

简介：

所谓 SQL 注入，就是通过把 SQL 命令插入到 Web 表单提交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的 SQL 命令。具体来说，它是利用现有应用程序，将（恶意的）SQL 命令注入到后台数据库引擎执行的能力，它可以通过在 Web 表单中输入（恶意）SQL 语句得到一个存在安全漏洞的网站上的数据库，而不是按照设计者意图去执行 SQL 语句。[1] 比如先前的很多影视网站泄露 VIP 会员密码大多就是通过 WEB 表单递交查询字符暴出的，这类表单特别容易受到 SQL 注入式攻击。

一.环境：

PHP5.6+MySQL

测试：DVWA

对于大多数数据库而言，SQL 注入的原理基本相似，因为每个数据库都遵循一个 SQL 语法标准，但它们之间也存在着许多细微的差异，包括语法、函数不停。所以，在针对不同的数据库注入时，思路、方法也不可能完全一样。

二.注入的目的：

攻击者对数据库注入，无非是利用数据库获得更多的数据或者更大的权限，那么利用方式可以归为以下三类：查询数据、读写文件、执行命令。

在权限允许的情况下，通常数据库都支持以上三种操作。而攻击者对程序注入，无论任何数据库，无非都是在做这三件事，只不过不同的数据库注入的 SQL 语句不一样罢了。

三.PHP mysql 和 mysqli 函数：

（一）不同点：

设计开发允许 PHP 应用与 MySQL 数据库交互的早期扩展。mysql 扩展提供了一个面向过程的接口；并且是针对 MySQL4.1.3 或更早版本设计的。因此，这个扩展虽然可以与 MySQL4.1.3 或更新的数据库服务端进行交互，但并不支持后期 MySQL 服务端提供的一些特性。

PHP 的 mysqli 扩展：mysqli 扩展，我们有时称之为 MySQL 增强扩展，可以用于使用 MySQL4.1.3 或更新版本中新的高级特性；mysqli 扩展在 PHP 5 及以后版本中包含；mysqli 扩展有一系列的优势，相对于 mysql 扩展的提升主要有：面向对象接口、prepared 语句支持、多语句执行支持、事务支持、增强的调试能力、嵌入式服务支持。mysqli 是 php5 提供的新函数库，(i)表示改进，其执行速度更快.当然也更安全。

mysql 是非持续连接函数而 mysqli 是永远连接函数。也就是说 mysql 每次链接都会打开一个连接的进程而 mysqli 多次运行 mysqli 将使用同一连接进程,从而减少了服务器的开销。

（二）MySQLi 主要函数：

1.mysql_connect()函数：用于打开一个到 MySQL 服务器的新的连接

mysql_connect(host,username,password,dbname,port,socket);

参数：

host 可选，规定主机名或 ip 地址；username 可选，规定 MySQL 用户名；

password 可选，规定 MySQL 密码；dbname 可选，规定默认使用的数据库；

port 可选，规定禅师连接到 MySQL 服务器的端口；

socket 可选，规定 socket 或要使用的以命名的 pipe；

返回值：返回一个代表 MySQL 服务器的连接的对象

实例：`$con=mysqli_connect("localhost","wrong_user","my_password","my_db");`

2.`mysqli_error()`函数：返回最近调用函数的最后一个错误描述。

`mysqli_error(connection);`

参数：`connection` 必需,规定要使用的 MySQL 连接。

返回值：返回一个带有错误描述的字符串。如果没有错误发生则返回“”。

3.`mysqli_fetch_row()`函数：从结果集中取得一行，并作为枚举数组返回。

`mysqli_fetch_row(result);`

参数：`result` 必需，规定由 `mysqli_query()`、`mysqli_store_result()` 或 `mysqli_use_result()` 返回的结果集标识符。

返回值：返回一个与所取得行相对应的字符串数组。如果在结果集中没有更多的行则返回 `NULL`。

4.`mysqli_query()`函数：执行某个针对数据库的查询。

`mysqli_query(connection,query,resultmode);`

参数：

`connection` 必需，规定要使用的 MySQL 连接。

`query` 必需，规定查询字符串。

`resultmode` 可选。一个常量。可以是下列值中的任意一个：

`MYSQLI_USE_RESULT`（如果需要检索大量数据，请使用这个）

`MYSQLI_STORE_RESULT`（默认）

返回值：

针对成功的 `SELECT`、`SHOW`、`DESCRIBE` 或 `EXPLAIN` 查询，将返回一个 `mysqli_result` 对象。针对其他成功的查询，将返回 `TRUE`。如果失败，则返回 `FALSE`。

5. `mysqli_select_db()` 函数:用于更改连接的默认数据库。

`mysqli_select_db(connection,dbname);`

参数：

`connection` 必需。规定要使用的 MySQL 连接。

`dbname` 必需，规定要使用的默认数据库。

返回值：如果成功则返回 `TRUE`，如果失败则返回 `FALSE`。

6. `mysqli_set_charset()` 函数 规定当与数据库服务器进行数据传送时要使用的默认字符集。

`mysqli_set_charset(connection,charset);`

参数：

`connection` 必需。规定要使用的 MySQL 连接。

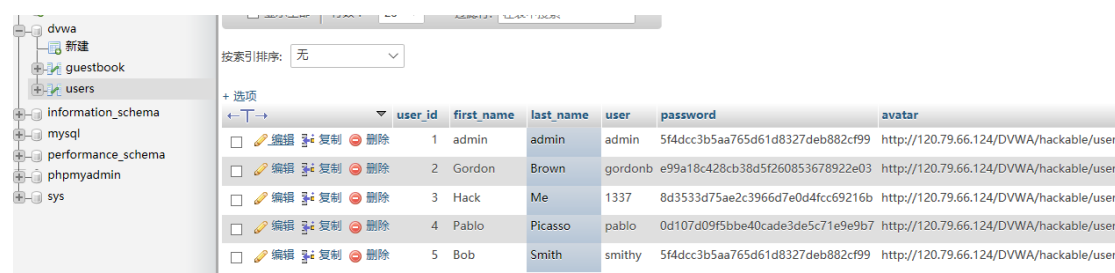
`charset` 必需。规定默认字符集。

返回值：如果成功则返回 `TRUE`，如果失败则返回 `FALSE`。

四.注入漏洞与数据库分析：

（一）数据库创建：

在 phpMyAdmin 中创建好库、表、字段和相应数据，这里我直接使用 DVWA 的数据库



The screenshot shows the phpMyAdmin interface. On the left, the database structure is listed: dvwa, guestbook, users, information_schema, mysql, performance_schema, phpmyadmin, and sys. The 'users' table is selected, and its data is displayed in a table format. The table has columns: user_id, first_name, last_name, user, password, and avatar. There are five rows of data.

	user_id	first_name	last_name	user	password	avatar
<input type="checkbox"/>	1	admin	admin	admin	5f4dcc3b5aa765d61d8327deb882cf99	http://120.79.66.124/DVWA/hackable/user
<input type="checkbox"/>	2	Gordon	Brown	gordonb	e99a18c428cb38d5f260853678922e03	http://120.79.66.124/DVWA/hackable/user
<input type="checkbox"/>	3	Hack	Me	1337	8d3533d75ae2c3966d7e0d4fcc69216b	http://120.79.66.124/DVWA/hackable/user
<input type="checkbox"/>	4	Pablo	Picasso	pablo	0d107d09f5bbe40cade3de5c71e9e9b7	http://120.79.66.124/DVWA/hackable/user
<input type="checkbox"/>	5	Bob	Smith	smithy	5f4dcc3b5aa765d61d8327deb882cf99	http://120.79.66.124/DVWA/hackable/user

（二）漏洞源码和分析：

```
<!DOCTYPE HTML>
<html>
<meta charset="utf-8">
<body>
<form method="post">
    <label>User ID:<input type="text" name="id"></label>
    <button type="submit" name="submit">Submit</button>
</form>

<?php
    if(isset($_POST["submit"])){
        $conn=mysqli_connect("localhost:3306","root","root","dvwa");
        $id=$_POST["id"];
        $sql="select first_name,last_name from users where user_id='$id'";
        $result=mysqli_query($conn,$sql);
        $row=mysqli_fetch_row($result);
        echo "<br/>ID:$id<br/>First name: $row[0]<br/>Last name: $row[1]<br/>";
    }
    mysqli_close($conn);
?>
</body>
</html>
```

分析：

html 部分构造了一个 post 提交的表单，表单信息又两部分，一个是查询的 id，一个是 submit。php 部分 if 判断是否获得 post 的 submit 的值，如果获得了说明表单信息已经提交了，于是连接数据库，将 post 上来的 id 值赋值给变量 id，构造查询语句，将查询结果赋值给 result 变量，将变量 result 的值划分成数组赋值给 row，输出查询结果。

（三）information_schema 库

在 mysql 数据库里有一个很重要的库 information_schema 库，这个库用来存放 mysql 数据库的重要信息，其中有两个表 tables 和 column，这两个表中存放了 mysql 数据库中所有库的表名和字段名

TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	TABLE_TYPE	ENGINE	VERSION	ROW_FORMAT	TABLE R
def	information_schema	CHARACTER_SETS	SYSTEM VIEW	MEMORY	10	Fixed	
def	information_schema	COLLATIONS	SYSTEM VIEW	MEMORY	10	Fixed	
def	information_schema	COLLATION_CHARACTER_SET_APPLICABILITY	SYSTEM VIEW	MEMORY	10	Fixed	
def	information_schema	COLUMNS	SYSTEM VIEW	InnoDB	10	Dynamic	
def	information_schema	COLUMN_PRIVILEGES	SYSTEM VIEW	MEMORY	10	Fixed	
def	information_schema	ENGINES	SYSTEM VIEW	MEMORY	10	Fixed	
def	information_schema	EVENTS	SYSTEM VIEW	InnoDB	10	Dynamic	
def	information_schema	FILES	SYSTEM VIEW	MEMORY	10	Fixed	
def	information_schema	GLOBAL_STATUS	SYSTEM VIEW	MEMORY	10	Fixed	
def	information_schema	GLOBAL_VARIABLES	SYSTEM VIEW	MEMORY	10	Fixed	
def	information_schema	KEY_COLUMN_USAGE	SYSTEM VIEW	MEMORY	10	Fixed	
def	information_schema	OPTIMIZER_TRACE	SYSTEM VIEW	InnoDB	10	Dynamic	
def	information_schema	PARAMETERS	SYSTEM VIEW	InnoDB	10	Dynamic	

TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME
def	information_schema	CHARACTER_SETS	CHARACTER_SET_NAME
def	information_schema	CHARACTER_SETS	DEFAULT_COLLATE_NAME
def	information_schema	CHARACTER_SETS	DESCRIPTION
def	information_schema	CHARACTER_SETS	MAXLEN
def	information_schema	COLLATIONS	COLLATION_NAME
def	information_schema	COLLATIONS	CHARACTER_SET_NAME
def	information_schema	COLLATIONS	ID
def	information_schema	COLLATIONS	IS_DEFAULT
def	information_schema	COLLATIONS	IS_COMPILED
def	information_schema	COLLATIONS	SORTLEN
def	information_schema	COLLATION_CHARACTER_SET_APPLICABILITY	COLLATION_NAME
def	information_schema	COLLATION_CHARACTER_SET_APPLICABILITY	CHARACTER_SET_NAME
def	information_schema	COLUMNS	TABLE_CATALOG
def	information_schema	COLUMNS	TABLE_SCHEMA
def	information_schema	COLUMNS	TABLE_NAME
def	information_schema	COLUMNS	COLUMN_NAME

而我们注入需要重点利用这个库和这两个表。

(四) 注入一般步骤：

1. 找寻注入点和确定闭合方式：

←
→
↻
120.79.66.124/SQL/SQL1.php

User ID:

ID:1

First name: admin

Last name: admin

从这个界面来看，可以通过输入 User ID 来查询，那么这里可能就存在注入漏洞。输入 1'# 1' and 1=1# 和 1' and 1=2# 试试

User ID:

ID:1'#

First name: admin

Last name: admin

User ID:

ID:1' and 1=1#

First name: admin

Last name: admin

User ID:

ID:1' and 1=2#

First name:

Last name:

为什么会有这样的结果，从源码分析来看，它的查询语句是这一段：

```
select first_name,last_name from users where user_id='$id';
```

而\$id 变量储存的是查询内容且用单引号引起，当我们输入 1'# 时查询语句就变成了

```
select first_name,last_name from users where user_id='1'#;
```

号把后面的注释掉了，查询语句不变，同理当我们输入 1' and 1=1 # 时查询语句变成

```
select first_name,last_name from users where user_id='1' and 1=1#';
```

注释掉了后面的内容但 and 1=1 被我们插入到了注入语句中且在页面被执行了，1 显然等于 1，所以查询语句正常执行。而当我们输入 1' and 1=2