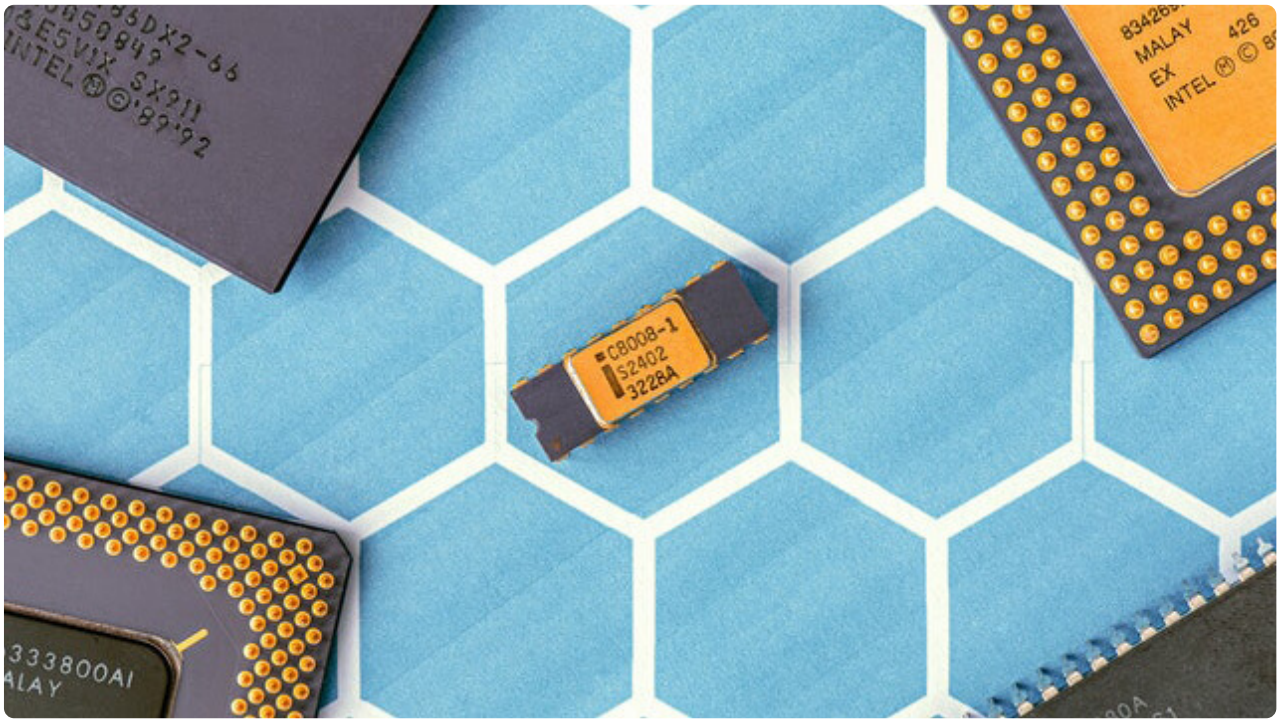
24 如何预防SQL注入?

更新时间：2019-10-03 12:04:09



> 不想当将军的士兵，不是好士兵。
>
> ——拿破仑

很多时候，我们关心的是程序是否能实现我们的预期功能，而忽视了 SQL 安全，这就存在 SQL 注入的风险，本节就一起来聊聊 SQL 注入及防御。

1 认识 SQL 注入

SQL 注入是利用某些数据库的外部接口将用户数据插入到实际的数据库操作语句中，从而达到入侵数据库乃至操作系统的目的。

SQL 注入产生的主要原因是：程序对用户输入的数据没有进行严格的过滤，导致非法数据库查询语句的执行。

SQL 注入具有很大的危害，可能会导致攻击者非法入侵系统，或者盗取数据，甚至清空数据等。

2 如何进行 SQL 注入攻击

为了方便理解 SQL 注入，我们来用实验模拟一下 SQL 注入攻击。

我们模拟一个用户登录验证的场景，首先创建用户表，并录入几个用户，语句如下：

```
drop table if exists t24;
CREATE TABLE `t24` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(20) DEFAULT NULL,
  `password` varchar(20) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB CHARSET=utf8mb4;

insert into t24(username,password) values ('mt','aaa'),('gg','bbb'),('mm','ccc');
```

验证用户登录的 PHP 代码（PHP 版本：7.0.33）：

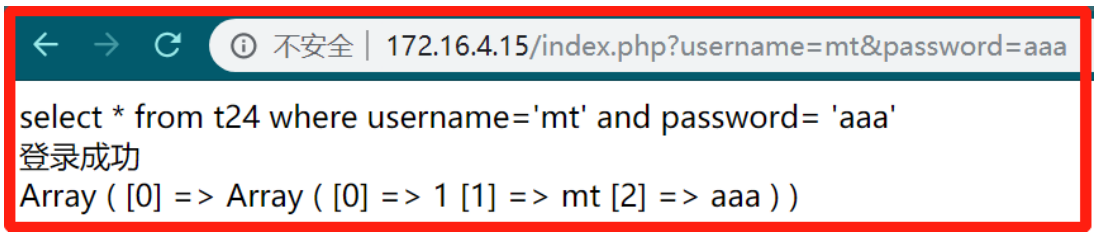
文件名：index.php

内容如下：

```
<?php
$servername="127.0.0.1";
$dbuser="muke_user";
$dbpassword="9Gcag71Gaa";
$dbname="muke";

$mysqli = new mysqli($servername,$dbuser,$dbpassword,$dbname); //配置MySQL连接

if($mysqli->connect_error){
	die('connect error:'.$mysqli->connect_errno);
}
$mysqli->set_charset('UTF-8'); // 设置数据库字符集
$username = isset($_GET['username']) ? $_GET['username'] : '';
$password = isset($_GET['password']) ? $_GET['password'] : '';
$sql = "select * from t24 where username='$username' and password= '$password'";
echo "$sql<br/>";
$result = $mysqli->query("$sql");

$data = $result->fetch_all(); // 从结果集中获取所有数据
if (empty($data))
{
	echo "登录失败";
} else {
	echo "登录成功";
}
echo "<br/>";
print_r($data);
?>
```

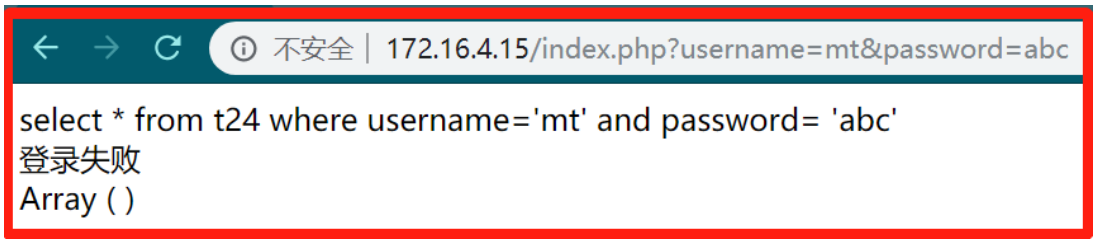
正常情况下，我们提交如下 URL，如果用户名密码正确的话，就可以使用 mt 这个用户登录系统了：

```
http://172.16.4.15/index.php?username=mt&password=aaa
```



如果不知道密码情况下乱输入一个密码，将无法登录通过验证：

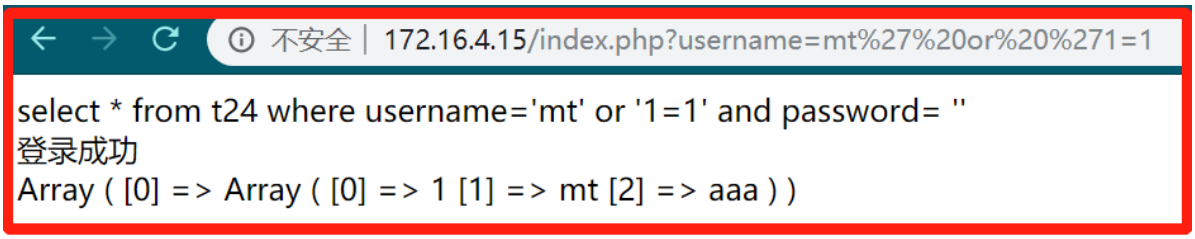
```
http://172.16.4.15/index.php?username=mt&password=abc
```



但是可以通过 SQL 注入的方式，让攻击者不知道密码的情况也可以通过验证，如下：

把 URL 改成：

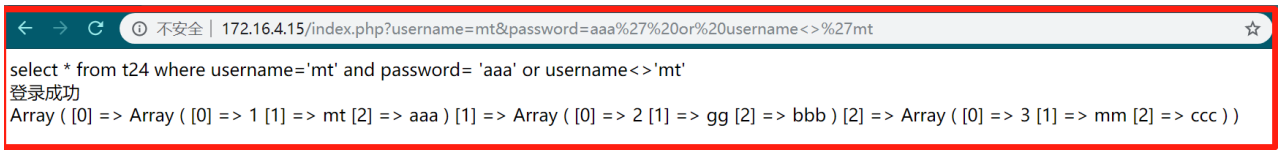
```
http://172.16.4.15/index.php?username=mt' or '1=1
```



显然我们不希望能通过这种方式登录系统，这就是一个 SQL 注入的例子。

这个例子中，甚至可以通过 SQL 注入捞取到所有用户信息，如下 URL：

```
http://172.16.4.15/index.php?username=mt&password=aaa' or username<>'mt
```



从上面截图中可以看到，URL 获取到表里所有用户的信息了。如果是生产环境，用户表有这种漏洞，可想而知他的危害有多大。

3 如何预防 SQL 注入

3.1 控制输入变量的格式

如上面例子中的 PHP 代码，可以优化成如下效果：

```php
<?php
$servername="127.0.0.1";
$dbuser="muke_user";
$dbpassword="9Gcag71Gaa";
$dbname="muke";

$mysqli = new mysqli($servername,$dbuser,$dbpassword,$dbname); //配置MySQL连接

if($mysqli->connect_error){
    die('connect error:'.$mysqli->connect_errno);
}
$mysqli->set_charset('UTF-8'); // 设置数据库字符集
$username = isset($_GET['username']) ? $_GET['username'] : '';
$password = isset($_GET['password']) ? $_GET['password'] : '';

//增加对输入用户名密码的判断，如果不是字母或者数字，就直接提示格式错误而退出。
if( !preg_match("/^[a-zA-Z0-9]{1,}$/",$username) || !preg_match("/^[a-zA-Z0-9]{1,}$/",$password) ) {
    die("You input username and password  format error ");
}

$sql = "select * from t24 where username='$username' and password= '$password'";
echo "$sql<br/>";
$result = $mysqli->query("$sql");

$data = $result->fetch_all(); // 从结果集中获取所有数据
if (empty($data))
{
    echo "登录失败";
} else {
    echo "登录成功";
}
echo "<br/>";
print_r($data);
?>
```

增加了对输入信息的判断，过滤掉一些带特殊字符的输入，我们再验证一下是否还会出现上面我们测试的 SQL 注入情况：

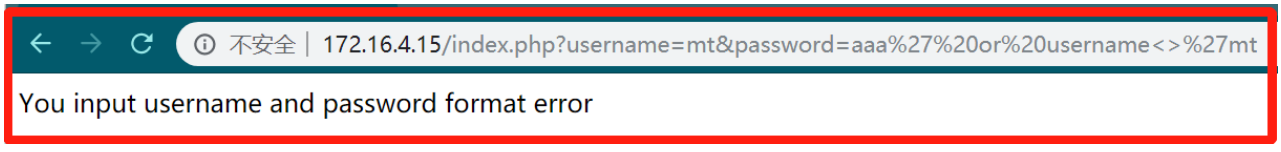
首先看这个 URL：

```
http://172.16.4.15/index.php?username=mt' or '1=1
```

发现会提示输入格式有问题，无法登录系统。

在看下上面能获取整张用户表数据的 SQL 注入：

```
http://172.16.4.15/index.php?username=mt&password=aaa' or username<>'mt
```



发现同样提示输入格式有问题，无法查询到任何数据。

3.2 转义特殊字符

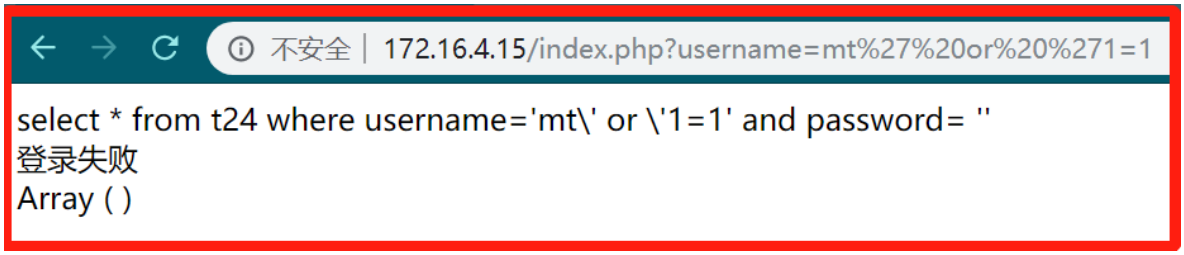
有时，我们程序是允许输入信息带特殊字符的，这种情况就可以使用转义的方式，防止 SQL 注入。

我们来实验下，在原始代码中，修改定义 sql 这一行为：

```
$sql = "select * from t24 where username='" . addslashes($username) . "' and password= '" . addslashes($password) . "'";
```

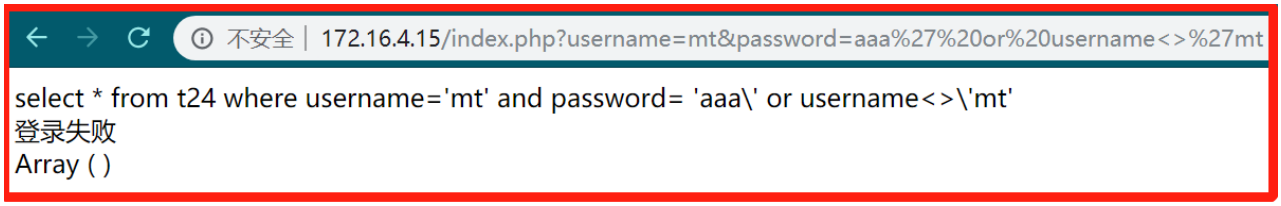
我们再输入前面会导致 SQL 注入的 URL：

```
http://172.16.4.15/index.php?username=mt' or '1=1
```



将传输的变量转义后，避免了 SQL 注入，发现登录失败。

```
http://172.16.4.15/index.php?username=mt&password=aaa' or username<>'mt
```



转义后，无法查询到整张用户表的信息了。

4 总结

本节讲解了 SQL 注入。

产生的原因：程序对用户输入的数据没有进行严格的过滤，导致攻击者增加一些特殊字符，从而改变传输到 MySQL 中的 SQL。

SQL 注入危害：会导致攻击者非法入侵系统，或者盗取数据，甚至清空数据等。

本节讲解了 SQL 注入攻击的方法，目的是让我们能认识到会导致 SQL 注入的一些漏洞，从而优化我们的代码，比如控制输入变量的格式或者转义特殊字符，从而避免程序被 SQL 注入。

5 问题

你写的程序是怎样预防 SQL 注入漏洞的？

6 参考资料

《深入浅出 MySQL》第 2 版 第 15 章 SQL 中的安全问题

}

23 细聊分布式事务

25 主键是否需要设置为自增?