




2020年江西工业互联网安全技术技能大赛WP

原创

置顶  [lcafe8](#) 于 2020-11-09 22:04:08 发布  1299  收藏 11

分类专栏: [网络安全 CTF](#) 文章标签: [网络安全](#) [python](#) [经验分享](#) [信息安全](#)

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/y920312/article/details/109586883>

版权



[网络安全](#) 同时被 2 个专栏收录

4 篇文章 0 订阅

订阅专栏



[CTF](#)

4 篇文章 0 订阅

订阅专栏

目录

黑客留下的文件

MMS协议分析

常见的对称加密

失窃的文件

findbackdoor

address

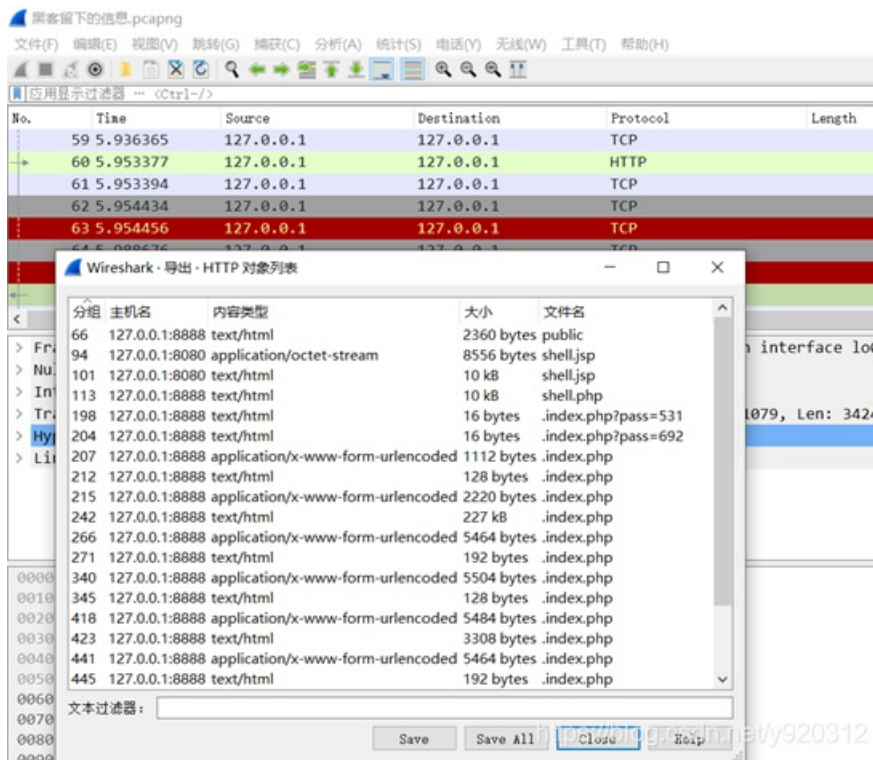
恶意文件分析

黑客留下的文件

1、前置知识

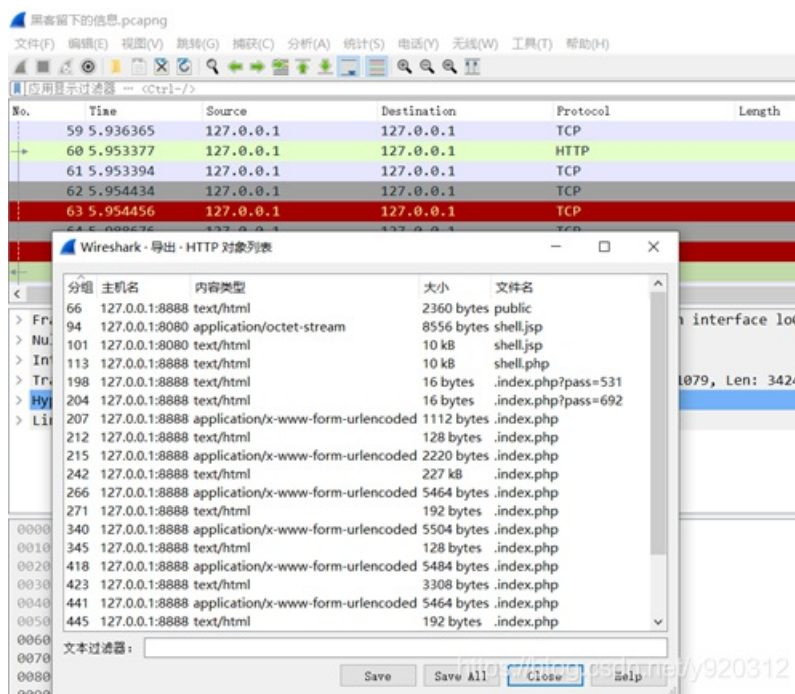
已知新型的php一句话木马如果接收到pass参数, 则会生成16位的随机秘钥, 存储到session中, 返回的内容是AES的密钥。

2、用Wireshark打开后, 首先看其http流量, 发现主要对shell.jsp和.index.php页面进行了请求。



3、通过查看http流量包发现GET /public/.index.php?pass=531和pass632符合新型的木马是随机生产的16位随机密钥。

通过传参pass=531 返回的PHPSESSID=0megcpp047qi2q4357p8ne8ld6 内容是"777ad23134150e3c"



pass=632返回的PHPSESSID=gl5g303u0a6qi7483oqiucnbb0 内容是"ad8d0732eaa9f749"

```

777ad23134150e3cGET /public/.index.php?pass=692 HTTP/1.1
Content-type: application/x-www-form-urlencoded
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/5.0; SLCC2; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0C; .NET4.0E)
Host: 127.0.0.1:8888
Accept: text/html, image/gif, image/jpeg, */*; q=.2, */*; q=.2
Connection: keep-alive

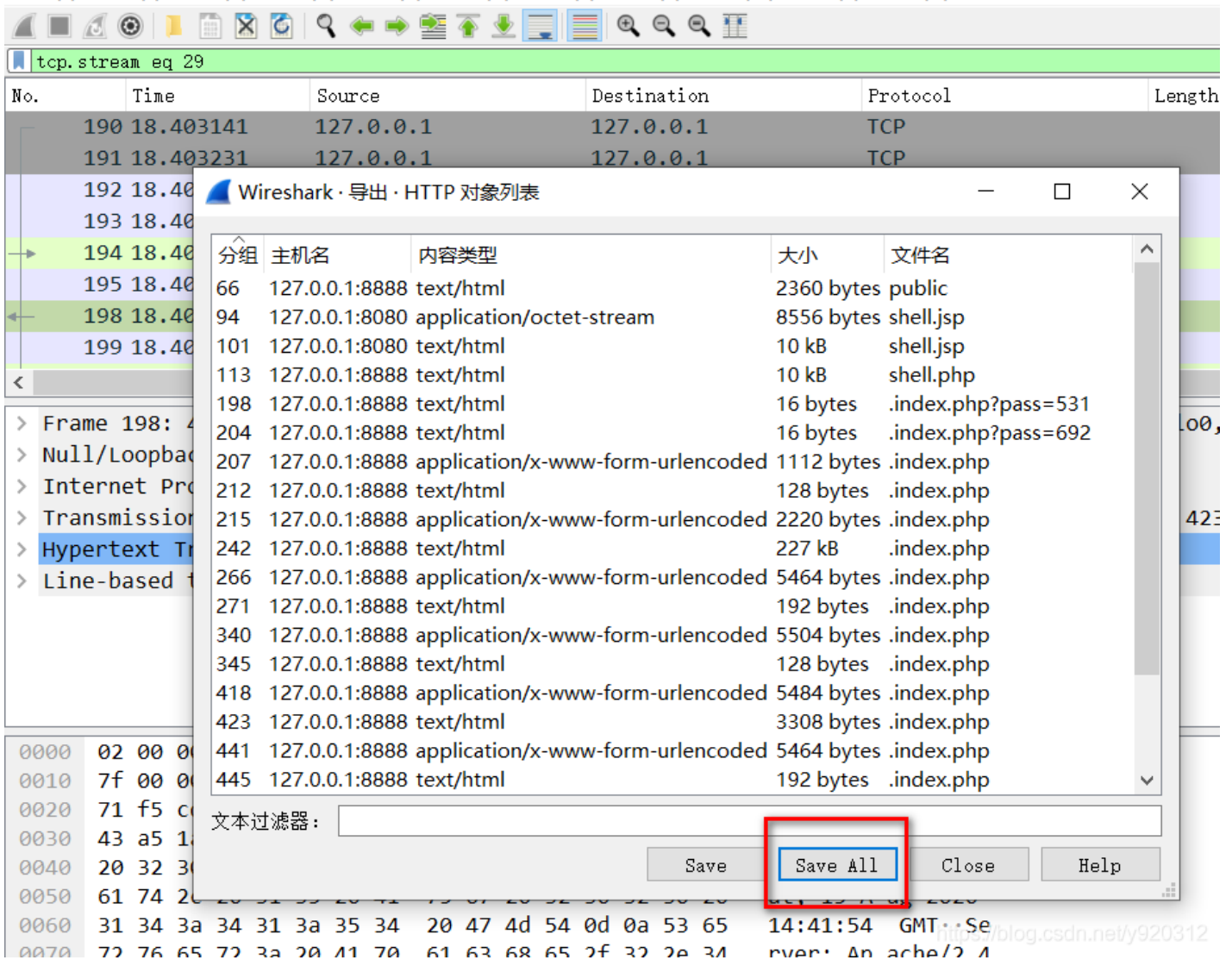
HTTP/1.1 200 OK
Date: Sat, 15 Aug 2020 14:41:54 GMT
Server: Apache/2.4.41 (Unix) PHP/7.3.10
X-Powered-By: PHP/7.3.10
Set-Cookie: PHPSESSID=gl5g303u0a6qi7483oqiucnbb0; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Content-Length: 16
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

ad8d0732eaa9f749POST /public/.index.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Cookie: PHPSESSID=gl5g303u0a6qi7483oqiucnbb0; path=/
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/5.0; SLCC2; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; InfoPath.3; .NET4.0C; .NET4.0E)
Cache-Control: no-cache
Pragma: no-cache
Host: 127.0.0.1:8888
Accept: text/html, image/gif, image/jpeg, */*; q=.2, */*; q=.2
Connection: keep-alive
Content-Length: 1112

```

通过查询两个PHPSESSID的值发现后面基本是PHPSESSID=gl5g303u0a6qi7483oqiucnbb0，所以猜测pass为692。

4、然后导出所有的HTTP文件发现大部分是.index.php的文件



5、导出之后的.index.php使用PHP脚本进行循环遍历，通过查看发现shell.php没啥有用的内容，有用的基本都在.index.php中，通过脚本去遍历输出.index.php解密的内容，脚本如下：

MMS协议分析

1、前置知识

1 PDU Type:无符号整型, 1byte, 标记状态, 注意上图中这行后面的0x0e, 代表连接请求, 还有其他类型如下所示。

0x1: ED Expedited Data, 加急数据

0x2: EA Expedited Data Acknowledgement, 加急数据确认

0x4: UD, 用户数据

0x5: RJ Reject, 拒绝

0x6: AK Data Acknowledgement, 数据确认

0x7: ER TPDU Error, TPDU错误

0x8: DR Disconnect Request, 断开请求

0xC: DC Disconnect Confirm, 断开确认

0xD: CC Connect Confirm, 连接确认

0xE: CR Connect Request, 连接请求

0xF: DT Data, 数据传输

四种MMS包, 分别是:

initiate-RequestPDU(启动-请求PDU)

confirmed-RequestPDU(确认-请求PDU)

initiate-ResponsePDU(启动-应答PDU)

confirmed-ResponsePDU(确认-应答PDU)

2、打开mms流量文件, 筛选mms协议流量包

No.	Time	Source	Destination	Protocol	Length	Info
827	249.459641	192.168.142.148	192.168.142.133	MMS	260	initiate-RequestPDU
828	249.459865	192.168.142.133	192.168.142.148	MMS	199	initiate-ResponsePDU
834	253.408009	192.168.142.148	192.168.142.133	MMS	119	01 confirmed-RequestPDU
835	253.408309	192.168.142.133	192.168.142.148	MMS	85	01 confirmed-ResponsePDU
837	253.891101	192.168.142.148	192.168.142.133	MMS	123	02 confirmed-RequestPDU
838	253.891384	192.168.142.133	192.168.142.148	MMS	85	02 confirmed-ResponsePDU
840	254.427026	192.168.142.148	192.168.142.133	MMS	118	03 confirmed-RequestPDU
841	254.427297	192.168.142.133	192.168.142.148	MMS	85	03 confirmed-ResponsePDU
843	255.357508	192.168.142.148	192.168.142.133	MMS	119	04 confirmed-RequestPDU
844	255.357730	192.168.142.133	192.168.142.148	MMS	85	04 confirmed-ResponsePDU
853	258.890930	192.168.142.148	192.168.142.133	MMS	123	05 confirmed-RequestPDU
854	258.891207	192.168.142.133	192.168.142.148	MMS	85	05 confirmed-ResponsePDU
858	259.688983	192.168.142.148	192.168.142.133	MMS	122	06 confirmed-RequestPDU

> Frame 827: 260 bytes on wire (2080 bits), 260 bytes captured (2080 bits)
> Ethernet II, Src: VMware_1d:07:17 (00:0c:29:1d:07:17), Dst: VMware_75:b2:38 (00:0c:29:75:b2:38)
> Internet Protocol Version 4, Src: 192.168.142.148, Dst: 192.168.142.133
> Transmission Control Protocol, Src Port: 4078, Dst Port: 102, Seq: 23, Ack: 15, Len: 206
> TPCT, Version: 3, Length: 206
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
> ISO 8327-1 OSI Session Protocol
> ISO 8823 OSI Presentation Protocol
> ISO 8650-1 OSI Association Control Service
> MMS

<https://blog.csdn.net/y920312>

3、发现是一个数据的传输过程。DT表示数据传输

```

> Transmission Control Protocol, Src Port: 4078, Dst Port: 102, ...
> TPKT, Version: 3, Length: 206
v ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
  Length: 2
  PDU Type: DT Data (0x0f)
  [Destination reference: 0x420000]
  .000 0000 = TPDU number: 0x00
  
```

首先是两个数据包是

initiate-RequestPDU(启动-请求PDU)

initiate-ResponsePDU(启动-应答PDU)

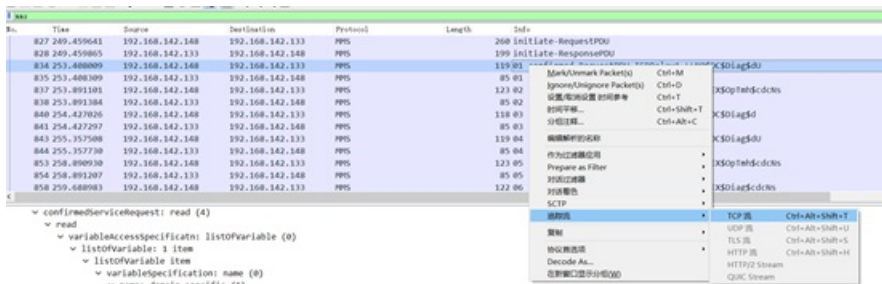
随后开始传输数据。

4、仔细观察流量包，发现confirmed-RequestPDU，发现字段和值都为以下格式

mms.domainId: IEDRelay1

mms.itemId: LLN0\$xx\$xx\$xx

5、没有思路，追踪TCP流



浏览TCP流，发现几个异常的数据包

```

...F.....a907...2.0...*(.&0$. "... IEDRelay1..LLN0$EX$OpTmh$cdcNs... ..a.0...
...F.....a907...2.0...*(.&0$. "... IEDRelay1..LLN0$EX$OpTmh$cdcNs... ..a.0...
...F.....a907...2.0...*(.&0$. "... IEDRelay1..LLN0$EX$OpTmh$cdcNs... ..a.0...
...F.....a907...2.0...*(.&0$. "... IEDRelay1..LLN0$EX$OpTmh$cdcNs... ..a.0...
...F.....a907...2.0...*(.&0$. "... IEDRelay1..LLN0$EX$OpTmh$cdcNs... ..a.0...
...F.....a907...2.0...*(.&0$. "... IEDRelay1..LLN0$EX$OpTmh$cdcNs... ..a.0...
...F.....a907...2.0...*(.&0$. "... IEDRelay1..LLN0$EX$OpTmh$cdcNs... ..a.0...
...F.....a907...2.0...*(.&0$. "... IEDRelay1..LLN0$EX$OpTmh$cdcNs... ..a.0...
  
```

```

./.-.+0).'.%. IEDRelay1..LLN0$DC$NamPlt$configRev... ..a.0...
...*(.&0$. "... IEDRelay1..LLN0$EX$OpTmh$cdcNs... ..a.0...
./.-.+0).'.%. IEDRelay1..LLN0$DC$NamPlt$configRev... ..a.0...
...*(.&0$. "... IEDRelay1..LLN0$EX$OpTmh$cdcNs... ..a.0...
...*(.&0$. "... IEDRelay1..LLay7sxCA9wSYrVLCbr... ..a.0...
...*(.&0$. "... IEDRelay1..LLN0$EX$OpTmh$cdcNs... ..a.0...
...*(.&0$. "... IEDRelay1..LLN0$EX$OpTmh$cdcNs... ..a.0...
./.-.+0).'.%. IEDRelay1..LLN0$DC$NamPlt$configRev... ..a.0...
./.-.+0).'.%. IEDRelay1..LLN0$DC$NamPlt$configRev... ..a.0...
  
```

```

./.-.+0).'.%. IEDRelay1..LLN0$DC$NamPlt$configRev... ..a.0...
...*(.&0$. "... IEDRelay1..LLN0$EX$OpTmh$cdcNs... ..a.0...
./.-.+0).'.%. IEDRelay1..LLN0$DC$NamPlt$configRev... ..a.0...
...*(.&0$. "... IEDRelay1..LLN0$EX$OpTmh$cdcNs... ..a.0...
...*(.&0$. "... IEDRelay1..LLay7sxCA9wSYrVLCbr... ..a.0...
...*(.&0$. "... IEDRelay1..LLN0$EX$OpTmh$cdcNs... ..a.0...
...*(.&0$. "... IEDRelay1..LLN0$EX$OpTmh$cdcNs... ..a.0...
./.-.+0).'.%. IEDRelay1..LLN0$DC$NamPlt$configRev... ..a.0...
./.-.+0).'.%. IEDRelay1..LLN0$DC$NamPlt$configRev... ..a.0...
  
```

分别为

3189流量包: LLN666i5250356j4249

3196流量包: LLN616732557968356j

4678流量包: LLAy7sxCA9wSYrVLCbr

6、猜测前三位为固定格式，观察发现666i5250356j4249，616732557968356j。类似于hex编码，但是i，j在16进制不存在，所以进行替换爆破。

```
import binascii

s1 = '666i5250356j4249'
s2 = '616732557968356j'

s = ['a', 'b', 'c', 'd', 'e', 'f']

for i in range(len(s)-1):
    tmp2 = s1.replace('i',s[i])
    tmp2 = tmp2.replace('j',s[i+1])
    print (tmp2.decode('hex'))
    tmp1 = s2.replace('j',s[i+1])
    print (tmp1.decode('hex'))
    print ('-----')
```

执行结果如下:

```
PS D:\Code> & "D:/Program Files/Python/Python27/python.exe" d:/Code/1.py
fjRP5kBI
ag2Uyh5k
-----
fkRP5lBI
ag2Uyh5l
-----
flRP5mBI
ag2Uyh5m
-----
fmRP5nBI
ag2Uyh5n
-----
fnRP5oBI
ag2Uyh5o
-----
```

发现如下爆破结果可以进行拼接组合得到flag

```
flRP5mBI
ag2Uyh5m
```

flag: **flag{flagRP5mBI2Uyh5m}**

常见的对称加密

1、观察所给文件中的s盒，发现与pyDes库中的s盒不同，遂替换里面的__sbox

```
# 1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2,
# 6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12],
#
## S8
# [13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7,
# 1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2,
# 7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8,
# 2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11],

# S1
[14, 13, 4, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7,
0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8,
4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0,
15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13],
# S2
[15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10,
3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5,
0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15,
13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9],
# S3
[10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8,
```

2、明文密码给了welcome，直接DES解密就好了，中间还有个unicode编码，直接用binascii库解码就可以了

EXP脚本如下：

```
# -*- coding:utf-8 -*-
import pyDes,binascii
c='a669ca04e31c244d9dd2decc1f2678a4624986567b25af5d'
key='welcome'.encode()+b'\x00'
print(key)
d=pyDes.des(key,'ECB')

print(d.decrypt(binascii.unhexlify(c)))
```

运行结果如下：



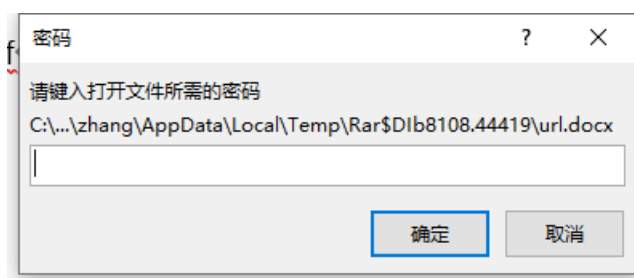
```
Run: des(1) x
C:\Users\zhang\AppData\Local\Programs\Python\Python38\p
b'welcome\x00'
b'flag{JXICS2020ISBEST}\x00\x00\x00'
Process finished with exit code 0
```

<https://blog.csdn.net/y920312>



得到压缩包密码：**aceduf**

5、对压缩包进行解密，发现word文件加密，还需要密码才能打开。



6、IDA定位关键代码

```

GetCurrentDirectoryA(0x208u, &Buffer);
sub_4091A0(&Buffer, "%s\\%s", &Buffer);
if ( !ISASStartup(0x202u, &WSADATA) && LOBYTE(WSADATA.wVersion) == 2 && HIBYTE(WSADATA.wHighVersion) == 2 )
{
while ( 1 )
{
Sleep(0x927C0u);
if ( !InternetCheckConnectionA("http://whois.chinaz.com", 1u, 0)
&& !InternetCheckConnectionA("http://www.baidu.com", 1u, 0)
&& !InternetCheckConnectionA("http://sirilagz.net", 1u, 0) )
{
break;
}
sub_407FC0(&Buffer);
v4 = &pszIn;
if ( pszIn )
{
do
*v4++ ^= 0x91u;
while ( *v4 );
}
pchOut = 4096;
!Is16BitData2(&pszIn, &pszOut, &pszOut, 0u, 0);
}
}

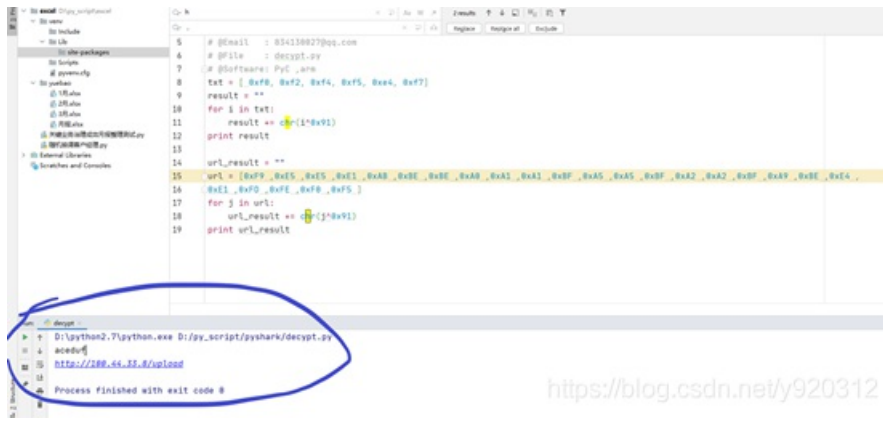
```

```

.data:0042A9F8 pszIn          db 0F9h          ;
.data:0042A9F8                db 0E5h          ;
.data:0042A9F9                db 0E5h
.data:0042A9FA                db 0E1h
.data:0042A9FB                db 0A0h
.data:0042A9FC                db 0BEh
.data:0042A9FD                db 0BEh
.data:0042A9FE                db 0A0h
.data:0042A9FF                db 0A1h
.data:0042AA00                db 0A1h
.data:0042AA01                db 0BFh
.data:0042AA02                db 0A5h
.data:0042AA03                db 0A5h
.data:0042AA04                db 0BFh
.data:0042AA05                db 0A2h
.data:0042AA06                db 0A2h
.data:0042AA07                db 0BFh
.data:0042AA08                db 0A9h
.data:0042AA09                db 0BEh
.data:0042AA0A                db 0E4h
.data:0042AA0B                db 0E1h
.data:0042AA0C                db 0FDh
.data:0042AA0D                db 0FEh
.data:0042AA0E                db 0F0h
.data:0042AA0F                db 0F5h

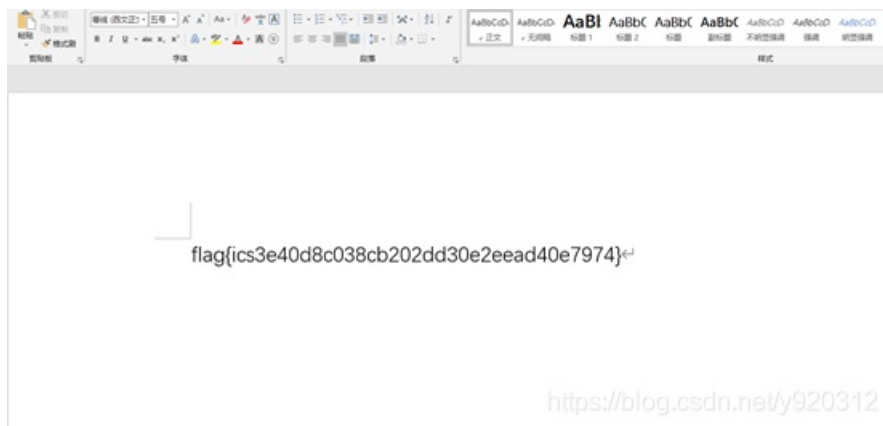
```

EXP解码如下:



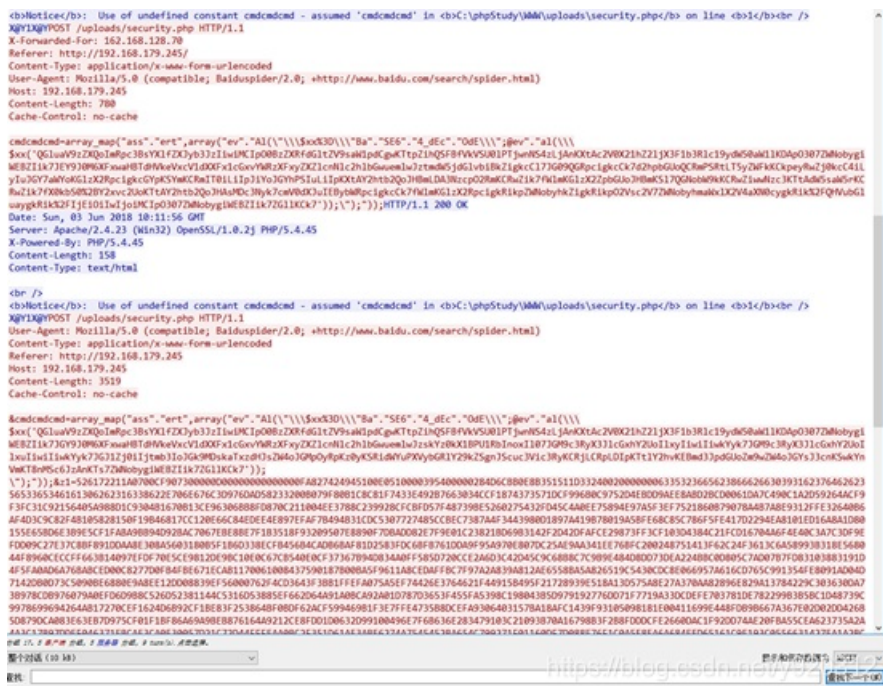
得到解密密码为: **http://100.44.33.8/upload**

7、输入密码, 得到flag.



findbackdoor

1、使用wireshark查看流量有点像菜刀流量



2、通过base64解密可知算法如下, post参数z1传参:

```
@ini_set("display_errors","0"); @set_time_limit(0); if(PHP_VERSION<'5.3.0') {@set_magic_quotes_runtime(0);}
echo("X@Y"); $f='C:\phpStudy\WWW\uploads\reverseshell.zip'; $c=$_POST["z1"]; $c=str_replace("\r","", $c);
$c=str_replace("\n","", $c); $buf=""; for($i=0;$i<strlen($c);$i+=2) $buf.=urdecode('%'.substr($c,$i,2));
echo(@fwrite(fopen($f,'w'),$buf)?'1':'0');; echo("X@Y"); die();
```

3、定位z1所在的流量包，发现为第22个流量包

The image shows a Wireshark capture of network traffic. In the packet list pane, packet 22 is highlighted. The packet details pane shows the following information:

- [Checksum Status: Unverified]
- Urgent pointer: 0
- [SEQ/ACK analysis]
- [Timestamps]
- TCP payload (1460 bytes)
- [Reassembled PDU in frame: 25]
- TCP segment data (1460 bytes)

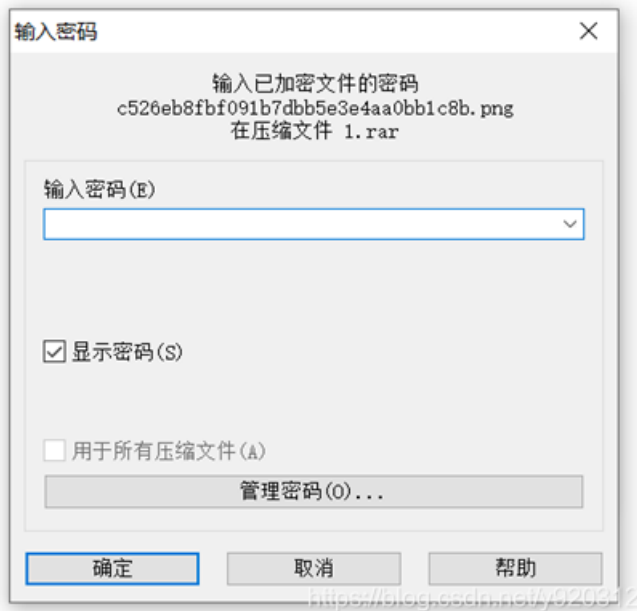
The packet bytes pane shows the raw data of the selected packet, with a red arrow pointing to the start of the payload at offset 0030.

4、将流量包数据复制到HxD，保存为rar文件。

The image shows a screenshot of the HxD hex editor. The editor displays the raw data of the selected packet in hexadecimal and ASCII format. The data is displayed in a grid with columns for offset, hexadecimal, and ASCII. The ASCII column shows the following text:

```
Rar!...I.s.....
...ù,t$"Q.àQ..9
T...œ*è"QQ.3$.
...c52éeb8fbf09
1b7dbb5e3e4aa0bb
1c8b.pngl-m.X#2
.ye..E.+C>I+vc.
Ll.t75qÙ-èu-Ns
Ùè«0«0..$À.À«0'
d-ùóú1É!V0Z...
eg..i-0k,y+...i
```

5、解压缩需要密码



6、可以看到压缩包里面的文件名为c526eb8fbf091b7dbb5e3e4aa0bb1c8b.png

猜测是MD5通过在线解密得到明文ics2020



7、获得口令解压得到一张图片，发现是二维码，但是只有部分。



8、放入010编辑器，修改图片高度257为500，修复图片到二维码图片。

```

0180h: 18 26 31 B0 60 31 0C 93 18 58 B0 18 86 49 0C 2C .&1°1.".X°.tI.,
0190h: 58 0C C3 24 06 16 2C 86 61 12 03 0B 16 C3 30 89 X.Ä?.,fa...Ä0%
01A0h: 81 05 8B 61 98 C4 C0 82 C5 30 4C 62 60 C1 62 18 ..<a"AA,Ä0Lb`Äb.
01B0h: 26 31 E8 0B 7D 01 4C AB E0 48 BA 5C 52 15 8F EC &1è.1.L&àH^R..ü

```

Name	Value	Start	Size	Color	Comment
struct PNG_SIGNATURE ...		0h	8h	Fg: Bg:	
struct PNG_CHUNK chun...	IHDR (Critic...	8h	19h	Fg: Bg:	
uint32 length	13	8h	4h	Fg: Bg:	
union CTYFE type	IHDR	Ch	4h	Fg: Bg:	
struct PNG_CHUNK_IH...	400 x 257 (x8)	10h	10h	Fg: Bg:	
uint32 width	400	10h	4h	Fg: Bg:	
uint32 height	257	14h	4h	Fg: Bg:	
ubyte bits	8	18h	1h	Fg: Bg:	
enum PNG_COLOR_SP...	TrueColor (2)	19h	1h	Fg: Bg:	
enum PNG_COMPRESSION...	Deflate (0)	1Ah	1h	Fg: Bg:	

https://blog.csdn.net/y920312



https://blog.csdn.net/y920312

9、扫码得到flag

纠错等级: H(30%)
版本: Auto
 Auto

已解码数据 1:

```

-----
位置:(210.3,166.9)-(464.0,167.0)-(210.2,420.7)-(464.2,420.6)
颜色正常,正像
版本:3
纠错等级:H,掩码:0
内容:
flag{Jxciit_ICS_zip_zip}
-----

```

https://blog.csdn.net/y920312

flag: flag{Jxciit_ICS_zip_zip}

address

1、前置知识。

104字段基本内容

```
▼ IEC 60870-5-104-Apci: <- I (0,0)
  START
  ApduLen: 14
  .... ..0 = Type: I (0x00)
  Tx: 0
  Rx: 0
▼ IEC 60870-5-104-Asdu: ASDU=10 C_IC_NA_1 Act IOA=0 'interrogation command'
  typeId: C_IC_NA_1 (100)
  0... .... = SQ: False
  .000 0001 = NumIx: 1
  ..00 0110 = CauseTx: Act (6)
  .0.. .... = Negative: False
  0... .... = Test: False
  OA: 0
  Addr: 10
▼ IOA: 0
  IOA: 0
  QOI: Station interrogation (global) (20)
```

帧格式

I:0

S:01

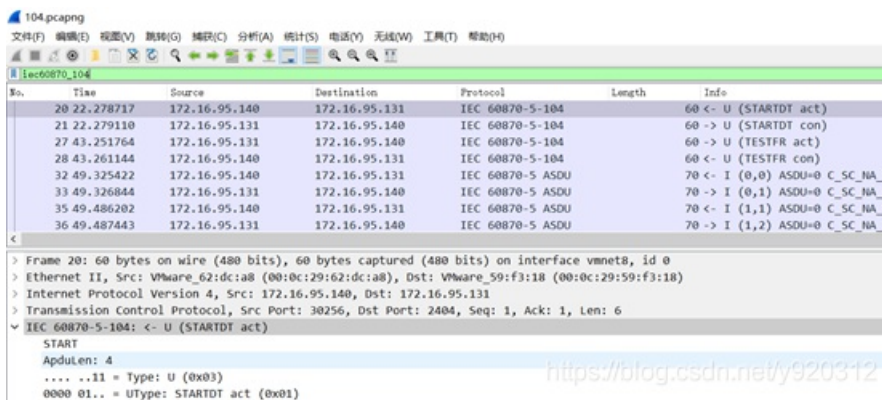
U:11

操作对象: Typeld

公共地址: Addr

信息体地址: IOA

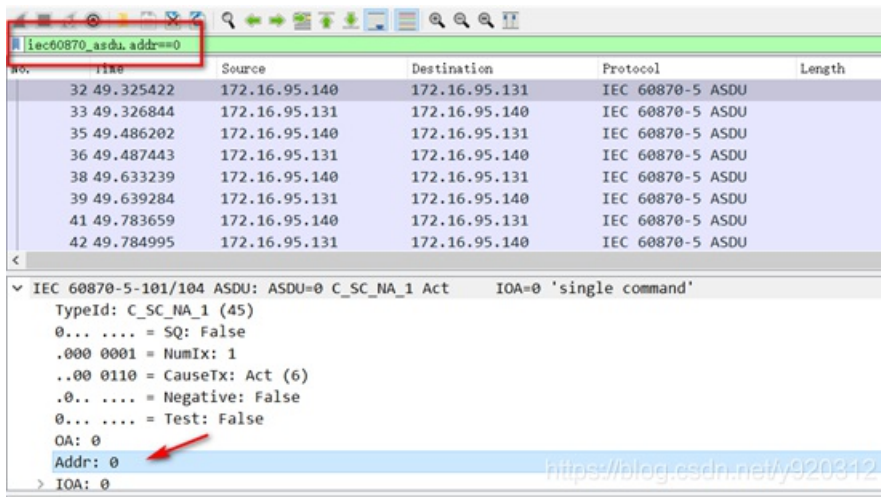
2、打开流量文件，筛选104协议的流量包



3、浏览数据包，发现ASDU存在多种值，其中ASDU=0最多。ASDU即是Addr的值。

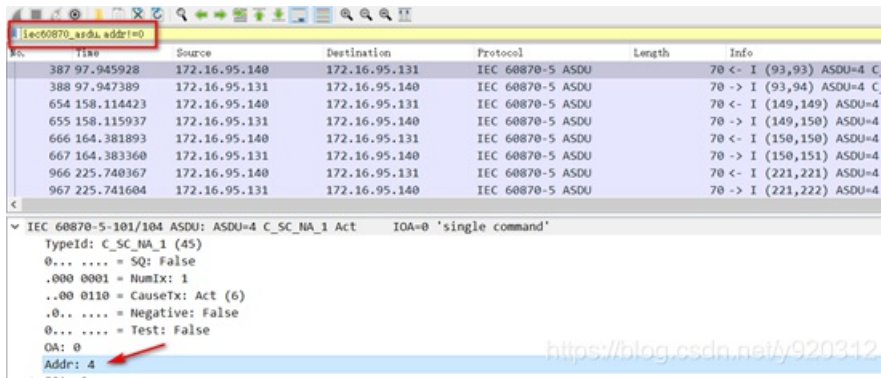
复制addr字段ie60870_asdu.addr，对addr进行筛选。

```
ie60870_asdu.addr==0
```

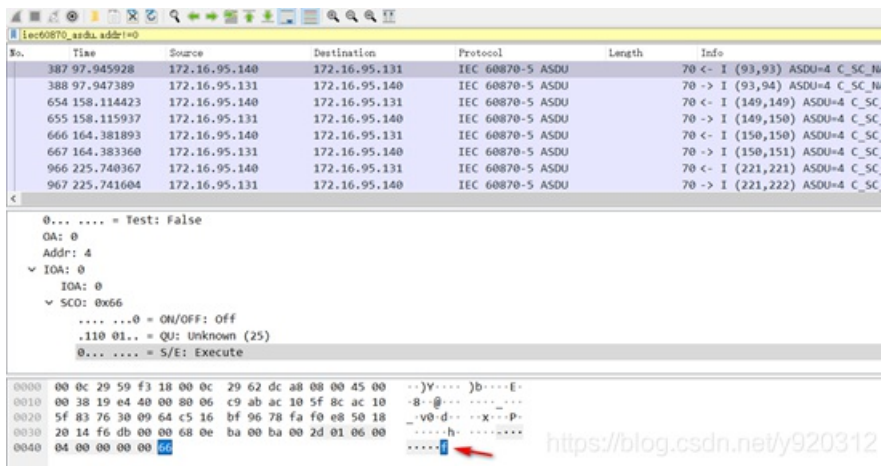


发现流量包比较多，大致浏览未发现有效信息。

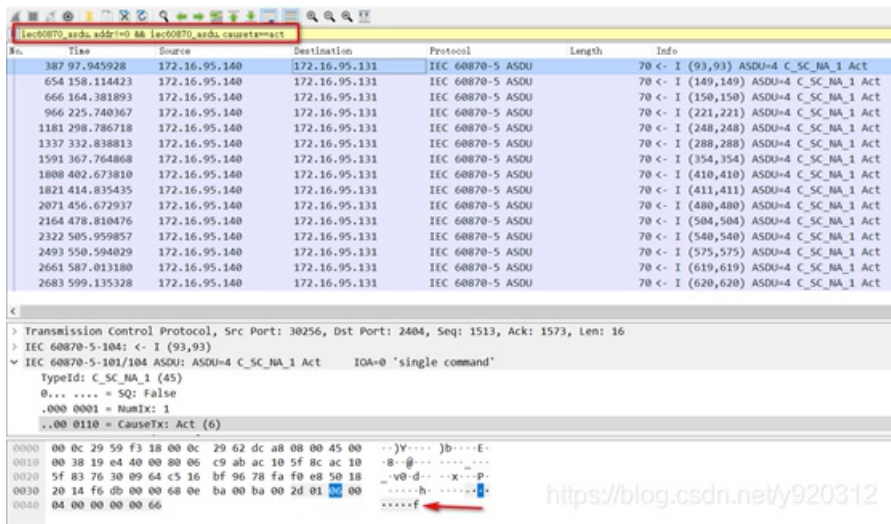
4、接着筛选`iec60870_asdu.addr!=0`,发现剩余的都是 (ASDU=4) `iec60870_asdu.addr==4`的流量包。总共只有30个流量包。



浏览数据包，发现act包数据都有一个字符，浏览前四个act包，发现组成起来为flag。



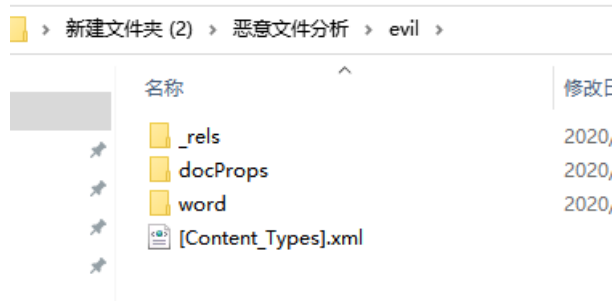
5、继续筛选act值的字段，`iec60870_asdu.causetx==act`，发现flag为：



6、组合起来，得到flag为：**flag{jx104_biu}**

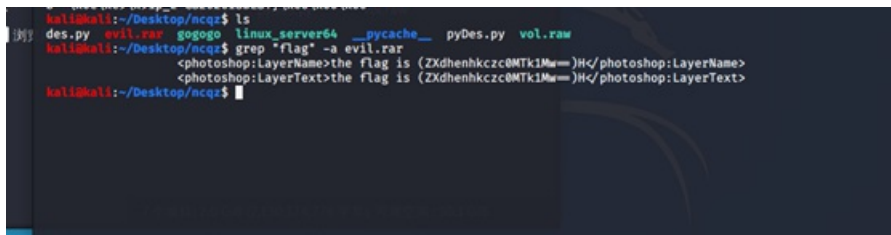
恶意文件分析

1、恶意文件分析解压出来发现是evil.rar再解压是



2、可知其有很多字符串，把evil.rar拖到Kali读取所有字符串通过grep筛选关键词发现有关键信息

the flag is (ZXdhenkczc0MTk1Mw==)H



ZXdhenkczc0MTk1Mw==通过base64解密得到ewazxds741953

ewazxds通过键盘的位置可以看出是一个"G"

741953通过键盘的位置可以看出是一个"K"



GK转十六进制是474B

所以flag为flag{474B}