**实验二实验报告**

口令破击

1. 搭建好网络攻防测试环境，配置好`Xampp`，把 软件放到`'xampp\htdocs'`文件夹下

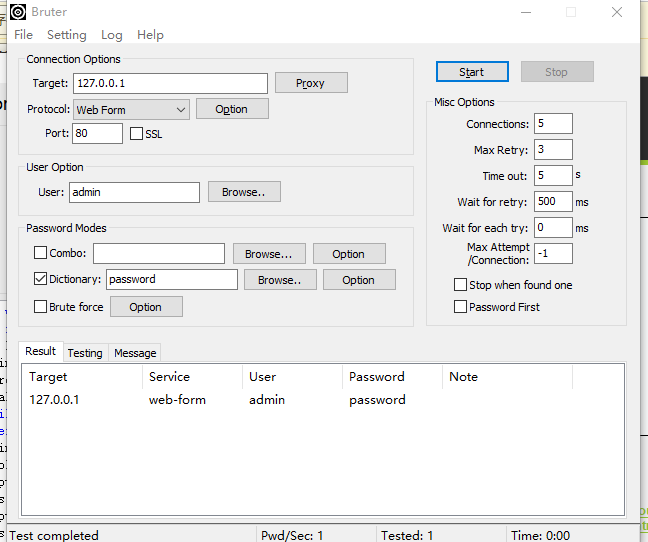
2. 启动`apache`和`mysql`，并且打开本地主机地址`http://localhost/DVWA/`

3. 打开暴力破解软件`bruter`设置`Target`和`protocol`以及网页地址，端口号80

4. 选择密码模式为字典攻击，并导入根据模板实例改写的字典信息

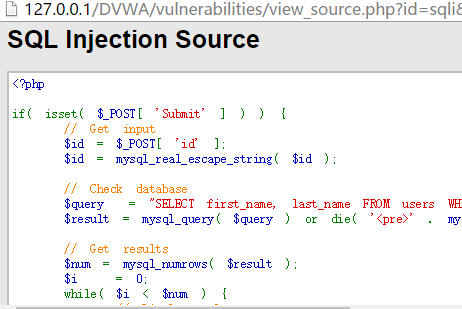
5. 启动`bruter`,运行结果如下图

这个实验

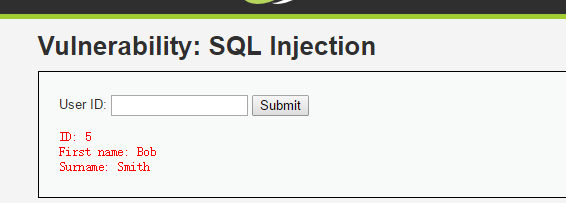
****

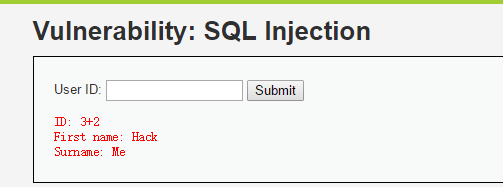
SQL Injection

首先还是按照上面的配置好环境，登陆对应页面，进入之后，Security level设为low，在左侧列表中选择“SQL Injection”，然后在右侧的“User ID”文本框中输入不同的数字就会显示相应的用户信息。

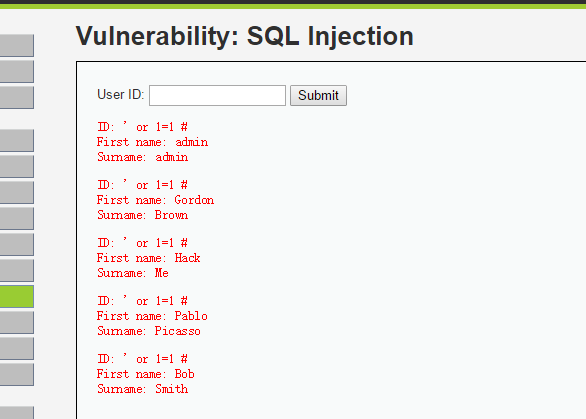


首先需要判断这里所传输的参数是文本型还是数字型，分别输入5和3+2，观察它们显示的结果并不一致，因而就判断出参数类型是文本型。

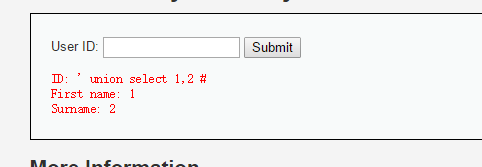




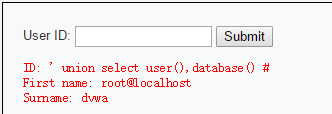
在文本框中输入“' or 1=1 #”就可以显示users表中的所有数据：



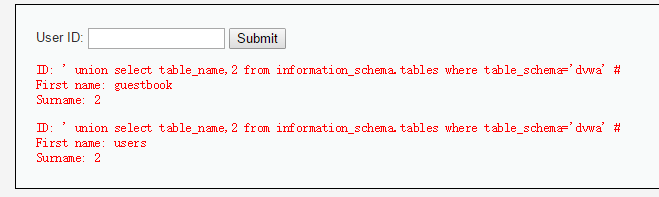
输入“' union select 1,2 #”判断出字段数量为2，并且两个字段都可以代入参数进行查询。



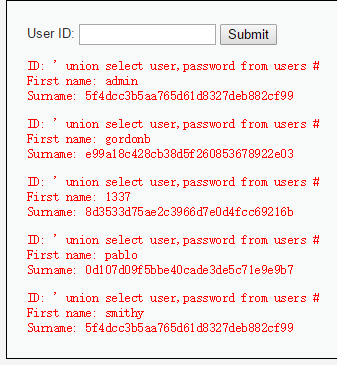
输入“' union select user(),database() #”检测出MySQL版本当前[数据库](http://www.2cto.com/database/" \t "http://www.2cto.com/Article/201512/_blank)名



输入“' union select table\_name,2 from information\_schema.tables where table\_schema='dvwa' #”爆出dvwa数据库中有两个表：guestbook、users，其中我们关心的很明显是users。

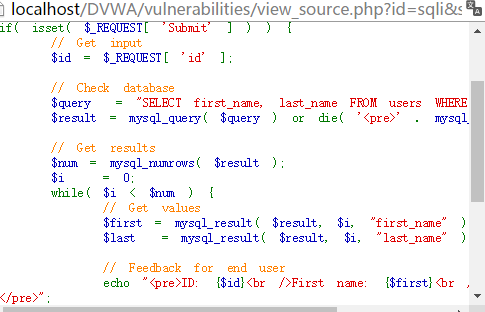


输入“' union select user,password from users #”，爆出用户名和密码。



**SQL Injection （medium 级别）**

将“DVWA Security”设置为medium中等级别，首先点击右下角的“View Source”查看此时的网页源码，主要观察与low级别的区别。



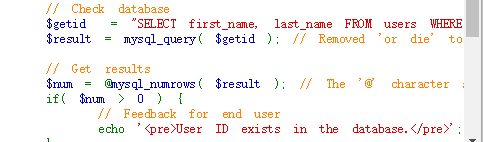
我们再仔细观察一下源码，可以发现参数id已经被改为了数字型，第三行语句中“user\_id = $id”，而之前的low级别是“user\_id = ‘$id’”，其实这就是DVWA故意留下的一个[漏洞](http://www.2cto.com/" \t "http://www.2cto.com/Article/201512/_blank)。在文本框中输入3以及1+2，发现两者显示的结果都是一样的，因而可以断定参数是数字型，这里就应该采用数字型的注入方法。

首先输入“1 and 1=1”，显示正常；输入“1 and 1=2”，显示不正常。判断存在注入点。  
分别输入“1 order by 1”、“1 order by 2”，显示正常；输入“1 order by 3”，显示错误，判断存在2个字段。  
输入“1 union select 1,2”判断出两个字段都可以代入参数进行查询。  
再接下来的操作就与之前相同了。所以不再赘述。

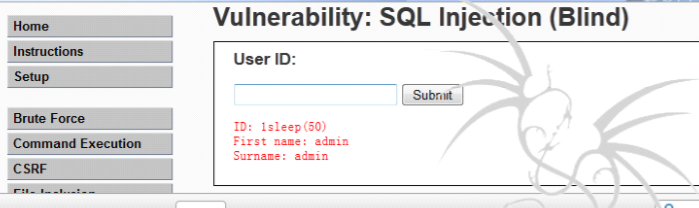
SQL Injection (Blind)

盲注就是指当我们输入一些特殊字符时，页面并不显示错误提示，这样我们只能通过页面是否正常显示来进行判断。

DVWA Security设置为low，然后选择SQL Injection (Blind)查看网页[源码](http://www.2cto.com/ym" \t "http://www.2cto.com/Article/201512/_blank)。可以发现与之前不同的是，在[mysql](http://www.2cto.com/database/MySQL/" \t "http://www.2cto.com/Article/201512/_blank)\_numrows()函数之前多加了一个@符号，后面的注释说明@符号可以抑制报错信息



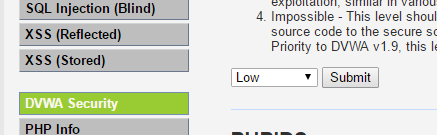
sleep()测试



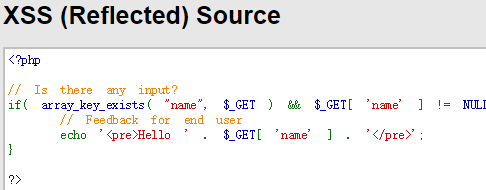
盲注其实对渗透并没有太大影响，我们输入“’ or 1=1 #”仍然可以显示出所有的数据。整个渗透过程也与之前基本一致。

**Reflected Cross Site Scripting (XSS)**

选择xss(Reflected)安全级别仍然为low



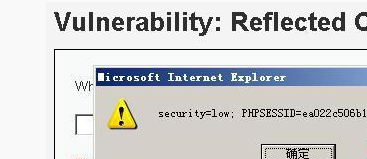
low级别的网页代码，可以看到这里对用于接收用户数据的name参数没有进行任何过滤，就直接在网页中输出，因而造成了XSS[漏洞](http://www.2cto.com/" \t "http://www.2cto.com/Article/201601/_blank)。



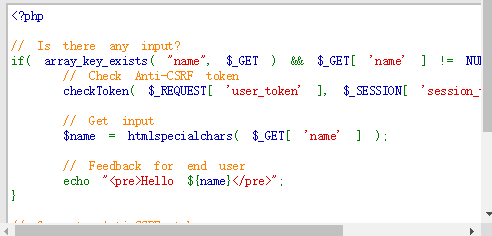
输入一段最基本的XSS语句来实现弹框：“alert(‘hi’)”：



输入一段语句来弹出cookie：“alert(document.cookie)”。

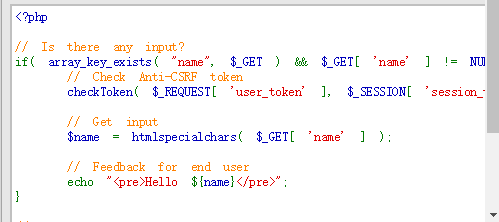


再查看medium级别的XSS[源码](http://www.2cto.com/ym" \t "http://www.2cto.com/Article/201601/_blank)，这里在输出name参数中的数据之前，先利用str\_replace()函数进行了处理，处理的目的是将替换成空值。



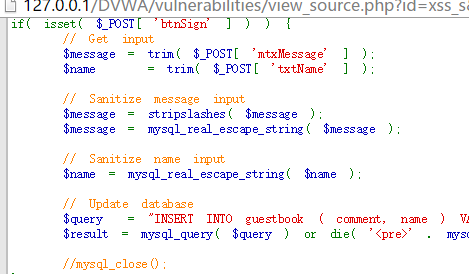
我们只要使用大写的就可以绕过，其次XSS也并非只能使用[Java](http://www.2cto.com/kf/ware/Java/" \t "http://www.2cto.com/Article/201601/_blank)Script，通过HTML语句也可以跨站，比如“”语句。是[html](http://www.2cto.com/kf/qianduan/css/" \t "http://www.2cto.com/Article/201601/_blank)中的图片标记，这条语句表示在网页中插入一张图片，“src=1”指定了图片文件的URL，如果图片不存在（这里肯定是不存在了），那么将会弹出错误提示框，从而实现弹框的效果。

最后再来查看high级别的源码，这里利用了htmlspecialchars()函数进行过滤。这个函数可以把& （和号）、"（双引号）、'（单引号）、（大于）这些敏感符号都进行转义，所有的跨站语句中基本都离不开这些符号，因而只需要这一个函数就阻止了XSS漏洞，所以跨站漏洞的代码防御还是比较简单的。

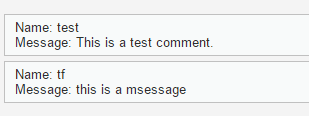


**Stored Cross Site Scripting (XSS）**

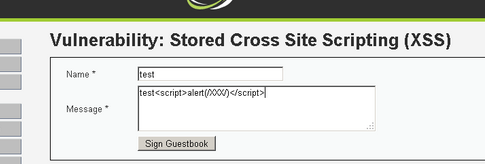
储存式跨站脚本（ XSS ）是一种最危险的跨站脚本。允许用户存储数据的Web 应用程序都有可能接触到这种类型的攻击。查看源码



正常情况下，在输入姓名和message学信息后会出现下面的：

：

提交的文本框中输入“<script>alert(/XXX/)</script>”，提交成功后，以后只要访问该页面就会弹出对话框。





实验总结：

1. 学习了Xammp的使用及其各部分的功能
2. 学会了如何在实际过程中着手分析网络安全漏洞
3. 对sql注入有了具体的认识，并且知道了在不同的安全级别模式下如何对应分析。blind模式下如何分析，
4. 知道了如何通过查看网页源代码找出可能存在的漏洞
5. 对跨站脚本注入有了较为深刻的认识。并且通过对反射型xss和存储式跨站脚本的实践，知道了生活中网络威胁的因素。

困难解决：

1.apache启动过程中出现问题，原因是80端口被占用，通过 netstat查找之后，关闭相应进程

2.在口令破解部分，由于最初没有把protocol设置成为web form,运行bruter好多次都出现失败的情况，改正之后，能够正确运行并找出密码

3在sql injection注入过程中，虽然我在配置文件当中把安全级别更改为low，我以为这样每次运行的时候就会默认安全级别为low，然而每次启动的时候系统都会默认为impossible，每次都无法准确运行，手动更改之后，就可以了。