

# 2020 第四届强网杯 线上赛 Misc\_Writeup

原创

末初 于 2020-11-07 17:00:12 发布 1110 收藏 4

分类专栏: [CTF\\_MISC\\_Writeup](#)

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/mochu777777/article/details/109549349>

版权



[CTF\\_MISC\\_Writeup](#) 专栏收录该内容

246 篇文章 46 订阅

订阅专栏

## 目录

- [upload](#)
- [签到](#)
- [问卷调查](#)
- [miscstudy](#)

## upload

下载附件, 打开是流量包文件, wireshark打开

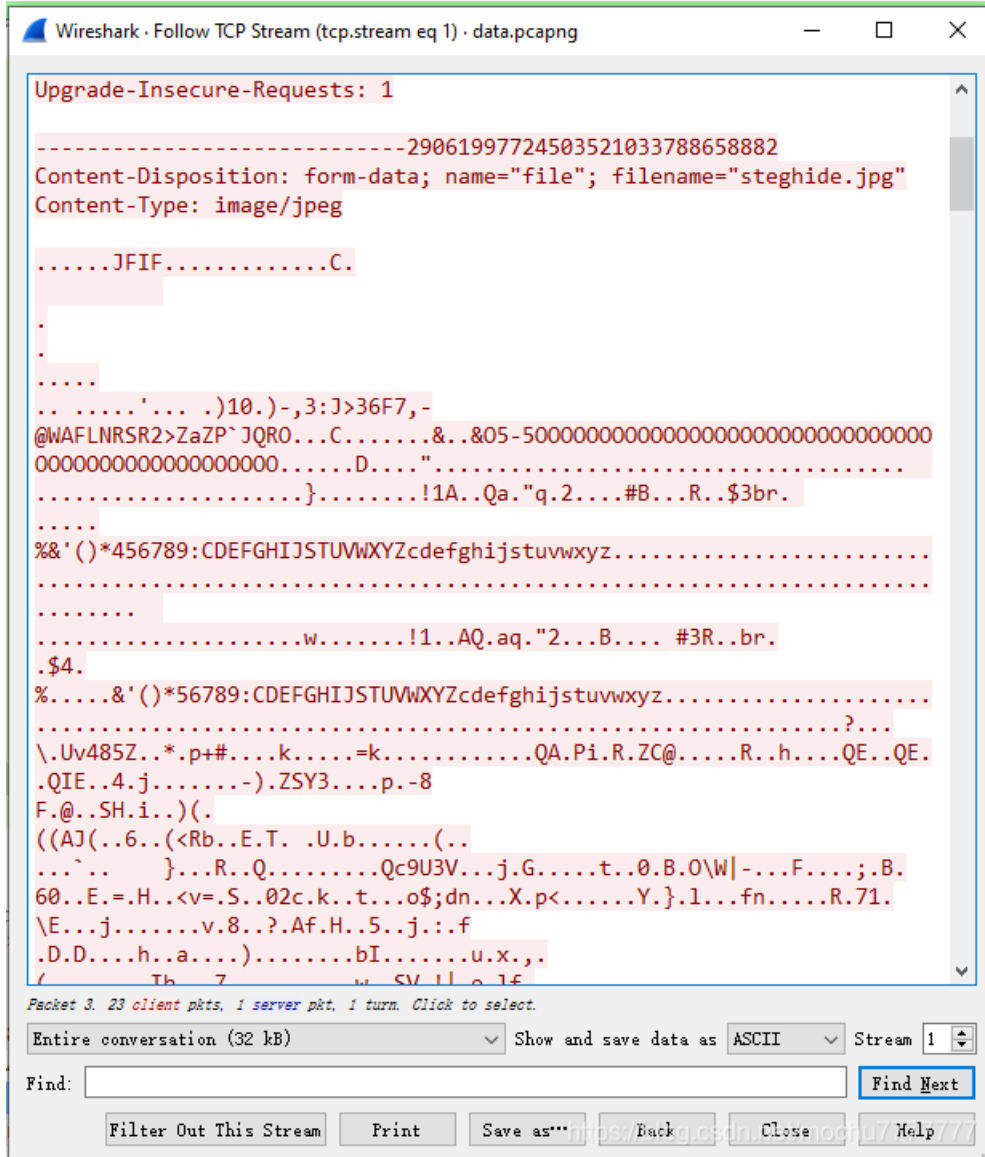
```
0000 00 0c 29 25 4d 82 00 50 56 c0 00 08 08 00 45 00  .)SM:P.V...E
0010 02 26 ff d2 40 00 00 06 9d 28 c0 a8 6d 01 c0 a8  &.@... (m...
0020 6d 84 f1 b4 00 50 12 3c 17 06 0c c4 79 62 80 18  m...P<...yb...
0030 02 02 0e 4e 00 01 01 08 0a 72 e1 e4 50 01 69  .H...P-1
0040 2f 26 47 45 54 20 2f 20 48 54 50 2f 31 2e 31  /&GET / HTTP/1.1
0050 0d 0a 48 6f 73 74 3a 20 31 39 32 2e 31 36 30 2e  .Host: 192.168.
0060 31 30 39 2e 31 33 32 0d 0a 55 73 65 72 2d 41 67  109.132-User-Ag
0070 65 6e 7a 3a 20 4d 6f 7a 69 6c 6c 61 2f 35 2e 30  ent: Moz illa/5.0
0080 20 28 57 69 6e 64 6f 77 73 20 4e 54 20 31 30 2e  (window s NT 0.
```

查看 http 的包, 追踪一下

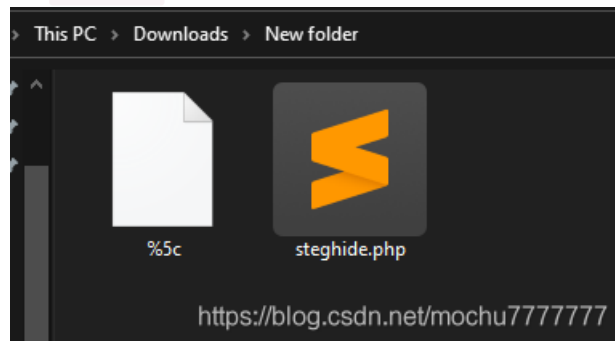
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.109.1	192.168.109.132	HTTP	564	GET / HTTP/1.1

24 002181	192.168.109.132	192.168.109.1	HTTP	650 HTTP/1.1 200 OK (text/html)
25 11.016447	192.168.109.1	192.168.109.132	HTTP	381 POST /steghide.php HTTP/1.1 (JPEG JFIF image)
26 11.020453	192.168.109.132	192.168.109.1	HTTP	269 HTTP/1.1 200 OK

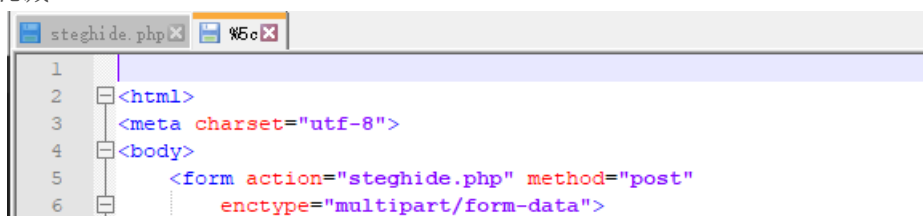
很明显是POST上传的图片



File->Export Object->HTTP... 将文件 Save all 保存出来，得到如下：



%5c 有提示 steghide 隐藏



```
7 <label for="file">文件名:</label>
8 <input type="file" name="file" id="file" />
9 <input type="submit" name="submit" value="提交" />
10 <!--i use steghide with a good password-->
11 </form>
12 </body>
13 </html>
14
```

<https://blog.csdn.net/mochu7777777>

steghide.php 用notepad++打开

```
steghide.php %5cX
1 -----29061997724503521033788658882
2 Content-Disposition: form-data; name="file"; filename="steghide.jpg"
3 Content-Type: image/jpeg
4
5 xFFxDBxFFxEO NUL DLE JFIF NUL SOH SOH NUL NUL SOH NUL NUL xFF xDB NUL C NUL
6
7 VI
8 DC1
9 VJNAKVISOSO
```

去掉前面这四行，保存修改后缀为 jpg 或者 png 都行，得到如下图：



然后把照片丢进 kali 使用 steghide 工具提取隐藏信息

```
root@kali:/home/mochu7/Desktop# steghide info steghide.jpg
"steghide.jpg":
  format: jpeg
  capacity: 1.6 KB
Try to get information about embedded data ? (y/n) y
Enter passphrase:
steghide: could not extract any data with that passphrase!
root@kali:/home/mochu7/Desktop#
```

有密码，在网上找个爆破steghide密码的脚本，如下：

```

# -*- coding: utf8 -*-
#python2
from subprocess import *

def foo():
    stegoFile='a.jpg'#这里填图片名称
    extractFile='hide.txt'#输出从图片中得到的隐藏内容
    passFile='english.dic'#字典,用的是Advanced Archive Password Recovery的字典

    errors=['could not extract','steghide --help','Syntax error']
    cmdFormat='steghide extract -sf "%s" -xf "%s" -p "%s"'
    f=open(passFile,'r')

    for line in f.readlines():
        cmd=cmdFormat %(stegoFile,extractFile,line.strip())
        p=Popen(cmd,shell=True,stdout=PIPE,stderr=STDOUT)
        content=unicode(p.stdout.read(),'gbk')
        for err in errors:
            if err in content:
                break
        else:
            print content,
            print 'the passphrase is %s' %(line.strip())
            f.close()
            return

if __name__ == '__main__':
    foo()
    print 'ok'
    pass

```

```

Applications  Places  QTerminal

File  Actions  Edit  View  Help

root@kali:/home/mochu7/Desktop# ls
a.jpg brute.py english.dic
root@kali:/home/mochu7/Desktop# python2 brute.py english.dic
wrote extracted data to "hide.txt".
the passphrase is 123456
ok
root@kali:/home/mochu7/Desktop# ls
a.jpg brute.py english.dic hide.txt
root@kali:/home/mochu7/Desktop# cat hide.txt
flag{te11_me_y0u_like_it}root@kali:/home/mochu7/Desktop#
root@kali:/home/mochu7/Desktop#
root@kali:/home/mochu7/Desktop# steghide extract -sf a.jpg
Enter passphrase:
wrote extracted data to "flag.txt".
root@kali:/home/mochu7/Desktop# ls
a.jpg brute.py english.dic flag.txt hide.txt
root@kali:/home/mochu7/Desktop# cat flag.txt
flag{te11_me_y0u_like_it}root@kali:/home/mochu7/Desktop#
root@kali:/home/mochu7/Desktop#
root@kali:/home/mochu7/Desktop#

https://blog.csdn.net/mochu777777

```

密码是: 123456

hide.txt 已经提取了隐藏的flag的内容, 或者也可以 steghide extract -sf a.jpg 然后输入密码, 得到 flag.txt

```
flag{te11_me_y0u_like_it}
```

签到



老天爷，哪次比赛让我签到拿个一血也行啊~

```
flag{welcome_to_qwb_S4}
```

## 问卷调查



枯了，比赛的时间问卷调查最后出来的，竟然没看到，发现的时候已经结束了...orz

```
flag{Welc0me_t0_qwbS4_Hope_you_play_h4ppily}
```

## miscstudy

Hint: 本题目flag由7个部分构成，第一个部分为flag{level1...，最后一个部分为 !!!} 每一关都会存有flag的一部分，将所有flag的字符串拼接即为最后flag



首先下载附件解压是一个流量包，打开后筛选 http 的包，访问这个url

Wireshark packet capture details for an HTTP GET request:

- GET /fonts/1 HTTP/1.1\r\n
- Host: 39.99.247.28\r\n
- Connection: keep-alive\r\n
- Cache-Control: max-age=0\r\n
- Upgrade-Insecure-Requests: 1\r\n
- User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.125 Safari/537.36\r\n
- Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.9\r\n
- Accept-Encoding: gzip, deflate\r\n
- Accept-Language: zh-CN,zh;q=0.9\r\n
- If-None-Match: "da4-5ad340100ca51"\r\n
- If-Modified-Since: Wed, 19 Aug 2020 05:09:10 GMT\r\n
- \r\n
- [Full request URI: http://39.99.247.28/fonts/1]
- [HTTP request 1/1]
- [Response in frame: 49]

Source: 192.168.43.109, Destination: 39.99.247.28, Protocol: HTTP, Length: 593 bytes.

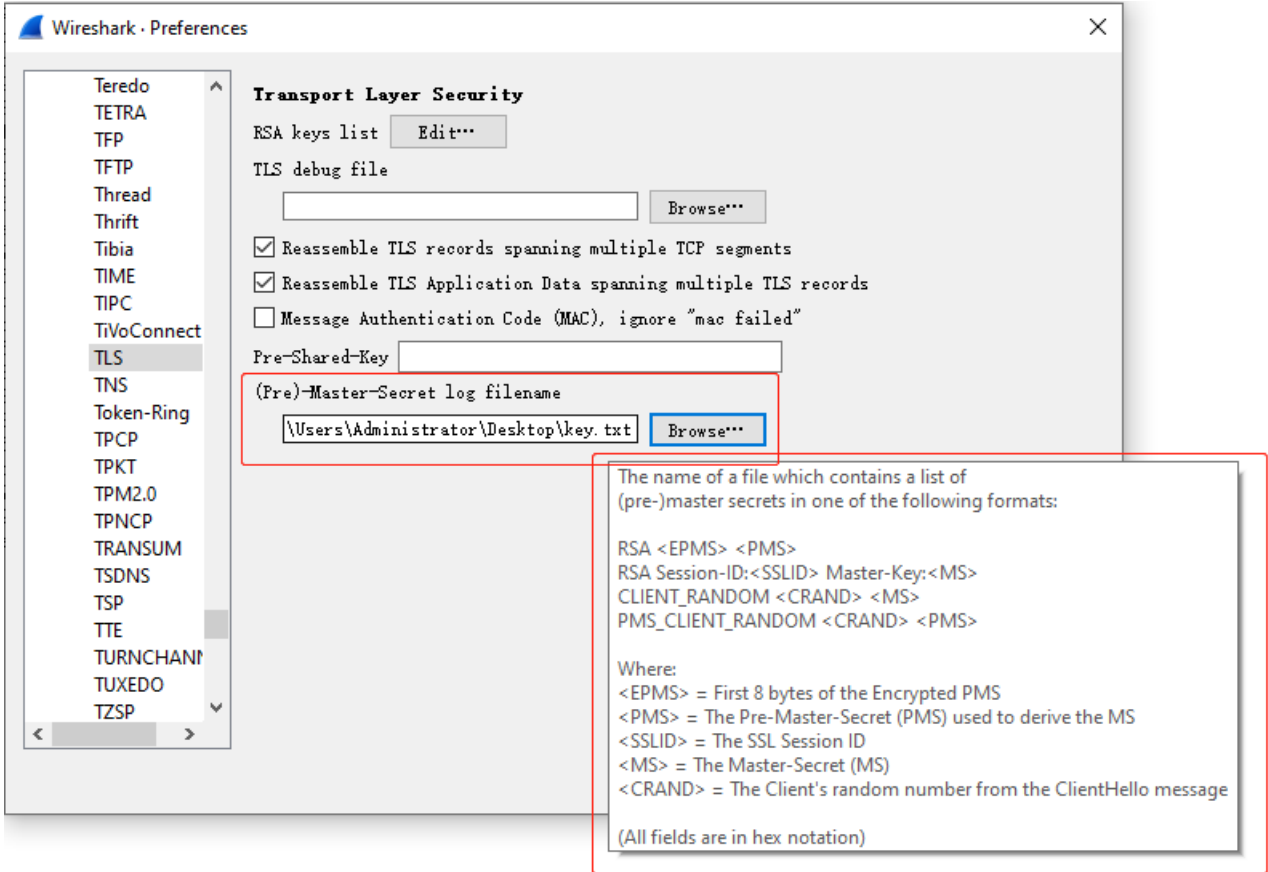
得到 level1 和 level2 的flag

Hex-encoded data from the browser console:

```
CLIENT_RANDOM ac85f424e7a74d096ae8e209a49552c1753811fd3d6ae74c9277bd30362c83f0 0e7ed0e7e726bc2f4277f3334ba7fb78896ef6973e7ecc1fc7246b362df1ff52057e74012bb4df0c2f87b1bfe353c5d8
CLIENT_RANDOM 9e5fa494ae09a50ab5230594217fa0b5e8fcb4c8974acb2c4484436b3b894be 1f9f13ee505e3310b3b4ce3c43254e55906aed5f876179e45ad2904931f1e5ce2c943534fc4700082c7db79652b9fd57
CLIENT_HANDSHAKE_TRAFFIC_SECRET b1f1e098a93d6d2e325923f65fcdce5c67cb22a09374f2e44ac2c8368c93a1ac c93077a61453735933fc6c17f2788ad3fca6d037ea62940e3ba7659ae51b56eb
SERVER_HANDSHAKE_TRAFFIC_SECRET b1f1e098a93d6d2e325923f65fcdce5c67cb22a09374f2e44ac2c8368c93a1ac 8e7165569e3e5a6526050820d934b3dd3df60e441fea13bc321a724bfcce50570
CLIENT_TRAFFIC_SECRET_0 b1f1e098a93d6d2e325923f65fcdce5c67cb22a09374f2e44ac2c8368c93a1ac 52a92bc9764b239be26ca0d92615802e27fb8f9aa7ace3cb44bb24225a387d2
SERVER_TRAFFIC_SECRET_0 b1f1e098a93d6d2e325923f65fcdce5c67cb22a09374f2e44ac2c8368c93a1ac 0c8439418b351620f2e6a733f50025582a3aaf8e9731bd3ade007b041fa989bc
EXPORTER_SECRET b1f1e098a93d6d2e325923f65fcdce5c67cb22a09374f2e44ac2c8368c93a1ac 0a51ca8275a7c20c9eacd4860f7ba14252ab94714cec2dfcfc8c62ff203d615
CLIENT_HANDSHAKE_TRAFFIC_SECRET c52c413417f78c3d831a75ffd34dccc3c434ede3cd456f1fe25f8cebc301e502 7efd12a4b19eab96eade9dcd2abbc01a12c904646e381c41d30e6934d1a9dc2c
SERVER_HANDSHAKE_TRAFFIC_SECRET c52c413417f78c3d831a75ffd34dccc3c434ede3cd456f1fe25f8cebc301e502 79d2c31214d0af49a4397cbeaeb86ca44bc947c79b538894472b7f0fe4c970
CLIENT_TRAFFIC_SECRET_0 c52c413417f78c3d831a75ffd34dccc3c434ede3cd456f1fe25f8cebc301e502 593cbe062157b7e56acd9626219e15d484e6a77b7cd09fd96d83f57c5e210208
SERVER_TRAFFIC_SECRET_0 c52c413417f78c3d831a75ffd34dccc3c434ede3cd456f1fe25f8cebc301e502 bf7de866e92d313d12fb37b427315f234cd504309bc33fb7c8a87e4d1093a0a4
EXPORTER_SECRET c52c413417f78c3d831a75ffd34dccc3c434ede3cd456f1fe25f8cebc301e502 8e8da12784192f2aa70b4e4ca5f832836f0fddcc2cc32f11bf2522dcb7cbcd
CLIENT_RANDOM ba069d4ce42f6d0d78eb17d1a0632be516c764061a1e241597601220f12ee9 1f9f13ee505e3310b3b4ce3c43254e55906aed5f876179e45ad2904931f1e5ce2c943534fc4700082c7db79652b9fd57
SERVER_HANDSHAKE_TRAFFIC_SECRET e9f981e3b3cfe26377db3e666afd10b15f03025e2488bda2bf58eal2d2bce45fb faf948ebf001b1c140a0d222f8149a82fec2b1637b35bda5e5517b4585912c8
CLIENT_HANDSHAKE_TRAFFIC_SECRET f52dbb843bb7035d499c5b234e7615506decebf1b209762dd93a5ac64d9e49eb 93f07c90d8e7827e945dc46e38442f808df78297a97d6279ab9e5f2cbf2c8
SERVER_HANDSHAKE_TRAFFIC_SECRET f52dbb843bb7035d499c5b234e7615506decebf1b209762dd93a5ac64d9e49eb 8d04cc7aa472ab4c803f9e6ae62322a646b37f7019ab21c825826008e1d19a
CLIENT_TRAFFIC_SECRET_0 e9f981e3b3cfe26377db3e666afd10b15f03025e2488bda2bf58eal2d2bce45fb f252e57801505de1828ae5d6cbf83b05828caf4d895f1ba7e68028cc6d0e971b
SERVER_TRAFFIC_SECRET_0 e9f981e3b3cfe26377db3e666afd10b15f03025e2488bda2bf58eal2d2bce45fb 212b74eb937a83e90fc1a75440eec00dc017c59308af7c2bc789f2be65f98f
EXPORTER_SECRET e9f981e3b3cfe26377db3e666afd10b15f03025e2488bda2bf58eal2d2bce45fb 545be7571622b35e9602c2264502f86e94acf2d0d858e0be95291509e251b63
CLIENT_TRAFFIC_SECRET_0 f52dbb843bb7035d499c5b234e7615506decebf1b209762dd93a5ac64d9e49eb d701e889f60049d6c02f36c42dc07517f3e00219830d85e9acd32ef32d6c3
SERVER_TRAFFIC_SECRET_0 f52dbb843bb7035d499c5b234e7615506decebf1b209762dd93a5ac64d9e49eb d0daac2a9e4d7adc2b7e49f254fc391f1a5ec2b6465ebf55a2c5c322c77121d
EXPORTER_SECRET f52dbb843bb7035d499c5b234e7615506decebf1b209762dd93a5ac64d9e49eb a03de91272cfe4d587e8bdf6eb5abd43b9bb23784c6044983ba706692e949f4
CLIENT_RANDOM 97aa926d6f1ba42e9ce6930d8f9c36cd13b879492ef8a2ee6d7ed3d8c3fe 1f9f13ee505e3310b3b4ce3c43254e55906aed5f876179e45ad2904931f1e5ce2c943534fc4700082c7db79652b9fd57
CLIENT_HANDSHAKE_TRAFFIC_SECRET e58fcacfef341dd4cf008c404792e7da77af7b8e8473636d55768614561e72f29 1f9f13ee505e3310b3b4ce3c43254e55906aed5f876179e45ad2904931f1e5ce2c943534fc4700082c7db79652b9fd57
CLIENT_RANDOM b24783c3f62e0af76ca7b6be43ba2788d900ac9170559c5f56f9f831b5ee7d2 1f9f13ee505e3310b3b4ce3c43254e55906aed5f876179e45ad2904931f1e5ce2c943534fc4700082c7db79652b9fd57
SERVER_HANDSHAKE_TRAFFIC_SECRET a06c95c8b7e6ffbe3293cb652d097b31768c821fc25c7166f09e2e620c2d213e 88dbd37dc096ae3f3c8258414607d7084cb04b0071923108af9c9b0e9bb657b
CLIENT_TRAFFIC_SECRET_0 a06c95c8b7e6ffbe3293cb652d097b31768c821fc25c7166f09e2e620c2d213e 8548a64e18bec1569bb870417a94901a0a145a3e5b27397bdc492277aa81626
SERVER_TRAFFIC_SECRET_0 a06c95c8b7e6ffbe3293cb652d097b31768c821fc25c7166f09e2e620c2d213e 674ale1d8648353ba50f6f906ecab65e6440c3e5592f25596e09cc04e045e6
EXPORTER_SECRET a06c95c8b7e6ffbe3293cb652d097b31768c821fc25c7166f09e2e620c2d213e 85b68b6e8e5041fbb171319cc1c108a2a66fc4b47907fb5a9969be0db7fa3f3
flag{level1_begin_and_level2_is_come}
```

flag{level1\_begin\_and\_level2\_is\_come}

除了flag之外，页面中这些其他的参数应该很明显就是 TLS 协议的 Master-Secret log file

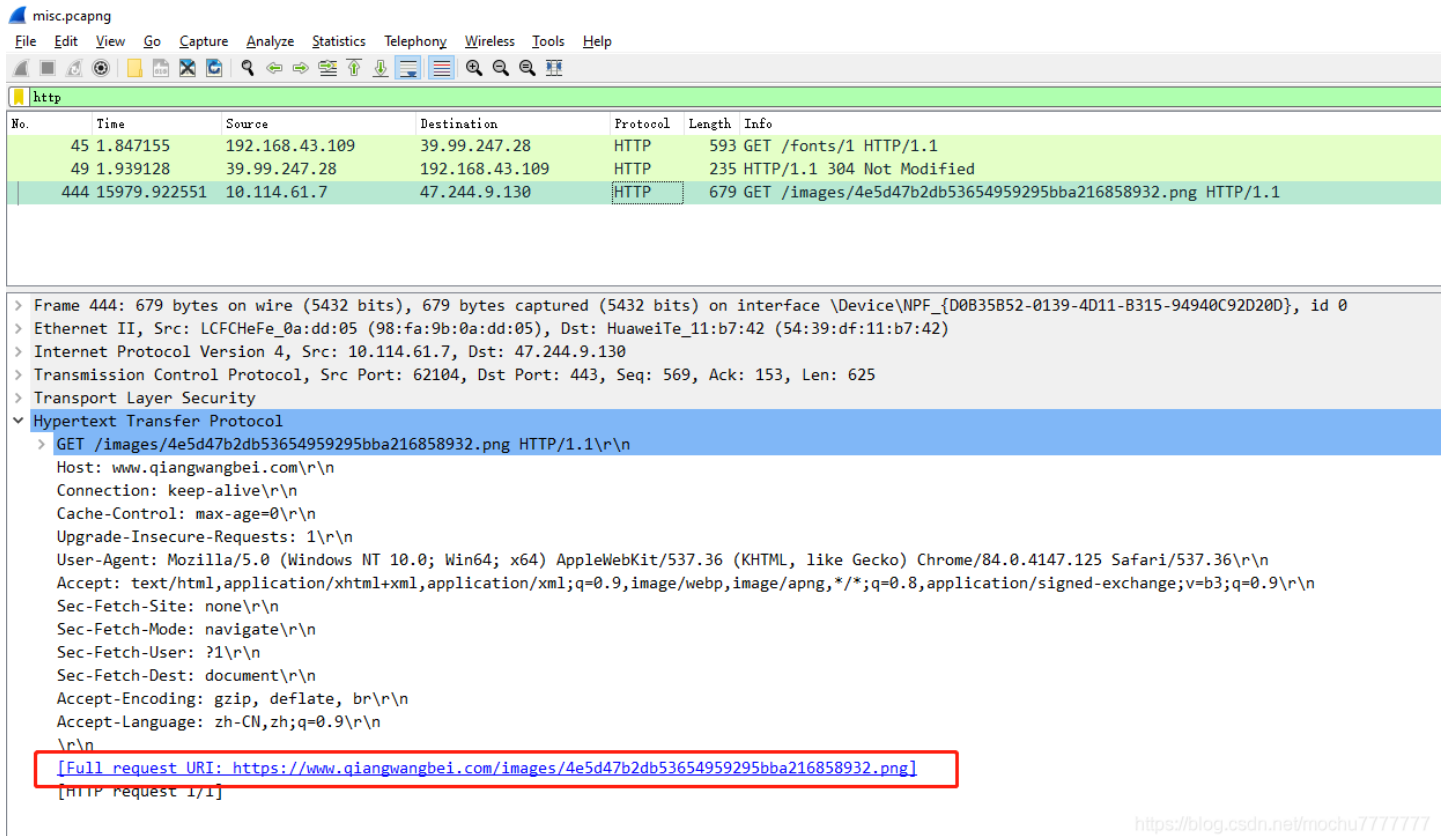


<https://blog.csdn.net/mochu777777>

讲这些参数保存为 ket.txt

在wireshark中，Edit->Preferences->Protocols->TLS->(Pre)-Master-Secret log filename 中选择 ket.txt 然后点击 OK

添加成功后，再次查看http协议的包，发现多了个包



<https://blog.csdn.net/mochu777777>



<https://blog.csdn.net/mochu777777>

保存图片，使用010 Editor或者winhex之类的16进制编辑工具查看

4e5d47b2db53654959295bba216858932.png X

编辑方式: 十六进制(H) 运行脚本 运行模板: PNG.bt

Offset	Hex	ASCII
3:FE10h	2D B0 28 3B EC 60 E5 97 E1 CE FC C7 DC F3 4C 18	-°(;i`â-áíúçÜóL.
3:FE20h	61 AD DB CB 40 F5 63 31 D7 AD B5 5F 2A 44 4C F1	a-ÛË@ôc1*-µ *DLñ
3:FE30h	9A 8D C5 8C CB B4 CD A5 5C EF C6 3F F3 BB DF 32	š.ĂĖĚ'Íŷ\iE?ó»82
3:FE40h	EE D2 A7 17 7E 18 FE 7F 76 67 DA E3 06 D2 B0 DB	iòš.~.p.vgŪă.0°Ü
3:FE50h	B0 00 03 20 49 44 41 54 4D 44 41 77 4D 44 41 77	... IDATMDAwMDAw
3:FE60h	4D 44 41 77 4D 44 41 77 4D 44 41 77 4D 44 41 77	MDAwMDAwMDAwMDAw
3:FE70h	4D 44 41 77 4D 44 41 77 4D 44 41 77 4D 44 41 77	MDAwMDAwMDAwMDAw
3:FE80h	4D 44 41 77 4D 44 41 77 4D 44 41 77 4D 44 41 77	MDAwMDAwMDAwMDAw
3:FE90h	4D 44 41 77 4D 44 41 77 4D 44 41 77 4D 44 41 77	MDAwMDAwMDAwMDAw)
3:FEA0h	4D 44 41 77 4D 44 41 77 4D 44 41 77 4D 44 41 77	MDAwMDAwMDAwMDAw
3:FEB0h	4D 44 41 77 4D 44 41 77 4D 44 41 77 4D 44 41 77	MDAwMDAwMDAwMDAw
3:FEC0h	4D 44 41 77 4D 44 41 77 4D 44 41 77 4D 44 41 77	MDAwMDAwMDAwMDAw
3:FED0h	4D 44 41 77 4D 44 41 77 4D 44 41 77 4D 44 41 77	MDAwMDAwMDAwMDAw
3:FEE0h	4D 44 41 77 4D 44 41 77 4D 44 41 77 4D 44 41 77	MDAwMDAwMDAwMDAw
3:FEF0h	4D 44 41 77 4D 44 41 77 4D 44 41 78 4D 54 45 78	MDAwMDAwMDAwMTEw
3:FF00h	4D 54 45 78 4D 54 45 78 4D 54 45 77 4D 44 41 77	MTEwMTEwMTEwMDAw
3:FF10h	4D 44 41 77 4D 54 41 77 4D 54 45 77 4D 44 41 77	MDAwMTAwMTEwMDAw
3:FF20h	4D 54 41 77 4D 44 41 77 4D 44 41 77 4D 44 45 78	MTAwMDAwMDAwMDEw
3:FF30h	4D 54 41 77 4D 44 41 78 4D 54 45 78 4D 54 45 78	MTAwMDAwMTEwMTEw
3:FF40h	4D 54 45 78 4D 54 41 77 4D 44 41 78 4D 54 45 78	MTEwMTAwMDAwMTEw
3:FF50h	4D 54 45 78 4D 54 45 78 4D 54 45 77 4D 44 41 77	MTEwMTEwMTEwMDAw
3:FF60h	4D 44 41 77 4D 54 45 77 4D 54 45 78 4D 44 41 77	MDAwMTEwMTEwMDAw
3:FF70h	4D 54 41 77 4D 44 41 77 4D 44 41 77 4D 44 45 78	MTAwMDAwMDAwMDEw
3:FF80h	4D 54 41 77 4D 44 41 78 4D 54 45 78 4D 54 45 78	MTAwMDAwMTEwMTEw
3:FF90h	4D 54 45 78 4D 54 41 77 4D 44 41 78 4D 54 41 77	MTEwMTAwMDAwMTAw
3:FFA0h	4D 44 41 77 4D 44 41 77 4D 54 45 77 4D 44 45 77	MDAwMDAwMTEwMDEw
3:FFB0h	4D 44 45 78 4D 54 45 78 4D 54 45 77 4D 44 45 77	MDEwMTEwMTEwMDEw
3:FFC0h	4D 44 41 77 4D 44 41 77 4D 44 41 78 4D 54 41 77	MDAwMDAwMDAwMTEw

模板结果 - PNG.bt

名称	值	开始	大小	颜色	注释
struct PNG_SIGNATURE sig		0h	8h	Fg: Bg:	
struct PNG_CHUNK chunk[0]	IHDR (Critical, ...	8h	19h	Fg: Bg:	
struct PNG_CHUNK chunk[1]	sRGB (Ancillary, ...	21h	Dh	Fg: Bg:	
struct PNG_CHUNK chunk[2]	gAMA (Ancillary, ...	2Eh	10h	Fg: Bg:	
struct PNG_CHUNK chunk[3]	pHYs (Ancillary, ...	3Eh	15h	Fg: Bg:	
struct PNG_CHUNK chunk[4]	IDAT (Critical, ...	53h	FFB1h	Fg: Bg:	
struct PNG_CHUNK chunk[5]	IDAT (Critical, ...	10004h	10000h	Fg: Bg:	
struct PNG_CHUNK chunk[6]	IDAT (Critical, ...	20004h	10000h	Fg: Bg:	
struct PNG_CHUNK chunk[7]	IDAT (Critical, ...	30004h	FF4Ch	Fg: Bg:	
struct PNG_CHUNK chunk[8]	IDAT (Critical, ...	3FE50h	32Ch	Fg: Bg:	
struct PNG_CHUNK chunk[9]	IDAT (Critical, ...	4017Ch	36Ch	Fg: Bg:	



```

> struct PNG_CHUNK chunk[10] IDAT (Critical, ... 404E8h C4Ch Fg: Bg:
> struct PNG_CHUNK chunk[11] IDAT (Critical, ... 41134h 20h Fg: Bg:
> struct PNG_CHUNK chunk[12] IEND (Critical, ... 41154h Ch Fg: Bg:

```

<https://blog.csdn.net/mochu777777>

chunk8-chunk11 的 IDAT 标志 后都跟着一串类似base64的编码，而且除了 chunk11 其他 chunk 的base编码看着应该就是大量的 01，其中 chunk11 中的 IDAT 后的内容比较不一样，复制出来解密得到：

```

4:1100h: 4D 44 41 77 4D 44 41 77 4D 44 41 77 4D 44 41 77 MDAwMDAwMDAwMDAw
4:1110h: 4D 44 41 77 4D 44 41 77 4D 44 41 77 4D 44 41 77 MDAwMDAwMDAwMDAw
4:1120h: 4D 44 41 77 4D 44 41 77 4D 44 41 77 4D 44 41 77 MDAwMDAwMDAwMDAw
4:1130h: AC 22 3C 02 00 00 00 14 49 44 41 54 62 47 56 32 bGV2ZWwzX3NOYXJ0e210
4:1140h: 5A 57 77 7A 58 33 4E 30 59 58 4A 30 58 32 6C 30 ZWwzX3NOYXJ0e210
4:1150h: A6 E9 37 2D 00 00 00 00 49 45 4E 44 AE 42 60 82 !é7-...IEND0B`
4:1160h:

```

模板结果 - PNG.bt

名称	值	开始	大小	颜色	注释
> struct PNG_SIGNATURE sig		0h	8h	Fg: Bg:	
> struct PNG_CHUNK chunk[0]	IHDR (Critical, ...	8h	19h	Fg: Bg:	
> struct PNG_CHUNK chunk[1]	sRGB (Ancillary, ...	21h	Dh	Fg: Bg:	
> struct PNG_CHUNK chunk[2]	gAMA (Ancillary, ...	2Eh	10h	Fg: Bg:	
> struct PNG_CHUNK chunk[3]	pHYs (Ancillary, ...	3Eh	15h	Fg: Bg:	
> struct PNG_CHUNK chunk[4]	IDAT (Critical, ...	53h	FFB1h	Fg: Bg:	
> struct PNG_CHUNK chunk[5]	IDAT (Critical, ...	10004h	10000h	Fg: Bg:	
> struct PNG_CHUNK chunk[6]	IDAT (Critical, ...	20004h	10000h	Fg: Bg:	
> struct PNG_CHUNK chunk[7]	IDAT (Critical, ...	30004h	FE4Ch	Fg: Bg:	
> struct PNG_CHUNK chunk[8]	IDAT (Critical, ...	3FE50h	32Ch	Fg: Bg:	
> struct PNG_CHUNK chunk[9]	IDAT (Critical, ...	4017Ch	36Ch	Fg: Bg:	
> struct PNG_CHUNK chunk[10]	IDAT (Critical, ...	404E8h	C4Ch	Fg: Bg:	
> struct PNG_CHUNK chunk[11]	IDAT (Critical, ...	41134h	20h	Fg: Bg:	
> struct PNG_CHUNK chunk[12]	IEND (Critical, ...	41154h	Ch	Fg: Bg:	

<https://blog.csdn.net/mochu777777>

## Base64编码转换

bGV2ZWwzX3NOYXJ0e210

---

解密结果以16进制显示

---

level3\_start\_it

<https://blog.csdn.net/mochu777777>

```
level3_start_it
```

chunk8-chunk10 三段中得到base64编码如下：







得到二维码，使用 QR Research 扫描



网盘下载得到压缩包 `level4.zip`，解压得到 `level4.jpg`



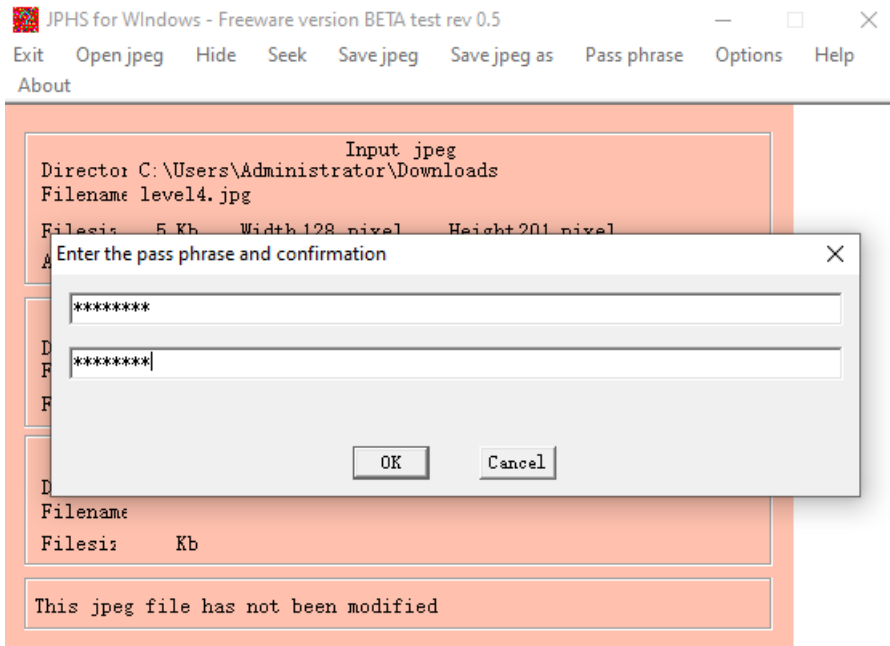
使用 `stegdetect` 检测发现有 `jphide`

```
→ level4 stegdetect level4.jpg
level4.jpg : jphide(*)
→ level4 █
```

使用 `stegbreak` 爆破密码，得到密码为：`power123`

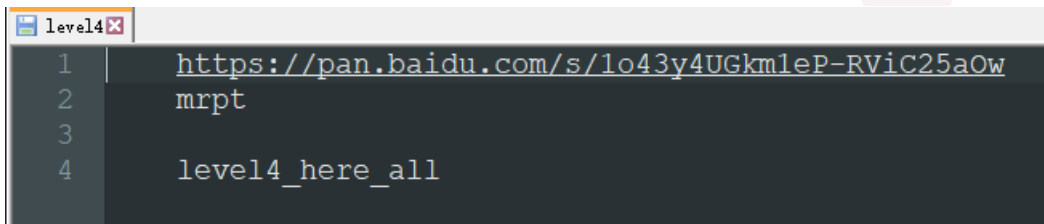
```
→ level4
→ level4 stegbreak -r /usr/share/stegbreak/rules.ini -f pawd_list.txt level4.jpg
Loaded 1 files ...
level4.jpg : jphide[v5](power123)
Processed 1 files, found 1 embeddings.
Time: 0 seconds: Cracks: 734,      inf c/s
→ level4 █
```

使用 `JPHS` 工具打开图片，点击 `Seek` 输入密码



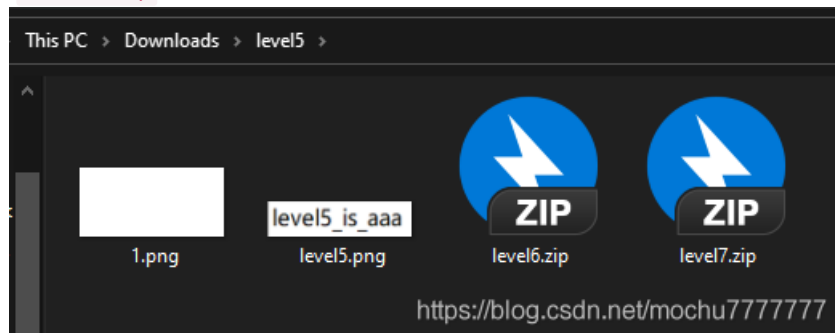
<https://blog.csdn.net/mochu777777>

然后会导出个文件（不需要输入文件后缀什么的，直接随便输入文件名导出即可）这里导出为 `level4`



`level4_here_all`

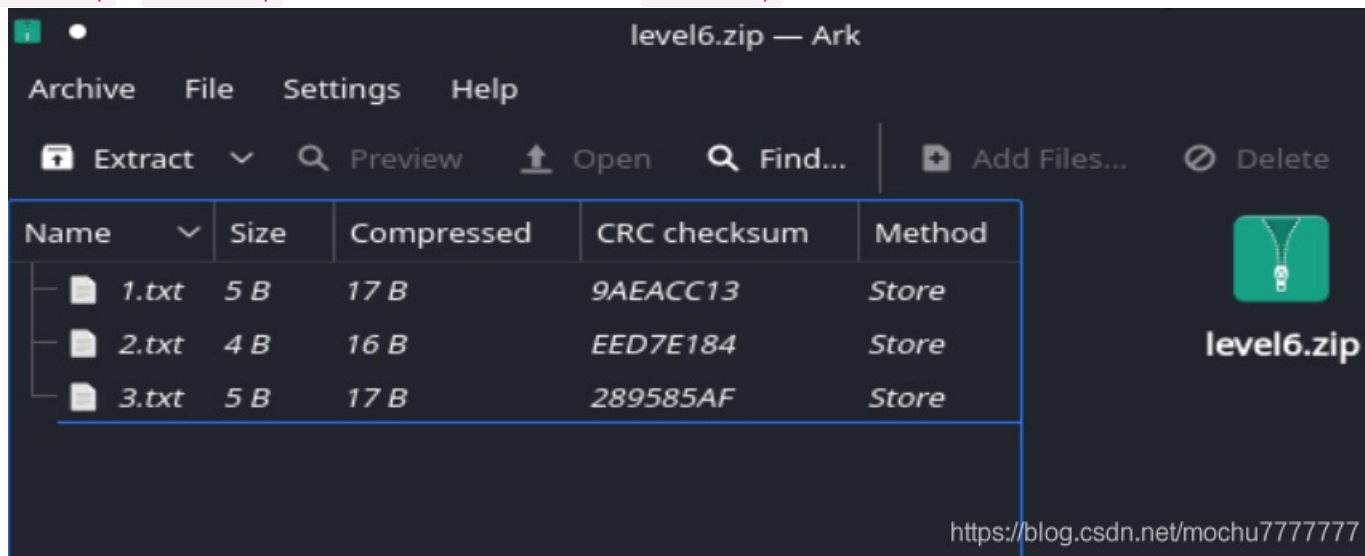
网盘下载附件，得到压缩包 `level5.zip`，解压，如下：



<https://blog.csdn.net/mochu777777>

`level5_is_aaa`

level6.zip 和 level7.zip 都有压缩密码，解不开，先看到 level6.zip



原始文件大小都是  $\leq 5$  Byte，猜测CRC32碰撞，网上很多其他的CRC32碰撞脚本都试了不行，最后找到这个Zip-CRC32碰撞脚本：

Zip-CRC32碰撞脚本: <https://github.com/kmyk/zip-crc-cracker>

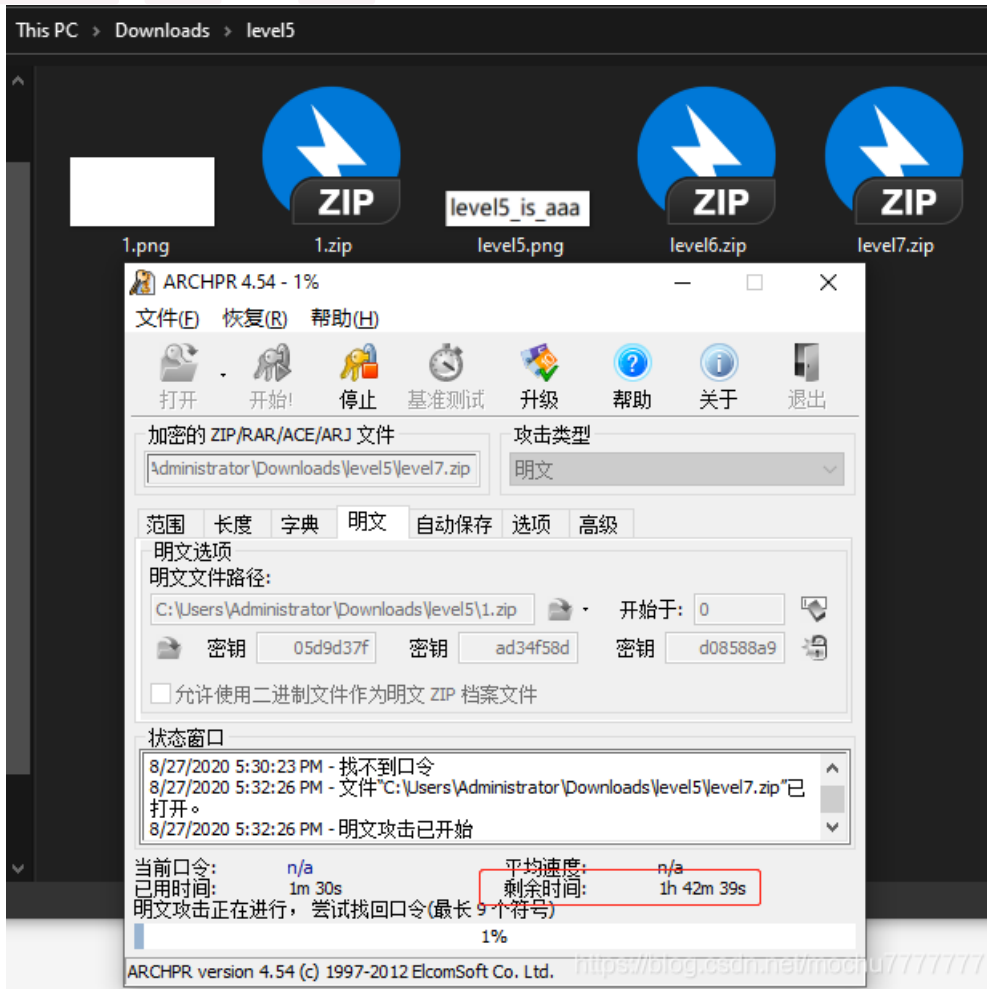
注: 该脚本笔者运行环境为Linux，Windows试过几次不行

```
→ level6 ls
crack.py level6.zip zip-crc-cracker
→ level6 python3 crack.py level6.zip
reading zip files...
file found: level6.zip / 2.txt: crc = 0xeed7e184, size = 4
file found: level6.zip / 3.txt: crc = 0x289585af, size = 5
file found: level6.zip / 1.txt: crc = 0x9aeacc13, size = 5
compiling...
searching...
crc found: 0xeed7e184: "6_is"
crc found: 0x9aeacc13: "level"
crc found: 0x289585af: "n*=em"
crc found: 0x9aeacc13: "p**dx"
crc found: 0x289585af: "ready"
crc found: 0x9aeacc13: "M;f\x0c "
crc found: 0x289585af: "Ot-\x0c!"
crc found: 0x9aeacc13: "Qt:\x0d4"
crc found: 0x289585af: "S;q\x0d5"
crc found: 0x289585af: "?H\x5c\x09q"
done
level6.zip / 2.txt : '6_is'
level6.zip / 3.txt : 'n*=em'
level6.zip / 3.txt : 'ready'
level6.zip / 3.txt : 'Ot-\x0c!'
level6.zip / 3.txt : 'S;q\r5'
level6.zip / 3.txt : '?H\\tq'
level6.zip / 1.txt : 'level'
level6.zip / 1.txt : 'p**dx'
level6.zip / 1.txt : 'M;f\x0c '
level6.zip / 1.txt : 'Qt:\r4'
→ level6
```

按照 1,2,3 的顺序排好

level6\_isready

使用 ARCHPR 打开 level7.zip，1.png 压缩成 1.zip 添加到明文密钥进行明文爆破

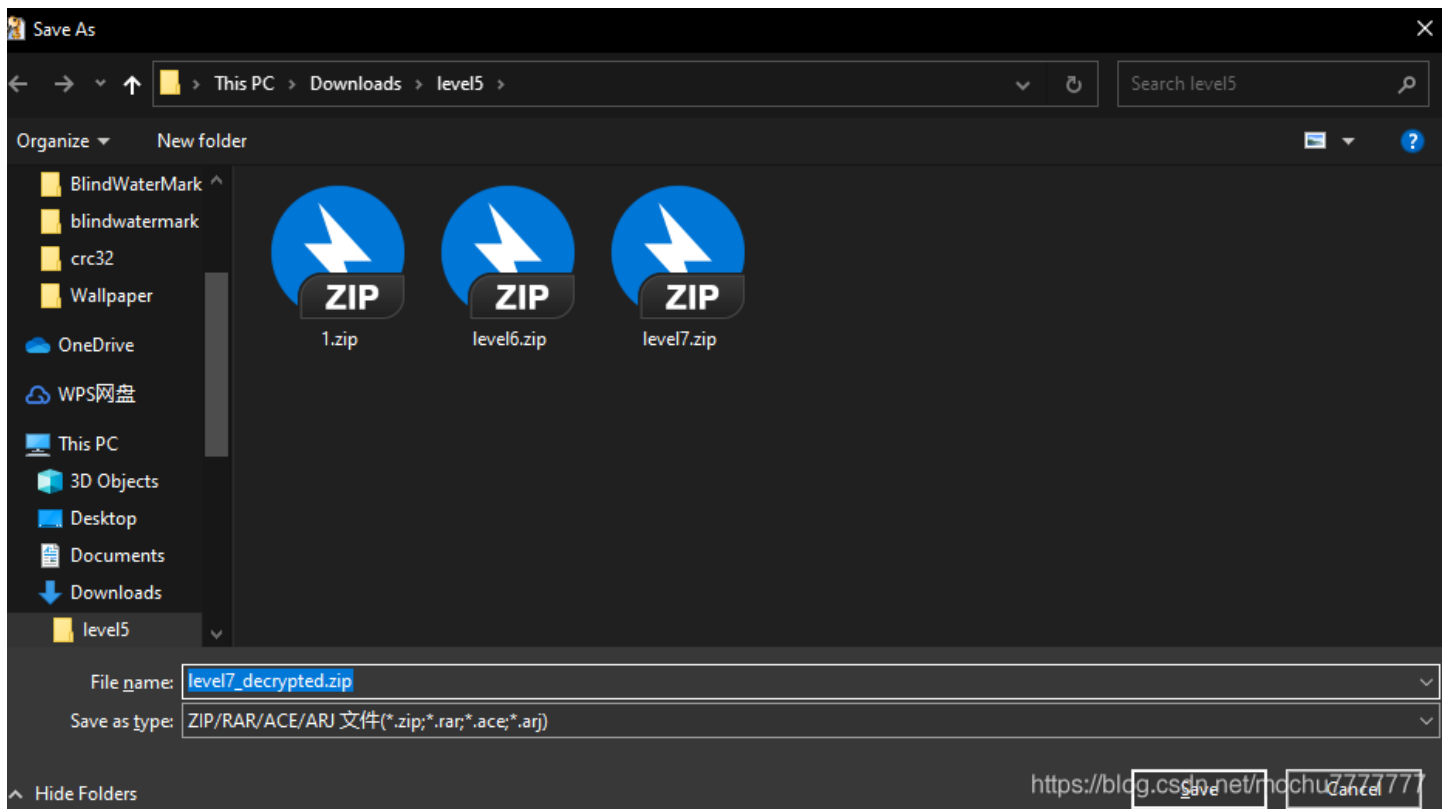
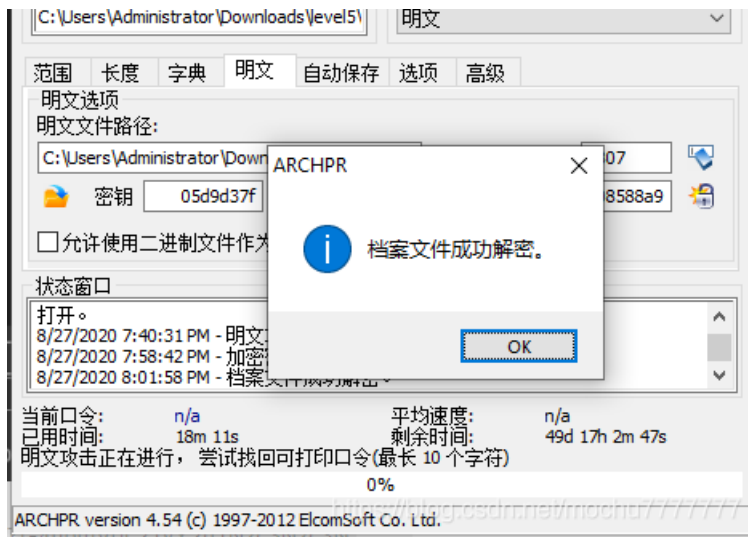


根据网上的师傅的说法, 等到 剩余时间 小于 1h 即可停止

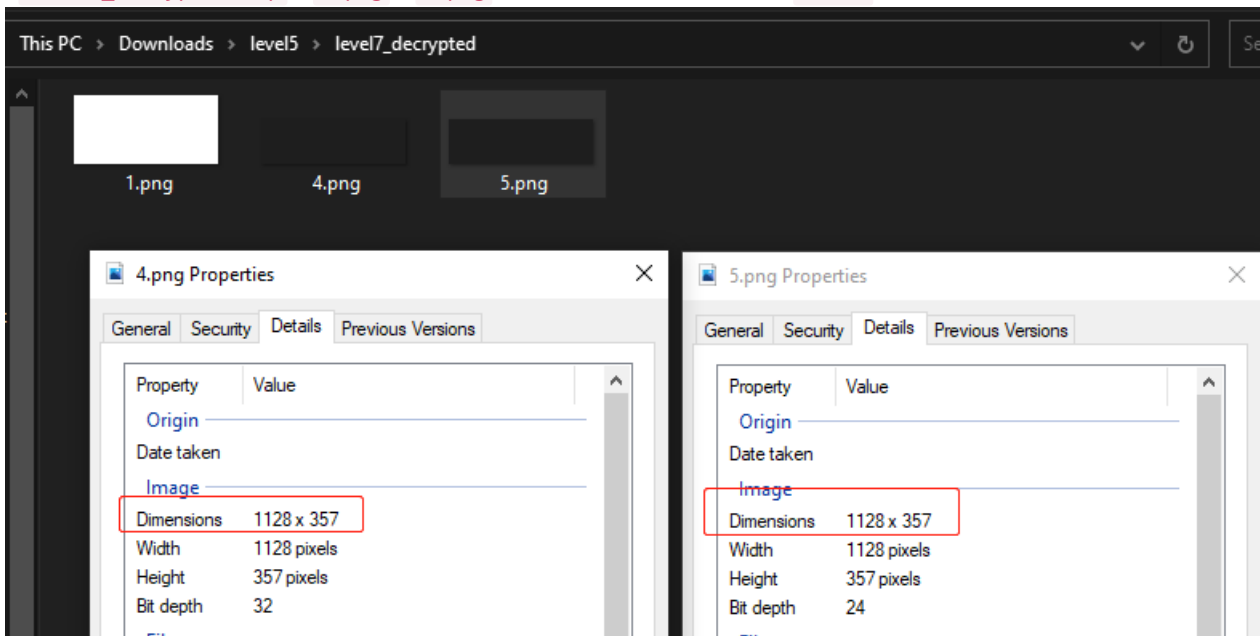


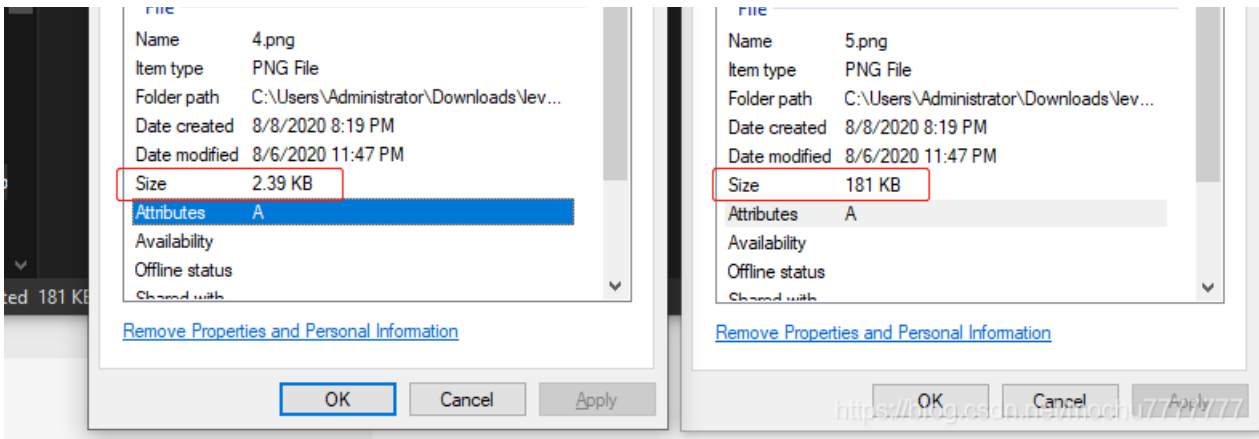
然后会自动保存名为 level7\_decrypted.zip 的已经解开加密的压缩包





直接解压 level7\_decrypted.zip，4.png 和 5.png 分辨率一样，size不一样，盲水印





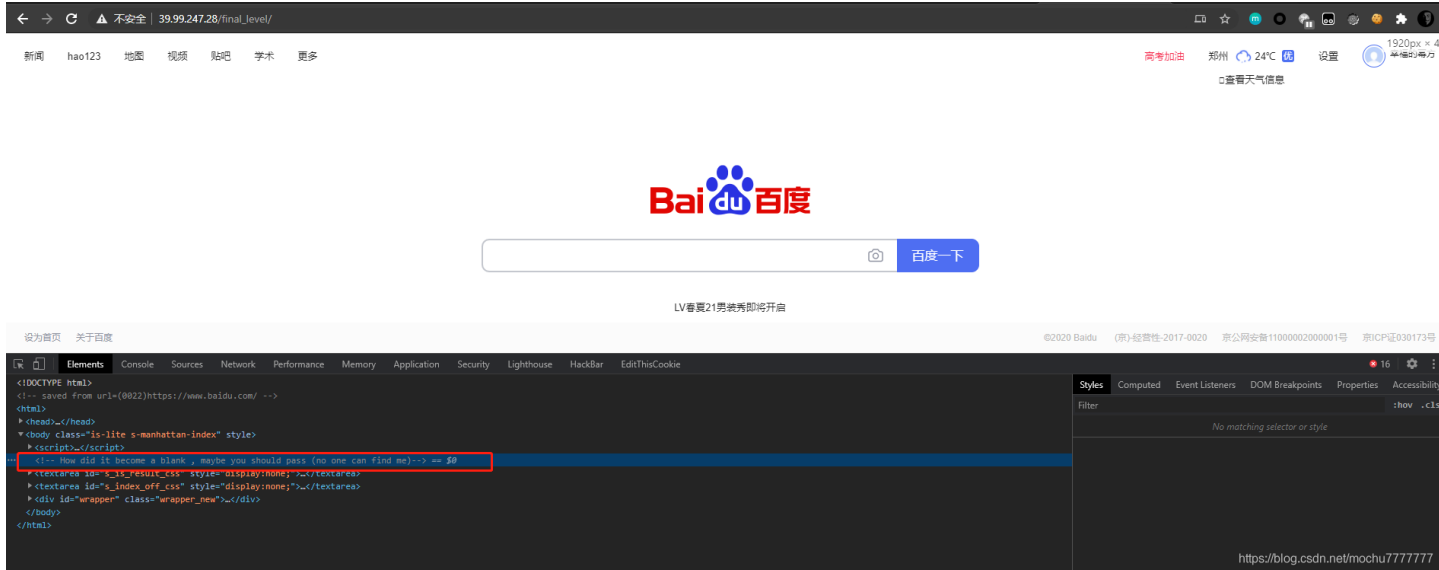
BlindWaterMask: <https://github.com/chishaxie/BlindWaterMark>

```
PS D:\Tools\Misc\BlindWaterMark> python3 .\bwmforpy3.py decode .\4.png .\5.png result.png
image<.\4.png> + image(encoded)<.\5.png> -> watermark<result.png>
```



level7ishere

并且得到最后一关的相关地址: [http://39.99.247.28/final\\_level/](http://39.99.247.28/final_level/)



## html snow隐写

html snow隐写解密网站: <http://fog.misty.com/perry/ccs/snow/snow/snow.html>



把网址格式填对，Password为题目注释中的括号里的内容

## Decryption

URL containing concealed message:

Password:

点击 **Decrypt**



```
the_misc_examaaaaaa_!!!}
```

```
the_misc_examaaaaaa_!!!}
```

综上所述，flag为其部分拼接：

```
flag{level1_begin_and_level2_is_comelevel3_start_itlevel4_here_alllevel5_is_aaalevel6_isreadylevel7isherethe_mis  
c_examaaaaaa_!!!}
```