

2019工业信息安全技能大赛个人线上赛第二场一共十道题，本人对做出的五道题目进行分析。

第一场：<https://xz.aliyun.com/u/21528>

### 1.1.破解加密数据

题目：某工控厂商自行研发了一套加密系统，这样的话只要不是系统内部人员，即使数据被窃听没关系了。你截获了一段密文：109930883401687215730636522935643

题目附件连接：[https://pan.baidu.com/s/1bKVpT\\_umaEt2fkqrlwmUKw](https://pan.baidu.com/s/1bKVpT_umaEt2fkqrlwmUKw)（提取码：tseo）

解题步骤：

打开附件，是一个代码段，分析代码，明文使用了Rabin加密，[Rabin加密](#)

```
m = "109930883401687215730636522935643539707"
e = 2
n = 0x6FBD096744B2B34B423D70C8FB19B541
assert(int(m.encode('hex'), 16) < n)
c = pow(int(m.encode('hex'), 16), e, n)
print c
#109930883401687215730636522935643539707
```

2. 通过资料了解加密方式后，编写解密脚本，将n分解出p和q，解密脚本如下：

```
import gmpy2
c=109930883401687215730636522935643539707
n=0x6fbd096744b2b34b423d70c8fb19b541
p=10848126018911527523
q=13691382465774455051
#■■■c^(1/2)modp,q■■■
r=pow(c, (p+1)/4,p)
s=pow(c, (q+1)/4,q)
#■■■■■■■■■■
pni=int(gmpy2.invert(p,q))
qni=int(gmpy2.invert(q,p))
a=(s*p*pni+r*q*qni)%n
a1=n-a
b=(s*p*pni-r*q*qni)%n
b1=n-b
print (['0', ''][len(hex(a))%2]+hex(a)[2:-1]).decode("hex")
print (['0', ''][len(hex(a1))%2]+hex(a1)[2:-1]).decode("hex")
print (['0', ''][len(hex(b))%2]+hex(b)[2:-1]).decode("hex")
print (['0', ''][len(hex(b1))%2]+hex(b1)[2:-1]).decode("hex")
```

3. 运行脚本得到Flag, Flag为flag\_EnCryp1

```
root@kali:~# python e.py
$0AVm0XjA00e
K0%0D0000XFf00
o0mbola f0FQ0000000E -WL,-z,relro -fno-strict-al
flag_Encryp1 -D FORTIFY_SOURCE=2 -g -fdebu
```


## 1.2.工控安全取证

题目：有黑客入侵了工控设备后再内网发起大量扫描，而且扫描次数不止一次。分析日志指出第四次发起扫描时数据包的编号，flag形式为{}

题目附件连接：[https://pan.baidu.com/s/1MEhe8A1htS0ew9Pav\\_JaUA](https://pan.baidu.com/s/1MEhe8A1htS0ew9Pav_JaUA)（提取码：q73d）

解题步骤：

2 > question 1565019927 2-2 > 工控安全取证

 [captur.pcap](#)

A horizontal toolbar containing 18 icons for document editing and navigation. The icons include: a corner crop tool, a square crop tool, a circular crop tool, a settings gear, a yellow folder, a document with a list, a document with a red X, a document with a blue C, a magnifying glass, left and right arrow keys, a double left arrow, a double right arrow, a double up arrow, a double down arrow, a document with a blue arrow, a document with a red arrow, a magnifying glass with a plus sign, a magnifying glass with a minus sign, a magnifying glass with an equals sign, and a table icon.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.9	192.168.0.99	ICMP	60	Echo (ping)
2	0.000078	192.168.0.99	192.168.0.9	ICMP	42	Echo (ping)
3	0.000044	192.168.0.9	192.168.0.99	TCP	60	52218 → 80
4	0.000119	192.168.0.99	192.168.0.9	TCP	54	80 → 52218
5	10.346091	192.168.0.9	192.168.0.99	TCP	60	52198 → 52
6	10.346199	192.168.0.99	192.168.0.9	TCP	54	52156 → 52
7	10.346137	192.168.0.9	192.168.0.99	TCP	60	52198 → 28
8	10.346235	192.168.0.99	192.168.0.9	TCP	54	28494 → 52
9	10.346167	192.168.0.9	192.168.0.99	TCP	60	52198 → 11
10	10.346246	192.168.0.99	192.168.0.9	TCP	54	11170 →

icmp

	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.9	192.168.0.99	ICMP	60	Echo (ping) request id=0x7ae9, seq=0/0, ttl=40 (reply in 2)
2	0.000078	192.168.0.99	192.168.0.9	ICMP	42	Echo (ping) reply id=0x7ae9, seq=0/0, ttl=255 (request in 1)
148007	1274.602300	192.168.0.9	192.168.0.99	ICMP	60	Echo (ping) request id=0x1e09, seq=0/0, ttl=47 (reply in 148008)
148008	1274.602365	192.168.0.99	192.168.0.9	ICMP	42	Echo (ping) reply id=0x1e09, seq=0/0, ttl=255 (request in 148007)
150655	1308.472790	192.168.0.99	192.168.0.9	ICMP	370	Destination unreachable (Port unreachable)
150753	1407.256096	192.168.0.9	192.168.0.99	ICMP	60	Echo (ping) request id=0xa373, seq=0/0, ttl=53 (reply in 150754)
150754	1407.256145	192.168.0.99	192.168.0.9	ICMP	42	Echo (ping) reply id=0xa373, seq=0/0, ttl=255 (request in 150753)
153165	1441.428990	192.168.0.99	192.168.0.9	ICMP	370	Destination unreachable (Port unreachable)
155847	1504.127684	192.168.0.99	192.168.0.9	ICMP	370	Destination unreachable (Port unreachable)
155987	1602.084879	192.168.0.1	192.168.0.99	ICMP	60	Echo (ping) request id=0xc77b, seq=0/0, ttl=52 (no response found)
155988	1602.084912	192.168.0.254	192.168.0.99	ICMP	60	Echo (ping) request id=0xc77b, seq=0/0, ttl=52 (no response found)
155989	1602.084941	192.168.0.199	192.168.0.99	ICMP	60	Echo (ping) request id=0xc77b, seq=0/0, ttl=52 (no response found)
155990	1602.084976	192.168.0.199	192.168.0.99	ICMP	60	Echo (ping) request id=0xc77b, seq=0/0, ttl=52 (no response found)

```

▼ Frame 155989: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
  Encapsulation type: Ethernet (1)
  Arrival Time: Aug 27, 2002 08:26:32.335700000 中国标准时间
  [Time shift for this packet: 0.000000000 seconds]
  Epoch Time: 1030407992.335700000 seconds
  [Time delta from previous captured frame: 0.000029000 seconds]
  [Time delta from previous displayed frame: 0.000029000 seconds]
  [Time since reference or first frame: 1602.084941000 seconds]
  Frame Number: 155989
  Frame Length: 60 bytes (480 bits)
  Capture Length: 60 bytes (480 bits)

```

**解题步骤：**

打开附件是一个文件，在linux中使用file指令查看文件是什么类型，发现文件是一个windows的PE文件。

f67b65b9346ee75a26f491b70bf6091b

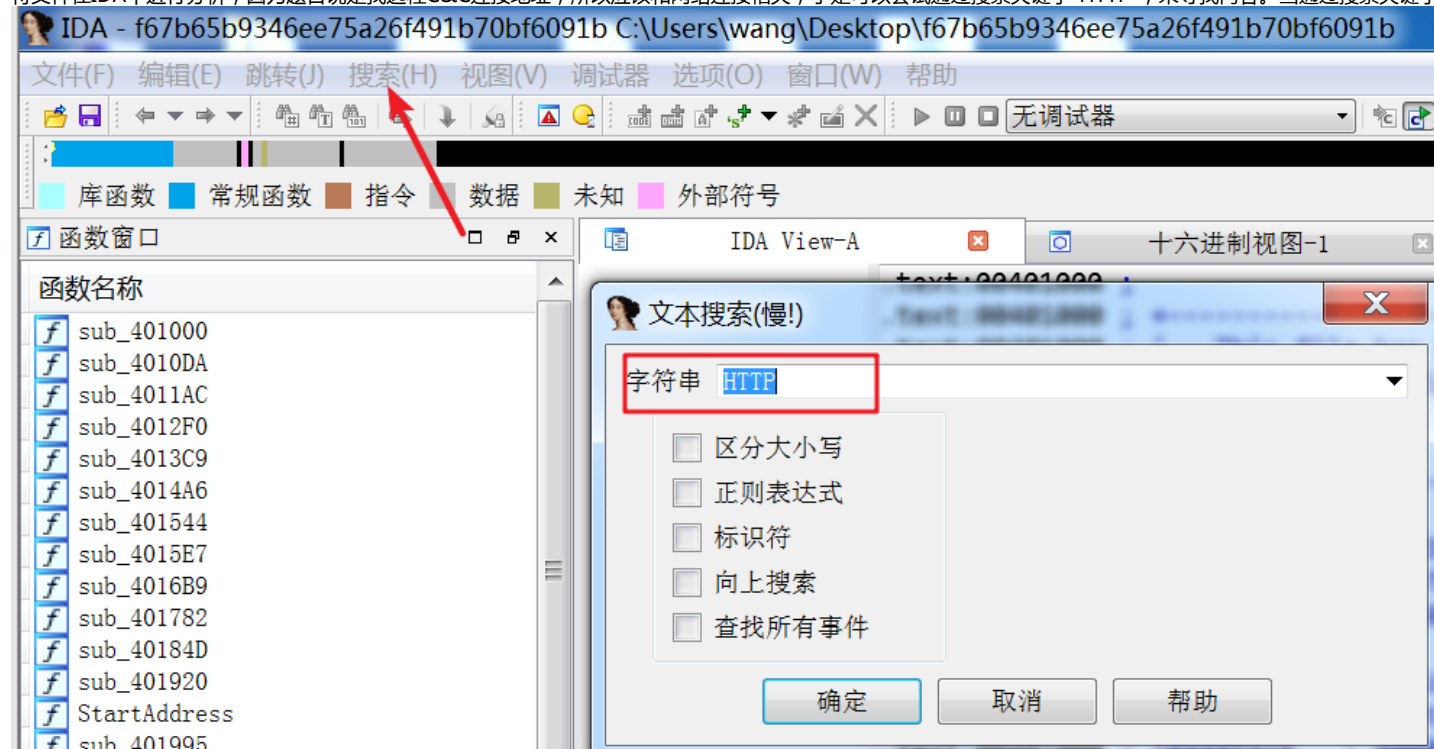
2017/6/14 15:21

文件

11 KB

```
root@kali:~# file f67b65b9346ee75a26f491b70bf6091b
f67b65b9346ee75a26f491b70bf6091b: PE32 executable (GUI) Intel 80386, for MS Windows
```

将文件在IDA中进行分析，因为题目说是找远程C&C连接地址，所以应该和网络连接相关，于是可以尝试通过搜索关键字"HTTP"，来寻找内容。当通过搜索关键字"HTTP"



```
.text:00401EC6
.text:00401ECB
.text:00401ECE
.text:00401ED1
.text:00401ED6
.text:00401ED9
.text:00401EDB
.text:00401EDE
.text:00401EE4
.text:00401EE7
.text:00401EEA
.text:00401EED
.text:00401EF0
```

```
call    sub_401B50
add     esp, 0Ch
mov     [ebp+var_C], eax
call    sub_402383
mov     edi, [ebp+hRequest]
mov     ecx, eax
mov     eax, [ebp+dwNumberOfBytesToWrite]
mov     esi, ds:WinHttpWriteData
add     eax, 14h
mov     [ebp+dwBytes], ecx
add     ecx, [ebp+var_C]
add     ecx, [ebp+var_10]
add     eax, ecx
```

```

.rdata:00403198 aMozilla40Compa: ; DATA XREF: sub_401DCE+65↑o
.rdata:00403198 text "UTF-16LE", 'Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1
.rdata:00403198 text "UTF-16LE", 'InfoPath.1)',0
.rdata:00403216 align 4
.rdata:00403218 ; const WCHAR pszProxyW ; DATA XREF: sub_40204D+11↑o
.rdata:00403218 text "UTF-16LE", '10.15.1.69:3128',0
.rdata:00403238 ; const WCHAR pwszVerb
.rdata:00403238 pwszVerb: ; DATA XREF: sub_40204D+60↑o
.rdata:00403238 text "UTF-16LE", 'POST',0
.rdata:00403242 align 4
.rdata:00403244 ; const WCHAR pswzServerName
.rdata:00403244 pswzServerName: ; DATA XREF: sub_402174+E↑o
.rdata:00403244 text "UTF-16LE", '5.39.218.152',0
.rdata:0040325E align 10h
.rdata:00403260 ; const WCHAR pMore
.rdata:00403260 pMore: ; DATA XREF: sub_4023A8+30↑o
.rdata:00403260 text "UTF-16LE", '..',0
.rdata:00403266 align 4

```


#### 1.4.特殊工控流量

题目：某10段工控网络中，工业协议中存在异常数据。请通过流量中的数据找寻flag


题目附件连接：<https://>（提取码：）

解题步骤：

1. 打开流量包发现工控协议只有S7Comm[西门子通信协议S7Comm](#)，

 ICS-2019-1.pcap

文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T)



s7comm

No.	Ti	Source	Destination	Protocol	Leng	Info
247790	3...	10.10.10.20	10.10.10.10	S7COMM	153	ROSCTR:[Job ] Function:[Read Var]
247792	3...	10.10.10.10	10.10.10.20	S7COMM	104	ROSCTR:[Ack_Data] Function:[Read Var]
247810	3...	10.10.10.20	10.10.10.10	S7COMM	153	ROSCTR:[Job ] Function:[Read Var]
247812	3...	10.10.10.10	10.10.10.20	S7COMM	104	ROSCTR:[Ack_Data] Function:[Read Var]
247833	3...	10.10.10.20	10.10.10.10	S7COMM	153	ROSCTR:[Job ] Function:[Read Var]
247835	3...	10.10.10.10	10.10.10.20	S7COMM	104	ROSCTR:[Ack_Data] Function:[Read Var]
247856	3...	10.10.10.20	10.10.10.10	S7COMM	153	ROSCTR:[Job ] Function:[Read Var]
247858	3...	10.10.10.10	10.10.10.20	S7COMM	104	ROSCTR:[Ack_Data] Function:[Read Var]
247881	3...	10.10.10.20	10.10.10.10	S7COMM	153	ROSCTR:[Job ] Function:[Read Var]
247883	3...	10.10.10.10	10.10.10.20	S7COMM	104	ROSCTR:[Ack_Data] Function:[Read Var]
247907	3...	10.10.10.20	10.10.10.10	S7COMM	153	ROSCTR:[Job ] Function:[Read Var]
247909	3...	10.10.10.10	10.10.10.20	S7COMM	104	ROSCTR:[Ack_Data] Function:[Read Var]
247935	3...	10.10.10.20	10.10.10.10	S7COMM	153	ROSCTR:[Job ] Function:[Read Var]
247937	3...	10.10.10.10	10.10.10.20	S7COMM	104	ROSCTR:[Ack_Data] Function:[Read Var]
247963	3...	10.10.10.20	10.10.10.10	S7COMM	153	ROSCTR:[Job ] Function:[Read Var]
247965	3...	10.10.10.10	10.10.10.20	S7COMM	104	ROSCTR:[Ack_Data] Function:[Read Var]
247990	3...	10.10.10.20	10.10.10.10	S7COMM	153	ROSCTR:[Job ] Function:[Read Var]
247994	3...	10.10.10.10	10.10.10.20	S7COMM	104	ROSCTR:[Ack_Data] Function:[Read Var]
248016	3...	10.10.10.20	10.10.10.10	S7COMM	153	ROSCTR:[Job ] Function:[Read Var]
248018	3...	10.10.10.10	10.10.10.20	S7COMM	104	ROSCTR:[Ack_Data] Function:[Read Var]
248043	3...	10.10.10.20	10.10.10.10	S7COMM	153	ROSCTR:[Job ] Function:[Read Var]
248045	3...	10.10.10.10	10.10.10.20	S7COMM	104	ROSCTR:[Ack_Data] Function:[Read Var]
248065	3...	10.10.10.20	10.10.10.10	S7COMM	153	ROSCTR:[Job ] Function:[Read Var]
248067	3...	10.10.10.10	10.10.10.20	S7COMM	104	ROSCTR:[Ack_Data] Function:[Read Var]
248087	3...	10.10.10.20	10.10.10.10	S7COMM	153	ROSCTR:[Job ] Function:[Read Var]
248089	3...	10.10.10.10	10.10.10.20	S7COMM	104	ROSCTR:[Ack_Data] Function:[Read Var]

因为题目说工控协议存在异常数据，因此，重点分析S7Comm的流量。首先在wireshark的过滤器中输入S7Comm，过滤出S7Comm的数据，根据S7Comm的数据格式分析

	PDU类型	作用
ACK_DATA		确认数据响应，响应JOB的请求
JOB		作业请求，由主设备发送的请求（例如，读/写存储器，读/写块，启动/停止设备，设置通信参数）

PDU数据部分携带有功能码，通过脚本分析7Comm的数据使用了那种功能码，考虑不同的功能可能会产生异常的数据，代码如下。

```
import pyshark
def func_s7():
    try:
        captures = pyshark.FileCapture("ICS-2019-1.pcap")#■■■■■■■■■■
        func_codes = {}
        for c in captures:
            for pkt in c:
                if pkt.layer_name == "s7comm":
                    if hasattr(pkt, "param_func"):#param_func■■■■■■
                        func_code = pkt.param_func
                        if func_code in func_codes:
                            func_codes[func_code] += 1
                        else:
                            func_codes[func_code] = 1
                    print(func_codes)
    except Exception as e:
        print(e)
if __name__ == '__main__':
    func_s7()
```

执行后的如图所示，一共存在三种功能码：0x04(读取值 Read Var)出现172683次、0xf0(建立通信 Setup communication)出现32次、0x05(写入值 Write Var)出现96次。

```
Microsoft Windows [版本 10.0.17763.615]
(c) 2018 Microsoft Corporation。保留所有权利。

C:\Users\wang99bin\OneDrive\桌面\ICS-2019-1>python 1.py
{'0x00000004': 172683, '0x000000f0': 32, '0x00000005': 96}

C:\Users\wang99bin\OneDrive\桌面\ICS-2019-1>
```

因为异常数据很有可能被黑客写入设备，因此首先重点分析功能码为0x05的流量，在wireshark中过滤流量s7comm.param.func==0x05，其中PDU为Job的数据包是有可能存在黑客写入的数据请求，于是人工审计PDU为Job的数据包，于是发现编号为88607的数据包在数据部分存在一串可以的10



No.	Time	Source	Destination	Protocol	Length	Info
843584	53440.537073	10.10.10.10	10.10.10.10	STCOPM	76	ROSCTR:[Ack_Data] Function:[Write Var]
857390	54450.230817	10.10.10.10	10.10.10.10	STCOPM	94	ROSCTR:[Job ] Function:[Write Var]
857387	54450.252568	10.10.10.10	10.10.10.10	STCOPM	76	ROSCTR:[Ack_Data] Function:[Write Var]
857457	54453.969869	10.10.10.10	10.10.10.10	STCOPM	94	ROSCTR:[Job ] Function:[Write Var]
857458	54453.979363	10.10.10.10	10.10.10.10	STCOPM	76	ROSCTR:[Ack_Data] Function:[Write Var]
857555	54460.511814	10.10.10.10	10.10.10.10	STCOPM	94	ROSCTR:[Job ] Function:[Write Var]
857556	54460.521429	10.10.10.10	10.10.10.10	STCOPM	76	ROSCTR:[Ack_Data] Function:[Write Var]
857565	54460.717924	10.10.10.10	10.10.10.10	STCOPM	94	ROSCTR:[Job ] Function:[Write Var]
857568	54460.733923	10.10.10.10	10.10.10.10	STCOPM	76	ROSCTR:[Ack_Data] Function:[Write Var]
857915	54480.888551	192.168.1.10	10.10.10.10	STCOPM	192	ROSCTR:[Job ] Function:[Setup communication]   ROSCTR:[Job ] Function:[Read Var]   ROSCTR:[Job ] Function:[Write Var]
857922	54480.887460	10.10.10.10	192.168.1.10	STCOPM	76	ROSCTR:[Ack_Data] Function:[Write Var]
879758	55787.457115	10.10.10.10	10.10.10.10	STCOPM	94	ROSCTR:[Job ] Function:[Write Var]
879759	55787.473894	10.10.10.10	10.10.10.10	STCOPM	76	ROSCTR:[Ack_Data] Function:[Write Var]
880070	55799.452692	10.10.10.10	10.10.10.10	STCOPM	94	ROSCTR:[Job ] Function:[Write Var]
880070	55799.464343	10.10.10.10	10.10.10.10	STCOPM	76	ROSCTR:[Ack_Data] Function:[Write Var]
880332	55809.690878	10.10.10.10	10.10.10.10	STCOPM	94	ROSCTR:[Job ] Function:[Write Var]
880333	55809.695899	10.10.10.10	10.10.10.10	STCOPM	76	ROSCTR:[Ack_Data] Function:[Write Var]
880359	55811.007109	10.10.10.10	10.10.10.10	STCOPM	94	ROSCTR:[Job ] Function:[Write Var]
880360	55811.016473	10.10.10.10	10.10.10.10	STCOPM	76	ROSCTR:[Ack_Data] Function:[Write Var]
880625	55821.299585	10.10.10.10	10.10.10.10	STCOPM	94	ROSCTR:[Job ] Function:[Write Var]
880626	55821.300327	10.10.10.10	10.10.10.10	STCOPM	76	ROSCTR:[Ack_Data] Function:[Write Var]
880633	55821.405812	10.10.10.10	10.10.10.10	STCOPM	94	ROSCTR:[Job ] Function:[Write Var]
880634	55821.508382	10.10.10.10	10.10.10.10	STCOPM	76	ROSCTR:[Ack_Data] Function:[Write Var]
881547	55878.630357	192.168.1.10	10.10.10.10	STCOPM	192	ROSCTR:[Job ] Function:[Setup communication]   ROSCTR:[Job ] Function:[Read Var]   ROSCTR:[Job ] Function:[Write Var]
881554	55878.713633	10.10.10.10	192.168.1.10	STCOPM	76	ROSCTR:[Ack_Data] Function:[Write Var]
885955	56130.557913	10.10.10.10	10.10.10.10	STCOPM	94	ROSCTR:[Job ] Function:[Write Var]
885966	56130.567939	10.10.10.10	10.10.10.10	STCOPM	76	ROSCTR:[Ack_Data] Function:[Write Var]
886067	56130.665282	192.168.1.10	10.10.10.10	STCOPM	192	ROSCTR:[Job ] Function:[Setup communication]   ROSCTR:[Job ] Function:[Read Var]   ROSCTR:[Job ] Function:[Write Var][Malformed Packet]
886080	56130.567674	10.10.10.10	192.168.1.10	STCOPM	76	ROSCTR:[Ack_Data] Function:[Write Var]
962759	60430.370339	10.10.10.10	10.10.10.10	STCOPM	94	ROSCTR:[Job ] Function:[Write Var]
962759	60430.370101	10.10.10.10	10.10.10.10	STCOPM	76	ROSCTR:[Ack_Data] Function:[Write Var]
962821	60434.740110	10.10.10.10	10.10.10.10	STCOPM	94	ROSCTR:[Job ] Function:[Write Var]
962824	60434.751649	10.10.10.10	10.10.10.10	STCOPM	76	ROSCTR:[Ack_Data] Function:[Write Var]

## Wireshark · 分组 886067 · ICS-2019-1.pcap

- > Frame 886067: 192 bytes on wire (1536 bits), 192 bytes captured (1536 bits)
- > Ethernet II, Src: Ubiquiti\_83:db:16 (04:18:d6:83:db:16), Dst: Siemens\_89:59:82 (28:63:36:89:59:82)
- > Internet Protocol Version 4, Src: 192.168.1.10, Dst: 10.10.10.10
- > Transmission Control Protocol, Src Port: 49228, Dst Port: 102, Seq: 1, Ack: 1, Len: 138
- > TPKT, Version: 3, Length: 22
- > ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
- > TPKT, Version: 3, Length: 25
- > ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
- ▼ S7 Communication
  - > Header: (Job)
  - > Parameter: (Setup communication)
- > TPKT, Version: 3, Length: 51
- > ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
- ▼ S7 Communication
  - > Header: (Job)
  - > Parameter: (Read Var)
- > TPKT, Version: 3, Length: 40
- > ISO 8073/X.224 COTP Connection-Oriented Transport Protocol

```
0000 28 63 36 89 59 82 04 18 d6 83 db 16 08 00 45 00 (c6.Y... ..E.
0010 00 b2 19 7f 40 00 7f 06 0c 01 c0 a8 01 0a 0a 0a ...@... ..
0020 0a 0a c0 4c 00 66 35 6e 75 34 00 05 77 f4 50 18 ..L.f5n u4..w.P.
0030 5b 40 ee e9 00 00 03 00 00 16 11 e0 00 00 00 01 [e.....
0040 00 c1 02 02 00 c2 02 02 01 c0 01 09 03 00 00 19 .....
0050 02 f0 80 32 01 00 00 08 00 00 08 00 00 f0 00 00 ..2....
0060 01 00 02 00 f0 03 00 00 33 02 f0 80 32 01 00 00 ..... 3...2...
0070 01 ff 00 22 00 00 04 02 12 0e b2 ff 00 00 00 52 ..."....R
0080 82 55 51 e7 40 00 00 09 12 0e b2 ff 00 00 00 52 .UQ.@....R
0090 25 36 7d b1 40 00 00 0a 03 00 00 28 02 f0 80 32 %6}:@... (...2
00a0 01 00 00 02 09 00 12 00 05 05 36 39 37 33 35 66 .....69735f
00b0 36 65 36 66 37 34 35 66 37 32 36 35 36 31 36 63 6e6f745f 7265616c
```

# ASCII在线转换器-十六进制，十进制、二进制

ASCII转换到 ASCII (例: a b c)

is\_not\_real

添加空格

删除空格

☐ 将空白字符转换

十六进制转换到16进制(例:0x61或61或61/62) ☒ 删除 0x

69735f6e6f745f7265616c

## 1.5.简单流量分析

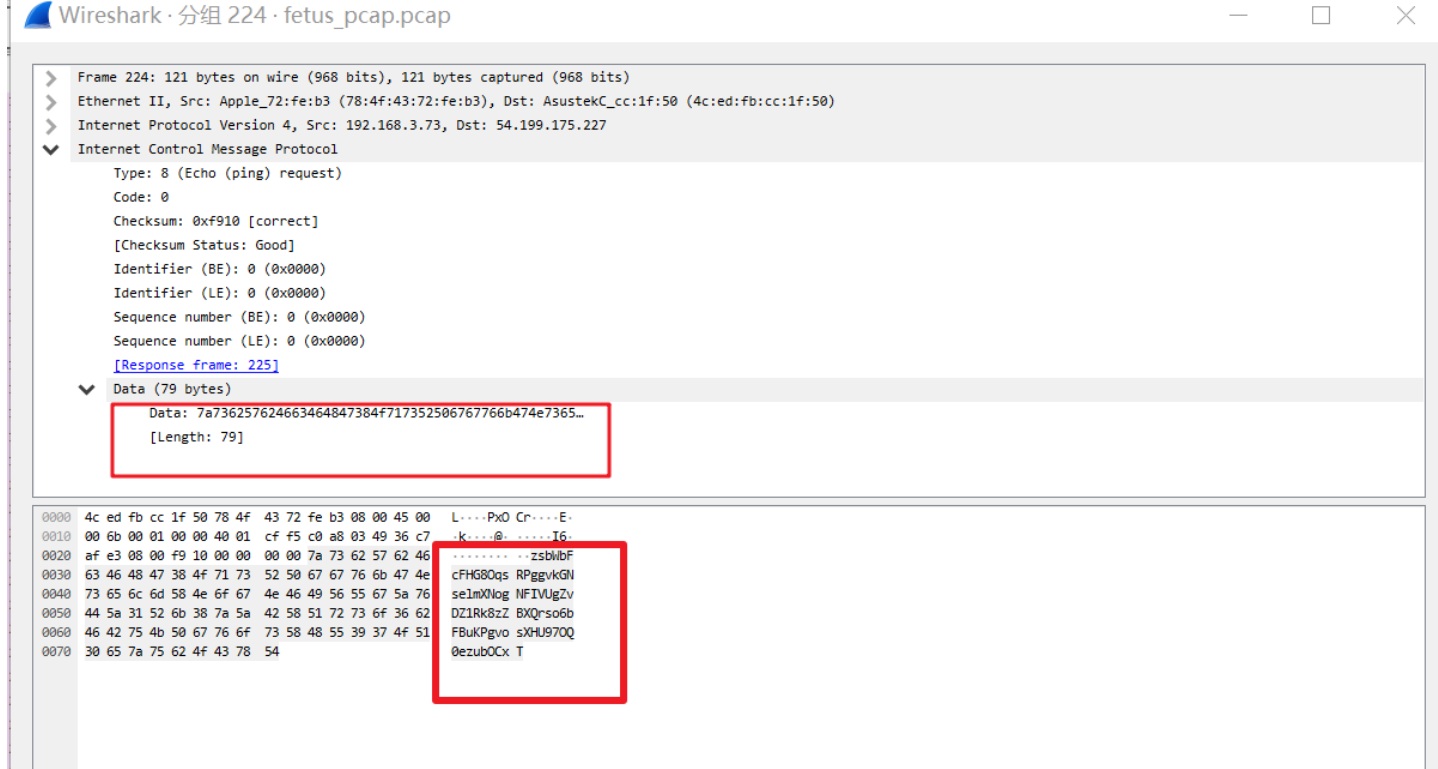
题目：不久前，运维人员在日常安全检查的时候发现现场某设备会不时向某不知名加发甜非正常的ICMP PING包。这引起了运维人员的注意，他在过滤出ICMP包分析并马上开始做应急处理很可能已被攻击的设备。运维人员到底发现了什么?flag形式为flag}  
题目附件连接：链接：[https://pan.baidu.com/s/1IbD\\_AWnXLWiUH-RqbQqq5q](https://pan.baidu.com/s/1IbD_AWnXLWiUH-RqbQqq5q) 提取码：j63x

打开数据包发现存在大量的ICMP的请求包和响应数据包。

fetus\_pcap.pcap

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.3.73	54.199.175.227	ICMP	121	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found)
2	0.615618	192.168.3.73	54.199.175.227	ICMP	148	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 3)
3	0.785965	54.199.175.227	192.168.3.73	ICMP	148	Echo (ping) reply id=0x0000, seq=0/0, ttl=33 (request in 2)
4	1.227968	192.168.3.73	54.199.175.227	ICMP	154	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found)
5	1.843180	192.168.3.73	54.199.175.227	ICMP	141	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found)
6	2.458320	192.168.3.73	54.199.175.227	ICMP	140	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 7)
7	2.628828	54.199.175.227	192.168.3.73	ICMP	140	Echo (ping) reply id=0x0000, seq=0/0, ttl=33 (request in 6)
8	3.072683	192.168.3.73	54.199.175.227	ICMP	151	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 9)
9	3.243498	54.199.175.227	192.168.3.73	ICMP	151	Echo (ping) reply id=0x0000, seq=0/0, ttl=33 (request in 8)
10	3.689121	192.168.3.73	54.199.175.227	ICMP	91	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 11)
11	3.860177	54.199.175.227	192.168.3.73	ICMP	91	Echo (ping) reply id=0x0000, seq=0/0, ttl=33 (request in 10)
12	4.382471	192.168.3.73	54.199.175.227	ICMP	160	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 13)
13	4.476824	54.199.175.227	192.168.3.73	ICMP	160	Echo (ping) reply id=0x0000, seq=0/0, ttl=33 (request in 12)
14	4.916816	192.168.3.73	54.199.175.227	ICMP	140	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 15)
15	5.087686	54.199.175.227	192.168.3.73	ICMP	140	Echo (ping) reply id=0x0000, seq=0/0, ttl=33 (request in 14)
16	5.534942	192.168.3.73	54.199.175.227	ICMP	151	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 17)
17	5.706391	54.199.175.227	192.168.3.73	ICMP	151	Echo (ping) reply id=0x0000, seq=0/0, ttl=33 (request in 16)
18	6.147555	192.168.3.73	54.199.175.227	ICMP	142	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 19)
19	6.318953	54.199.175.227	192.168.3.73	ICMP	142	Echo (ping) reply id=0x0000, seq=0/0, ttl=33 (request in 18)
20	6.762975	192.168.3.73	54.199.175.227	ICMP	160	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found)
21	7.378672	192.168.3.73	54.199.175.227	ICMP	132	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 22)
22	7.548568	54.199.175.227	192.168.3.73	ICMP	132	Echo (ping) reply id=0x0000, seq=0/0, ttl=33 (request in 21)
23	7.992678	192.168.3.73	54.199.175.227	ICMP	113	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 24)
24	8.164439	54.199.175.227	192.168.3.73	ICMP	113	Echo (ping) reply id=0x0000, seq=0/0, ttl=33 (request in 23)
25	8.688683	192.168.3.73	54.199.175.227	ICMP	115	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response found)
26	9.224885	192.168.3.73	54.199.175.227	ICMP	96	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (reply in 27)
27	9.396437	54.199.175.227	192.168.3.73	ICMP	96	Echo (ping) reply id=0x0000, seq=0/0, ttl=33 (request in 26)

分析发现请求包和响应包的数据部分都存在着内容，内容是一串字符内容，尝试进行解密，但是失败。



在分析发现ICMP数据部分（data）的长度对应的ASCII码有可能是flag值，利用脚本将ICMP数据部分的长度提取出来，发现是一串base64编码的字符串，然后利用base

```
#!/usr/bin/python
# coding=utf8
import pyshark
import base64
L_flag= []
packets = pyshark.FileCapture('fetus_pcap.pcap')
for packet in packets:
    for pkt in packet:
        if pkt.layer_name == "icmp":
            if int(pkt.type) != 0:
                L_flag.append(int(pkt.data_len))
c=len(L_flag)
for i in range(0,c):
    L_flag[i]=chr(L_flag[i])
print(''.join(L_flag))
print(base64.b64decode(''.join(L_flag)))
```

```
C:\Users\wang99bin\OneDrive\桌面\8-23\2019_工业信息安全技能大赛个人线上赛(第二场)\2.8简单流量分析\question_1565019978_2-8>python l.py
0jpcbm1vbmdvZGI6IToxNzg0Mzow0jk50Tk50jc60jpcbnVidW50dTokNiRMaEhSb21URSRNNOMQb jg0VWNGTEFHe3h4MmI4YV82bW02NGNfZnNvY21
b'::\nmongodb!:17843:0:99999:7:::\nubuntu:$6$LhHRomTE$M7C4n84UcFLAG{xx2b8a_6mm64c_fsociety}::'
```

1. 这道题真的是有一点脑洞，分析尝试了很多方法，没想到最后在ICMP数据长度上做了文章。

# 参考连接

- 2. [西门子通信协议S7Comm](#)
- 3. [Rabin加密](#)
- 4. [计网实验笔记（一）](#)

点击收藏 | 0 关注 | 1

[上一篇：一次CSRF测试引发的思考](#) [下一篇：Awesome-WAF readm...](#)

1. 0 条回复

- 动动手指，沙发就是你的了！

[登录](#) 后跟帖

先知社区

[现在登录](#)



[技术文章](#)

[社区小黑板](#)

[目录](#)

[RSS](#) [关于社区](#) [友情链接](#) [社区小黑板](#)