UNCTF2020部分writeup

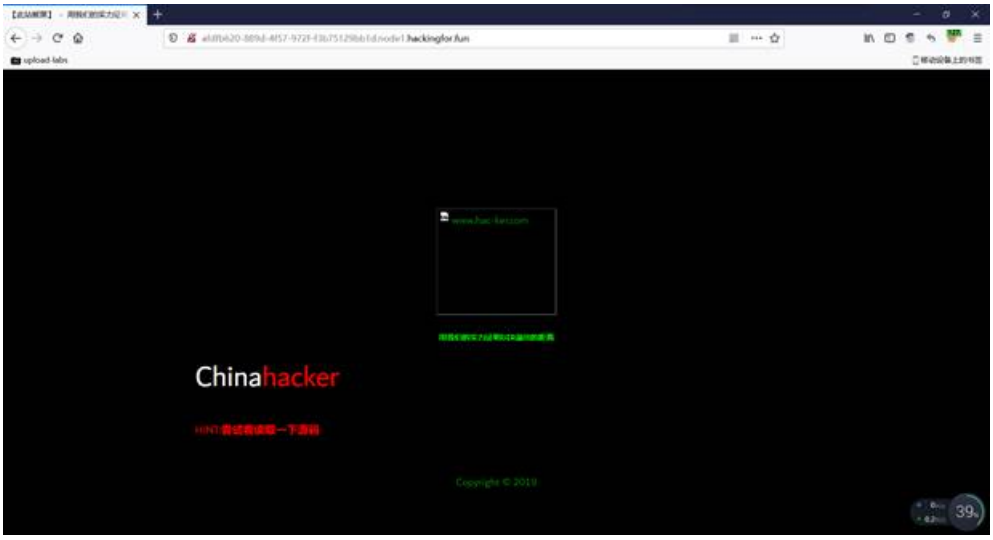
原创

桥畔某处人家 于 2020-11-28 19:07:40 发布 279 收藏 1

分类专栏： CTF 文章标签： 信息安全

本文链接：https://blog.csdn.net/cr_hunter/article/details/110289949

Web题： L0vephp

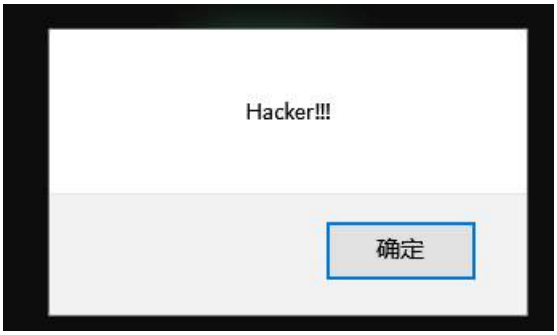


提示读取源代码，发现注释中有如下提示：

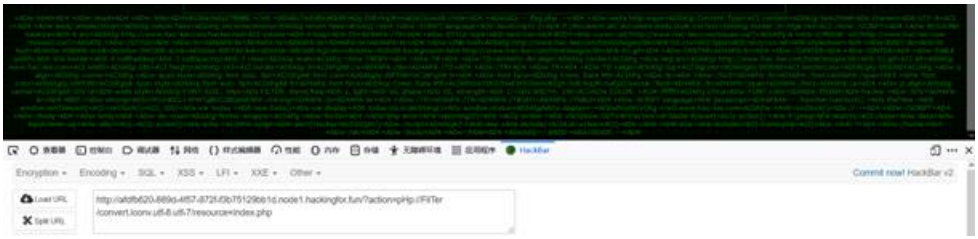
```
<title>【此站被黑】 — 用我们的实力证明DIR溢出的距离</title>
<!-- flag.php -->


<!-- B4Z0-@:OCnDf, -->
```

```
【B4Z0-@:OCnDf,】猜测是base家族编码，base85解出来为：get action
```

利用PHP伪协议读取，发现base被禁了，



于是采用utf-7：



还原得到：

```php
<?php
$flag = "unctf{7his_is_@_f4ke_f1a9}";
//hint:316E4433782E706870
?>
```

将【316E4433782E706870】十六进制转换为字符串【1nD3x.php】

316E4433782E706870

字符集 utf8(unicode编码)

1nD3x.php

访问1nD3x.php

```php
<?php

error_reporting(0);
show_source(__FILE__);
$code=$_REQUEST['code'];

$_=array('0','\'','\"','\&','\?','\<','\>','\|','\*','\`','\+','\-','\{','\}','\[','\]','\\\\','\/');
$__=array('eval','system','exec','shell_exec','assert','passthru','array_map','ob_start','create_function','call_user_func','call_user_func_array','array_filter','proc_open');
$blacklist1 = array_merge($_);
$blacklist2 = array_merge($__);

if (strlen($code)>16){
    die('Too long');
}

foreach ($blacklist1 as $blacklisted) {
    if (preg_match ('/' . $blacklisted . '/im', $code)) {
        die('NTTTTT');
    }
}

foreach ($blacklist2 as $blacklitem) {
    if (preg_match ('/' . $blacklitem . '/im', $code)) {
        die('Sry,try again');
    }
}

@eval($code);
?>
```

经过学长提示，发现可以用远程包含，利用include$_GET[1];可以完成，于是payload：

```
&1=data://text/plain,<?php system('ls /');
```

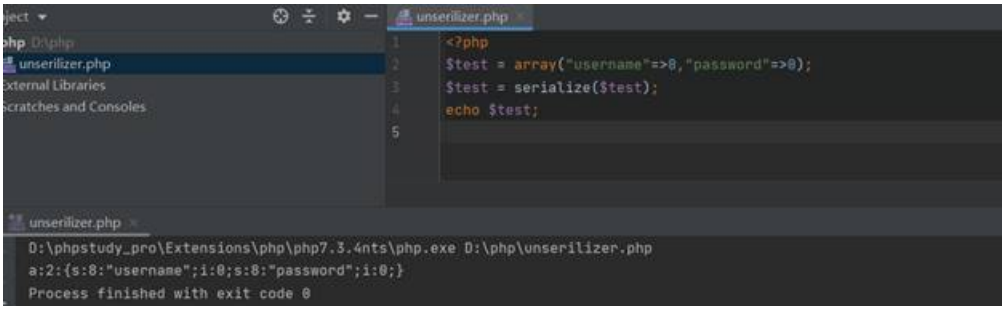
发现flag

拿到flag：

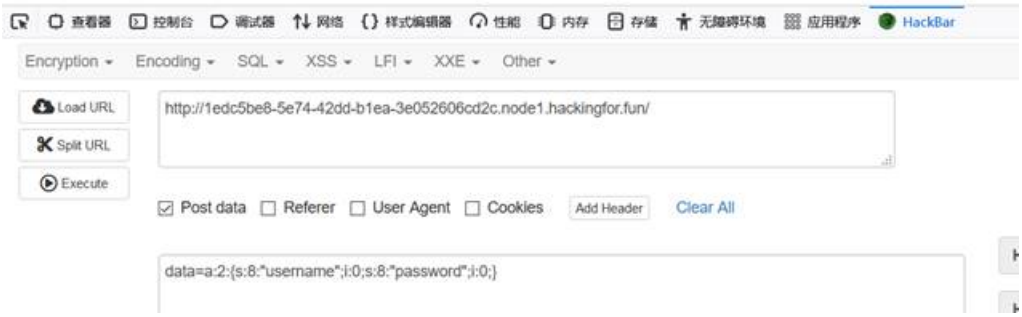
Web2：ezphp

分析代码：POST数据之后进行反序列化，反序列化的username等于一个字符串和password等于一个字符串，经测试，发现不是已经给出的字符串，看到两个等号想到弱类型比较，在php中0==‘abc’的值是True【字符串“admin”与数值0比较，先将admin强制转化成数值，由于“admin”是字符串，不包含数字，所以转化结果为0】

思路：只需传入一个序列化的数组，将username和password赋值为0即可

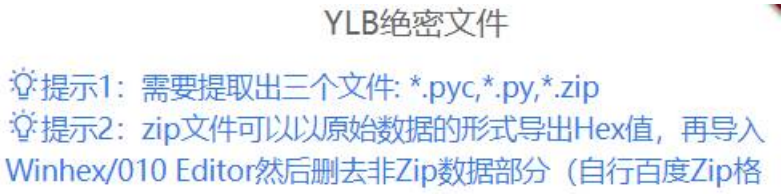


```
        echo "username or password error!";
}
FLAG{bc3e7a0b-caba-4b95-b3d3-55f5fe114a9d}
```



MISC：YLB绝密文件

下载附件打开，发现是个流量分析题，用wireshark打开分析，根据题目提示，我们需要提取出三个文件：



于是用http.request.method==POST 过滤所有的post包，看到有三次post的数据。

分别对应三个文件，一次提取出来即可

对应的三个文件为：

secret.cpython-38.pyc,反编译得到：

YLBSB.zip

解压发现YLBSB.xor

最后还有xor.py

分析xor.py可以得知YLBSB的二进制数据进行Base64编码之后再进行异或运算

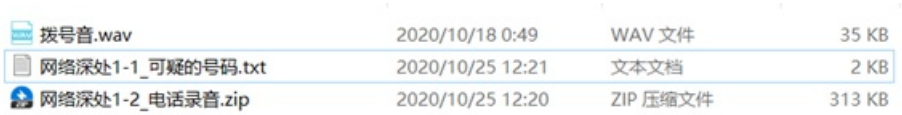
所以得到解密脚本

```
#coding:utf-8

import base64

key = 'YLBSB?YLBNB!'

file = open("YLBSB.docx", "wb")

enc = open("YLBSB.xor", "rb")

plain = enc.read().decode()

count = 0

for c in plain:

  d = chr(ord(c) ^ ord(key[count % len(key)]))

  file.write(d.encode())

  count = count + 1

with open('YLBSB.docx','rb') as fp:

  data = base64.b64decode(fp.read().decode())

with open('1.doc','wb') as f:

  f.write(data)
```

解码可得1.doc

得到doc直接ctrl+f搜索unctf发现结尾有flag字符串。

MISC：网络深处



网络侦查1-1_可疑的号码.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

题目内故事纯属虚构，完全架空。

你是一名学生，你在夜间路过一个电话亭，一个人鬼鬼祟祟的进入电话亭拨通了一个电话又拿出手机录了音，他反常的行为引起了你的注意，他走后你决定去电话亭看看。

电话亭里又一个皱巴巴的纸条，上面写着一串数字：

6368068417483687504777205288954926110397288189134951041127819192631740400603597761717124966060313732119498817791789244647988520022283702947365467004382106874861784922084718125702163810773410153219040799777733523081595853353767460268829074668938648158872741587329651857373729926971088623620615826466388417333610460860531272849005326588852205693502533834690477417426867301287636802530488836384465284217609291317839802783915569128934052144646248848245556478813523005503601614297588336572431312384783112199154491713583596166655704292307386212729885818 71，这很可能是刚才的人不小心丢在这里的，这显然不是电话号码，这使你更加好奇，你决定看看他拨的是什么电话号码。

你按了一下重拨键，想看看他拨打的电话号码，但是这个公用电话的屏幕坏了，之传出了一段拨号音，你迅速挂掉电话又重拨了一次并录下了拨号音。

回到寝室的你像弄清楚字条的含义，看来只有得到他拨打的电话才能搞明白字条的含义了。

得到电话号码以后，你拨通了他，里面传出一段杂音，一筹莫展的你决定将这件奇怪的事情告诉警察。

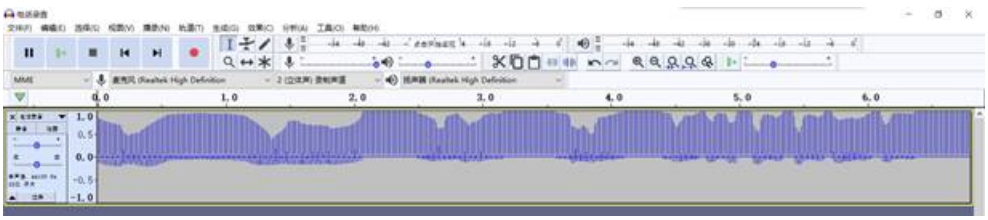
电话号码就是压缩包密码

根据提示，可以用软件将压缩包密码爆破出来：

大胆猜测以13开头，得到15975384265

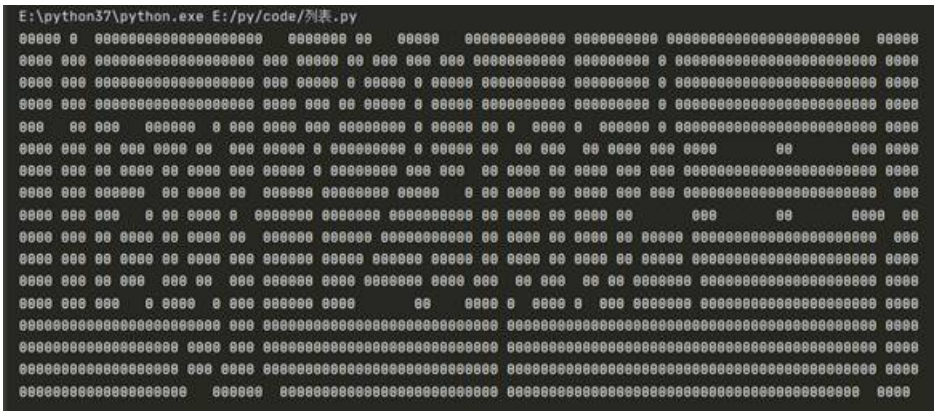
解压得到音频文件，发现是一段噪音，放进Audacity分析一下：

发现波形没什么特别，于是习惯性的看一下频谱：发现提示



百度之后，了解到tupper指的是【tupper自我指涉公式】，此公式的二维图像与公式本身外观一样，根据官方的脚本，跑一下：

```
from functools import reduce

# tupper自我指涉公式

def Tupper_self_referential_formula():
  k = 6368068417483687504777205288954926110397288189134951041127819192631740400603597761717124966060313732119498
81779178924464798852002228370294736546700438210687486178492208471812570216381077341015321904079977773352308159585
33537674602688290746689386481588727415873296518573737299269710886236206158264663884173336104608605312728490053 2
65888522056935025338346904774174268673012876368025304888363844652842176092913178398027839155691289340521446462488
84824555647881352300550360161429758833657243131238478311219915449171358359616665570429230738621272988581871

  def f(x, y):
    d = ((-17 * x) - (y % 17))
    e = reduce(lambda x, y: x * y, [2 for x in range(-d)]) if d else 1
    f = ((y // 17) // e)
    g = f % 2
    return 0.5 < g
  for y in range(k + 16, k - 1, -1):
    line = ""
    for x in range(0, 107):
      if f(x, y):
        line += "0"
      else:
        line += " "
    print(line)

if __name__ == '__main__':
  if Tupper_self_referential_formula():
    print(str(Tupper_self_referential_formula()))
```



拿到flag。这个题也有幸拿到了三血：

Crypto: 简单的RSA

e=
184376135702474457377046307761507757355092445256333035329218131229975499547418288558988423569005377466474146762720223979891611809964672407956619281172738376666154151535719592588478295281315194234862617575694540119403188495897301520315283235769978017882064575485318026638344183810615512275449374127347765817 81

n=
147282573611984580384965727976839351356009465616053475428039851794553880833177877211323318130843267847303264730088424552657129314295117614222630326581943132950689147833674506592824134135054877394753008169629583742916853056999371985307138775298080986801742942833212727949277517691311315098722536282119888605701

c=
140896698267670480175739817539898638657099087197096836734243016824204113452987617610944986742919793506024892638851339015015706164412994514598564989374037762836439262224649359411190187875207060663509777017529293145434535056275850555331099130633232844054767057175076598741233988533181035871238444008366306956934

可以发现给的e非常大，这就会导致算出来的d非常小，十分容易被攻击，而这就是维纳攻击，可以利用攻击脚本解出d。

```
D:\桌面\rsa-wiener-attack>python RSAwienerHacker.py
Hacked!
746513545063397828988614555413191780615835546049803635493013732811414198212 53
```

d= 746513545063397828988614555413191780615835546049803635493013732811414198212 53

接下来破解密文就十分简单了，

```
from Crypto.Util.number import long_to_bytes

d= 746513545063397828988614555413191780615835546049803635493013732811414198212 53

c= 140896698267670480175739817539898638657099087197096836734243016824204113452987617610944986742919793506024892638851339015015706164412994514598564989374037762836439262224649359411190187875207060663509777017529293145434535056275850555331099130633232844054767057175076598741233988533181035871238444008366306956934

n=147282573611984580384965727976839351356009465616053475428039851794553880833177877211323318130843267847303264730088424552657129314295117614222630326581943132950689147833674506592824134135054877394753008169629583742916853056999371985307138775298080986801742942833212727949277517691311315098722536282119888605701

m = pow(c, d, n)

plaintext = long_to_bytes(m)

print(plaintext)
```

拿到flag

```
E:\python37\python.exe E:/py/code/列表.py
b'unctf{wi3n3r_Att@ck}'
```