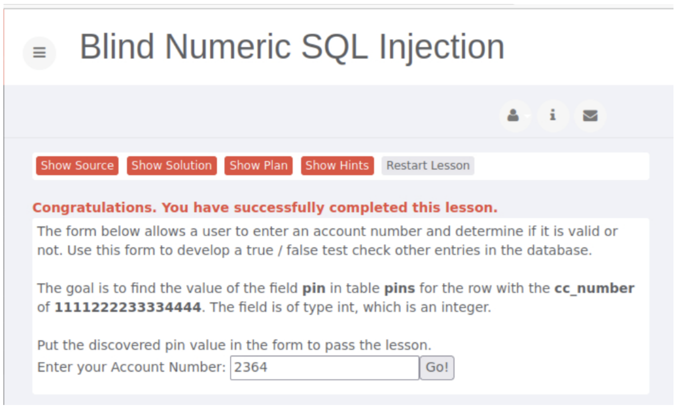
Here, in the given two tasks, I had already completed Blind Numeric SQL Injection, in last week’s lab, as a bonus tasks. This can be seen in Fig. 3. To complete this task, I have used the AND operator along with the account number in the form. My input into the form was like:
101 AND (select pin from pins where cc_number=’1111222233334444’) ’greater than’ X;
I repeatedly queried such a query, changing the values of X, starting from 1000, since it’s a 4 digit pin. And whenever I exceeded the range, the output was account invalid. Based on that, within 5-6 attempts, I found out the pin was 2364, hence solving the task.
The second task was regarding Blind String SQL Injection, where I had to find the value of the field ’name’, in table ’pins’ where cc_number was 4321432143214321. Since I was dealing with a varchar/string, as I learnt in lesson 4 of WebGoat 8 SQL Injection Advanced section, I used the concept of substring(). My input was as follows:
101 and (substring((select name from pins where cc_number=’4321432143214321’),X,1) ’lesser/greater than’ ’A..Z or a..z’);
where X was the Xth letter of the string I wanted to find out. In this way, after manually testing out true/false cases, I found out the name was Jill. In this way, I solved this task, as seen in Fig. 4.

IV. Conclusion

In summary, I successfully exploited various vulnerabilities related to Blind SQL Injection, and run chained SQL commands in susceptible fields.

References

[1] OWASP Blind SQL Injection



Blind Numeric SQL Injection

Congratulations. You have successfully completed this lesson.

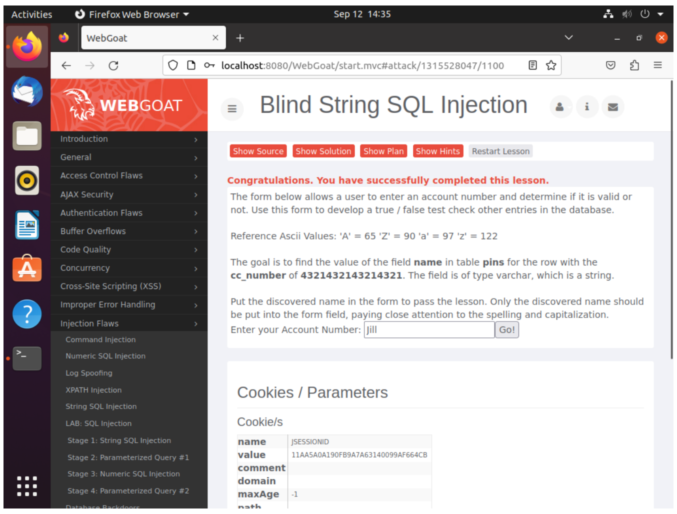
The form below allows a user to enter an account number and determine if it is valid or not. Use this form to develop a true / false test check other entries in the database.

The goal is to find the value of the field **pin** in table **pins** for the row with the **cc_number** of **1111222233334444**. The field is of type int, which is an integer.

Put the discovered pin value in the form to pass the lesson.

Enter your Account Number: 2364 Go!

Fig. 3. Blind Numeric SQL Injection Solved



localhost:8080/WebGoat/start.mvc#attack/1315528047/1100

Blind String SQL Injection

Congratulations. You have successfully completed this lesson.

The form below allows a user to enter an account number and determine if it is valid or not. Use this form to develop a true / false test check other entries in the database.

Reference Ascii Values: 'A' = 65 'Z' = 90 'a' = 97 'z' = 122

The goal is to find the value of the field **name** in table **pins** for the row with the **cc_number** of **4321432143214321**. The field is of type varchar, which is a string.

Put the discovered name in the form to pass the lesson. Only the discovered name should be put into the form field, paying close attention to the spelling and capitalization.

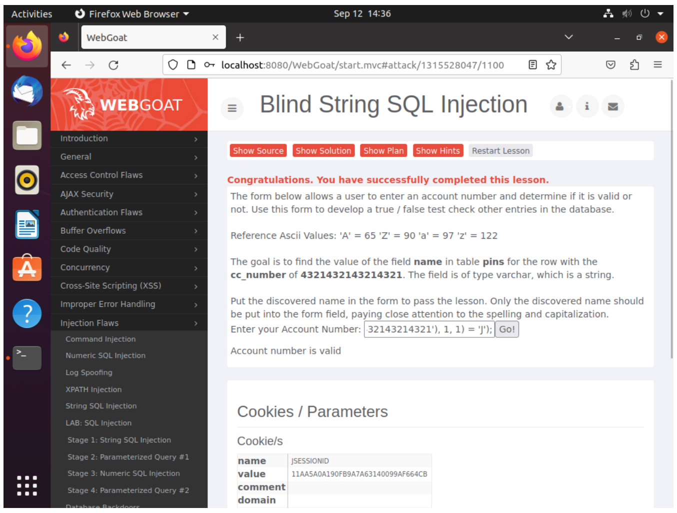
Enter your Account Number: Jill Go!

Cookies / Parameters

Cookie/s

name	JSESSIONID
value	11AA5A0A190FB9A7A63140099AF664CB
comment	
domain	
maxAge	-1

Fig. 4. Blind String SQL Injection Solved



localhost:8080/WebGoat/start.mvc#attack/1315528047/1100

Blind String SQL Injection

Congratulations. You have successfully completed this lesson.

The form below allows a user to enter an account number and determine if it is valid or not. Use this form to develop a true / false test check other entries in the database.

Reference Ascii Values: 'A' = 65 'Z' = 90 'a' = 97 'z' = 122

The goal is to find the value of the field **name** in table **pins** for the row with the **cc_number** of **4321432143214321**. The field is of type varchar, which is a string.

Put the discovered name in the form to pass the lesson. Only the discovered name should be put into the form field, paying close attention to the spelling and capitalization.

Enter your Account Number: 32143214321'), 1, 1) = 'J'); Go!

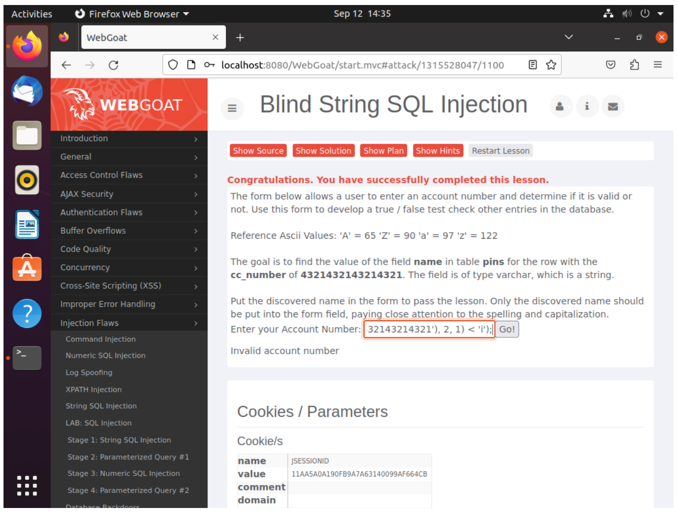
Account number is valid

Cookies / Parameters

Cookie/s

name	JSESSIONID
value	11AA5A0A190FB9A7A63140099AF664CB
comment	
domain	

Fig. 6. Blind String SQL Injection - True Case



localhost:8080/WebGoat/start.mvc#attack/1315528047/1100

Blind String SQL Injection

Congratulations. You have successfully completed this lesson.

The form below allows a user to enter an account number and determine if it is valid or not. Use this form to develop a true / false test check other entries in the database.

Reference Ascii Values: 'A' = 65 'Z' = 90 'a' = 97 'z' = 122

The goal is to find the value of the field **name** in table **pins** for the row with the **cc_number** of **4321432143214321**. The field is of type varchar, which is a string.

Put the discovered name in the form to pass the lesson. Only the discovered name should be put into the form field, paying close attention to the spelling and capitalization.

Enter your Account Number: 32143214321'), 2, 1) < 'i'); Go!

Invalid account number

Cookies / Parameters

Cookie/s

name	JSESSIONID
value	11AA5A0A190FB9A7A63140099AF664CB
comment	
domain	

Fig. 5. Blind String SQL Injection - False Case