SQL 注入攻击

1. 实验概述

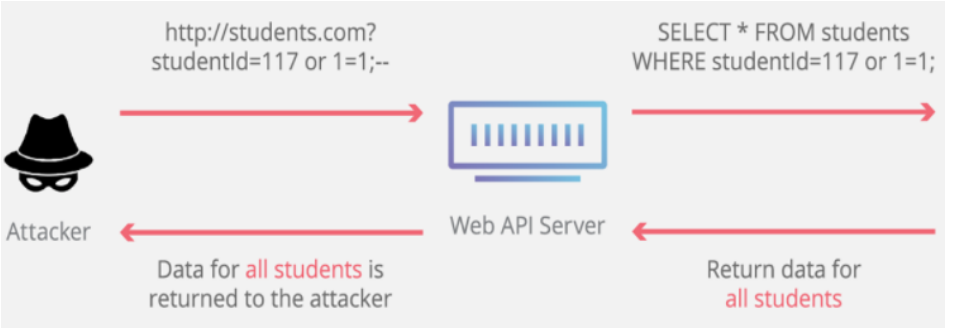
SQL 注入是发生于应用程序与数据库层的安全漏洞。简而言之，是在输入的字符串之中注入 SQL 指令，在设计不良的程序当中忽略了字符检查，那么这些注入进去的恶意指令就会被数据库服务器误认为是正常的 SQL 指令而执行，因此遭到破坏或是入侵。

实验目的：理解 SQL 注入及其问题，掌握利用 SQL 注入攻击数据库的方法。

2. SQL 注入原理

一个典型的 Web 应用包含三个主要部分：Web 浏览器、Web 服务器以及数据库。浏览器运行在客户端，其主要功能是获取 Web 服务器内容，并将其交给用户，与用户进行交付以及获取用户输入。Web 服务器负责生成并传输相应内容给浏览器，它通常依赖一个独立的数据库服务器进行数据管理。浏览器使用 HTTP 协议与 Web 服务器交互，Web 服务器则使用数据库语言（SQL）与数据库交互。

SQL 注入攻击可以破坏数据库，但用户本身并不与数据库直接交互，罪魁祸首是 Web 服务器，它为用户数据传入数据库提供了通道。如果这条通道没有正确搭建，那么恶意用户就可以通过这条通道攻击数据库。



举例来说，下面是一个根据 ID 和密码来获取数据的接口：

http://www.example.com/getdata.php?EID=EID5000&Password=passwd123

这个接口 URL 在 Web 服务器端会被转化为 SQL 语句：

SELECT Name, Salary FROM employee WHERE eid='$id' and password='$pwd'

如果用户在输入数据的时候，在 EID 部分输入了#符号，则可以跳过密码验证，因为#在 SQL 语句中为注释符号：

SELECT Name, Salary FROM employee WHERE eid='EID5000' #' and password='$pwd'

如果用户在输入数据的时候，在 EID 部分输入了 OR 1=1，则可以获取所有数据：

SELECT Name, Salary FROM employee WHERE eid='a' OR 1=1 and password='$pwd'

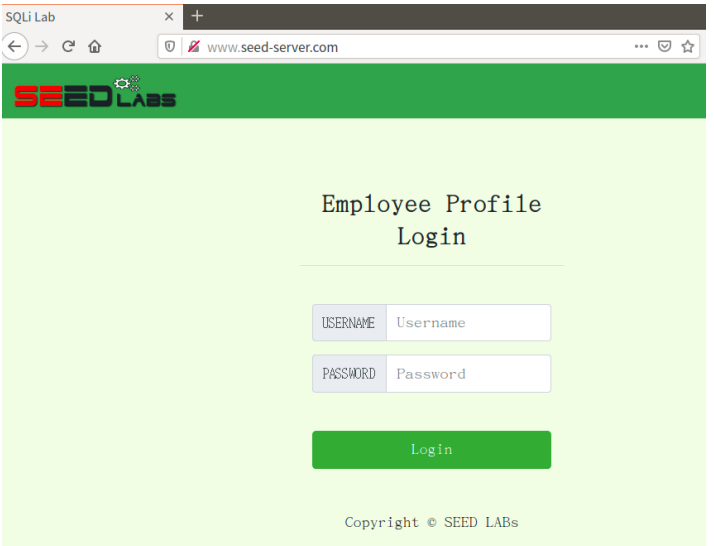
以上便是 SQL 注入攻击的基本原理，本质上来说是用户输入的数据没有经过验证就直接在服务器端执行了。

3. 实验内容

登录到虚拟机后，进入主目录下的 Labsetup 文件夹，启动 docker 容器，使用命令 docker-compose up：

```
osr@osr:~$ ls
Labsetup  公共的  模板  视频  图片  文档  下载  音乐  桌面
osr@osr:~$ cd Labsetup/
osr@osr:~/Labsetup$ sudo docker-compose up
[sudo] osr 的密码:
Creating network "labsetup_net-10.9.0.0" with the default driver
Creating www-10.9.0.5 ...
Creating mysql-10.9.0.6 ...
Creating www-10.9.0.5
Creating mysql-10.9.0.6 ... done
Attaching to www-10.9.0.5, mysql-10.9.0.6
www-10.9.0.5 |  * Starting Apache httpd web server apache2
AH00558: apache2: Could not reliably determine the server's fully
```

成功启动后，打开 Firefox 浏览器，输入网址 www.seed-server.com，可以看到一个叫 Employee 的 Web 应用，是本次实验的攻击目标：



应用内置了以下账户：

Name	Employee ID	Password	Salary	Birthday	SSN
Admin	99999	seedadmin	400000	3/5	43254314
Alice	10000	seedalice	20000	9/20	10211002
Boby	20000	seedboby	50000	4/20	10213352
Ryan	30000	seedryan	90000	4/10	32193525
Samy	40000	seedsamy	40000	1/11	32111111
Ted	50000	seedted	110000	11/3	24343244

3.1 SQL 注入攻击 SELECT 语句

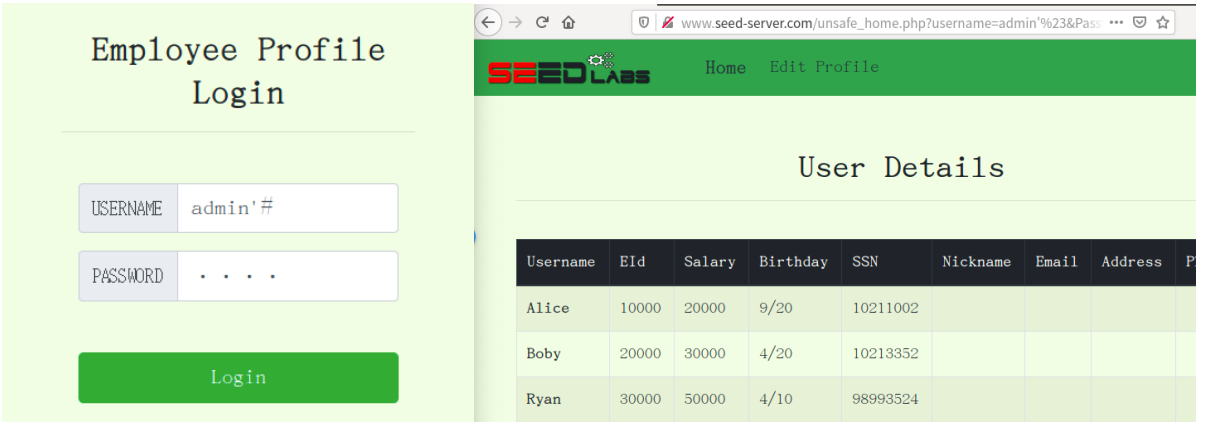
使用 docker exec 进入到 seed-image-www-sqli 的 container，找到目录 /var/www/SQL_Injection 下的 unsafe_home.php 文件，查看其代码：

```
osr@osr:~/Labsetup/image_mysql$ sudo docker ps
[sudo] osr 的密码:
CONTAINER ID        IMAGE                  COMMAND                  CREATED
ORTS                NAMES
6a4d256c4c1f        seed-image-mysql-sqli  "docker-entrypoint.s…"   About an hour ago
306/tcp, 33060/tcp  mysql-10.9.0.6
520309b41847        seed-image-www-sqli    "/bin/sh -c 'service…"   About an hour ago
                    www-10.9.0.5
osr@osr:~/Labsetup/image_mysql$ sudo docker exec -it 520309b41847 /bin/bash
root@520309b41847:/# cd /var/www/SQL_Injection/
root@520309b41847:/var/www/SQL_Injection# ls
css       index.html  seed_logo.png            unsafe_edit_frontend.php
defense   logoff.php  unsafe_edit_backend.php  unsafe_home.php
root@520309b41847:/var/www/SQL_Injection# cat unsafe_
cat: unsafe_: No such file or directory
root@520309b41847:/var/www/SQL_Injection# cat unsafe_home.php
```

可以发现验证用户登录的代码如下，存在 SQL 注入漏洞：

```
$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);
```

在浏览器中打开 Web 应用 www.seed-server.com，用户名输入 admin'#，可以发现无论输入什么密码都可以成功登录管理员用户：



同时可以看到浏览器的 URL 中直接展示了 username=admin’%23，说明登录请求是使用 HTTP GET 方式发送的，可以直接用 curl 进行攻击：

curl http://www.seed-server.com/unsafe_home.php?username=admin%27%23

curl 中特殊字符需要转义（撇号：%27 空格: %20 井号：%23）

3.2 SQL 注入攻击 UPDATE 语句

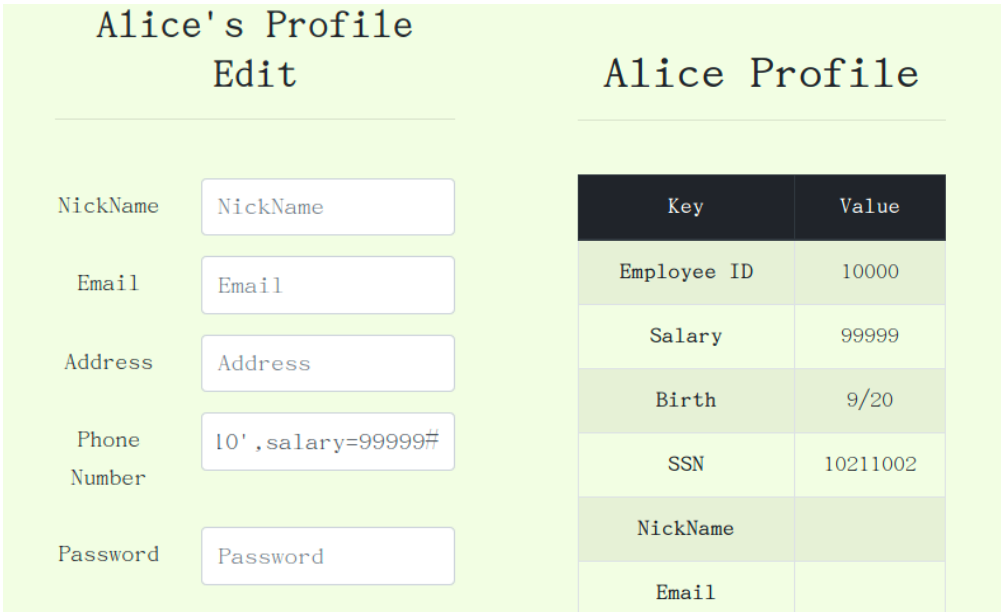
使用 docker exec 进入到 seed-image-www-sqli 的 container，找到目录 /var/www/SQL_Injection 下的 unsafe_edit_backend.php 文件，查看其代码：

```
osr@osr:~/Labsetup/image_mysql$ sudo docker exec -it 520309b41847 /bin/bash
root@520309b41847:/# cd /var/www/SQL_Injection/
root@520309b41847:/var/www/SQL_Injection# ls
css       index.html  seed_logo.png            unsafe_edit_frontend.php
defense   logoff.php  unsafe_edit_backend.php  unsafe_home.php
root@520309b41847:/var/www/SQL_Injection# nano unsafe_edit_backend.php
root@520309b41847:/var/www/SQL_Injection#
```

可以发现后端编辑资料的代码如下，存在 SQL 注入漏洞：

```
$hashed_pwd = sha1($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);
```

在浏览器中打开 Web 应用 www.seed-server.com，使用 alice 的账户进行登录，编辑用户资料。Alice 可以通过 SQL 注入攻击修改自己的工资，只需要在 Phone Number 处输入 10’,salary=99999#



之所以在 Phone Number 处输入是因为 Phone Number 是 SET 字段的最后一部分。Alice 还可以利用 SQL 注入攻击修改其他人的工资，只需要在 Phone Number 处输入 10’,salary=0 WHERE ID=2#

3.3 SQL 注入攻击防御措施

防御 SQL 注入攻击的方法有三种： 1、过滤掉代码 2、通过编译把代码变成数据 3、把代码和数据分开。

过滤掉代码和把代码变成数据是指把特殊符号，#，’等进行过滤，或者编码。比如把撇号进行编码，编码前：aaa' OR 1=1 # 编码后：aaa\' OR 1=1 #

PHP 的 mysqli 有一个内置方法，称为 mysqli::real_escape_string()，可以编码 sql 中的特殊字符。 比如获取用户名时使用：

$username=$mysqli-> real_escape_string($_GET[‘username’])

请用 real_escape_string()方法修改 www.seed-server.com/defense 应用，使用 docker exec 进入到 seed-image-www-sqli 的 container，找到目录/var/www/SQL_Injection/defense 目录下的 unsafe.php 文件，修改其代码：

```
root@520309b41847:/var/www/SQL_Injection# ls
css       index.html  seed_logo.png            unsafe_edit_frontend.php
defense   logoff.php  unsafe_edit_backend.php  unsafe_home.php
root@520309b41847:/var/www/SQL_Injection# cd defense/
root@520309b41847:/var/www/SQL_Injection/defense# ls
getinfo.php  index.html  style_home.css  unsafe.php
root@520309b41847:/var/www/SQL_Injection/defense# nano unsafe.php
```

保存之后，在 www.seed-server.com/defense 应用的用户名输入 admin’#，看看是否可以登录成功。

4. 实验报告提交

请完成上述整个实验内容，将整个实验过程以报告的形式记录，并上传提交到实训教学系统中。