漏洞演示平台之 SQL injection 篇

一、 什么是 SQL 注入

SQL 注入是指：应用程序在向后台数据库传递 SQL（Structured Query Language，结构化查询语言）查询时，如果为攻击者提供了影响该查询的能力，则会引发 SQL 注入。攻击者通过影响传递给数据库的内容来修改 SQL 自身的语法和功能，并且会影响 SQL 所支持数据库和操作系统的功能和灵活性。SQL 注入不只是一种会影响 Web 应用的漏洞；对于任何从不可信源获取输入的代码来说，如果使用了该输入来构造动态 SQL 语句，那么就很可能也会受到攻击。

二、 SQL 注入攻击的危害

1、非法操作数据库中的数据。

2、恶意篡改网页内容。登录后台后发布的内容，也可以发布对首页的更新，更新一些非法信息，对系统进行添加账户或者是数据库账号。

3、私自添加系统帐号或者是数据库使用者帐号。

4、网页挂木马。

三、 SQL 注入漏洞存在的原因

应用的开发人员操作不规范，构造了未经处理的动态字符串，产生了不当的转义字符处理、类型处理、查询集处理、错误处理等，并没有正确地配置数据库，导致攻击者能够控制发送给 SQL 查询的输入，并将该输入解析为代码而非数据。

四、 设计目标

让使用者能去了解 SQL Injection 漏洞，通过了解 SQL 注入的原理，掌握 SQL 注入漏洞的利用技巧。SQL 注入有许多防御方式，但几乎没有完全保证可以百分百阻止注入的方式，在本作品中的输入栏输入 SQL 注入的代码，只要能在结果页面显示注入成功，即可算完成 SQL 注入攻击。

五、 作品详解

［1］第一关——未设置任何防御手段

［2］第二关——过滤关键字：

使用过滤关键字（or,select,union等SQL中的关键字）的方法防止SQL注入，但因为仅过滤小写字符，可采用OR，SELECT等方式，大写的SQL仍是合法的。具体关键代码如下：

```
if (preg_match("/^.*((union)|(select)|and|or)+.*$/",$username))
    {

$username=preg_replace('/(union)|(select)|and|or/','',$username);
    }
```

［3］第三关——大小写关键字过滤：

采用大小写不敏感的方式过滤了用户输入的SQL关键字：

```
if (preg_match("/^.*((union)|(select)|and|or)+.*$/i",$username))
{
    $username=preg_replace('/(union)|(select)|and|or/i','',$username);
}
```

［4］第四关——循环过滤关键字：

采用循环过滤的方式确保过滤了用户输入中所有关键字：

```
while (preg_match("/^.*((union)|(select)|and|or)+.*$/i",$username))
{
    $username=preg_replace('/(union)|(select)|and|or/i','',$username);
}
```

［5］第五关——过滤空格：

使用过滤空格的方式防止SQL注入：

```
while(preg_match("/^.*((union)|(select)|and|or| )+.*$/i",$username))
{
    $username=preg_replace('/(union)|(select)|and|or| /i','',$username);
}
```

［6］第六关——使用了转义某些特殊符号（如单引号"\'"、反斜杠"\\"）的方式，在每个特殊符号前自动添加转义符号"\"从而实现对这类特殊符号的“过滤”：

```
    $username=addslashes($username);
    $password=addslashes($password);
```

［7］第七关——参数化语句：

采用了SQL参数化查询的方式防止注入，这是目前较为有效的防止注入的方法：

```
$stmt = $db->prepare("select * from users where username=? and
password=?");
$stmt->bind_param('ss', $username,$password);

$stmt->execute();
```