

# 2022第二届网刃杯网络安全大赛部分wp

原创

[3tefanie、zhou](#) 已于 2022-04-26 09:53:41 修改 1846 收藏 1

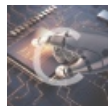
分类专栏: [CTF](#) 文章标签: [安全](#)

于 2022-04-25 16:33:39 首次发布

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/luochen2436/article/details/124405895>

版权



[CTF 专栏收录该内容](#)

18 篇文章 0 订阅

订阅专栏

## 文章目录

ics

easyiec

ncsubj

xyp07

carefulguy

喜欢移动的黑客

LED\_BOOM

re

freestyle

## ics

### easyiec

题目描述: 小Q刚刚入职了电力部门, 刁钻的主管让他学习第一堂课工控ctf, 但是小Q从来没有接触过ctf, 你能帮助他吗?

打开数据包，追踪tcp流，在流1发现flag

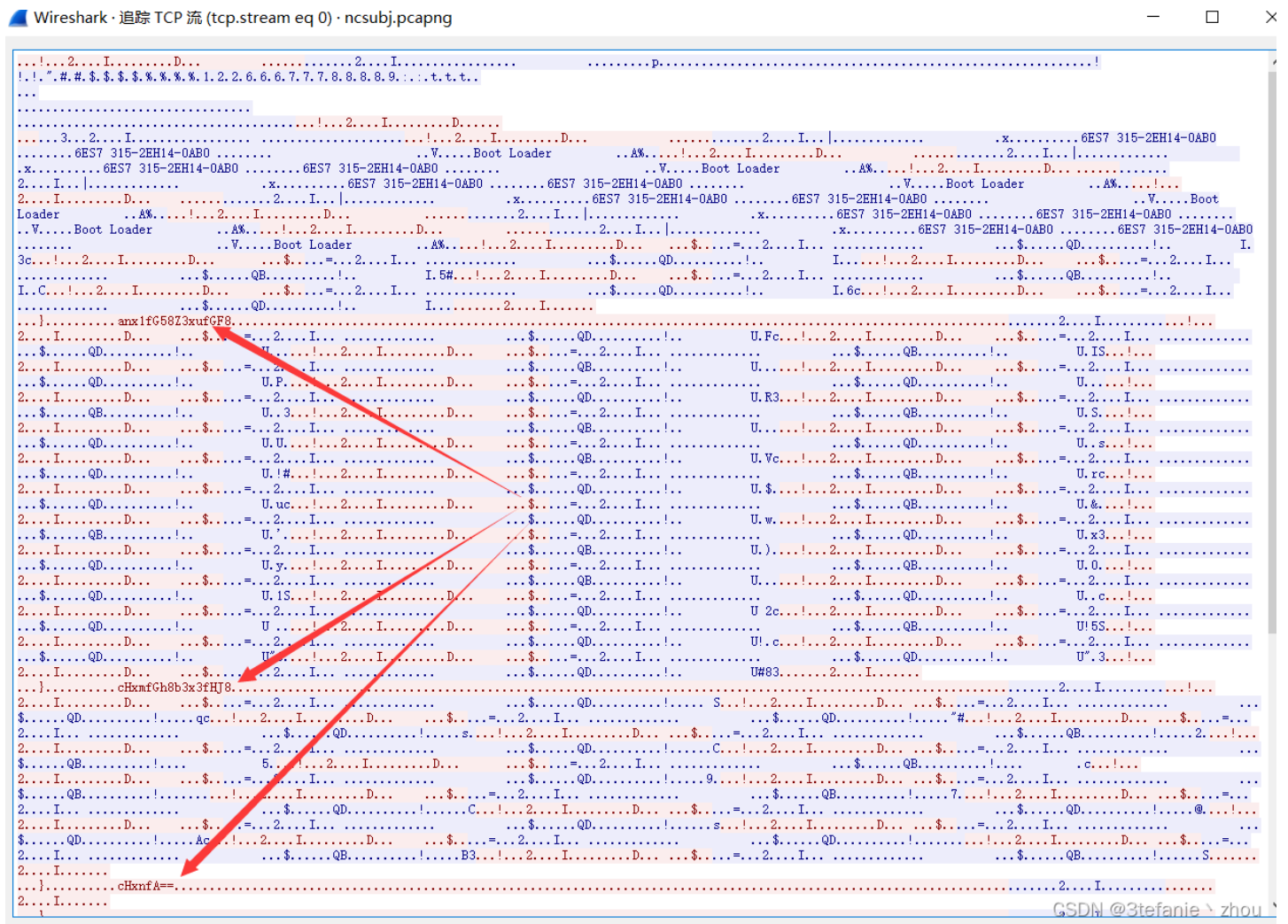


flag{e45y\_1eci04}

## ncsubj

题目描述: wowowow, 某厂商上位机TIA PORTIAL软件受到了hacker勒索软件的加密攻击, 不过好在我们的监测系统捕获了攻击者非法操作的流量, 具体的解密需要你自己去慢慢发现哟, flag格式为flag{}

打开数据包，追踪tcp流，在流0发现三串异常数据，如下：



```
anx1fG58Z3xufGF8
cHxmfGh8b3x3fHJ8
cHxnFA==
#拼接起来
anx1fG58Z3xufGF8cHxmfGh8b3x3fHJ8cHxnFA==
```

进行Base64解码

```
anx1fG58Z3xufGF8cHxmfGh8b3x3fHJ8cHxnFA==
```

编码源格式:  文本  Hex 解码结果: 自动检测 中文编码: UTF-8 编码 解码

```
j|u|n|g|n|a|p|i|h|o|w|r|p|g|
```

CSDN @3tefanie \ zhou

jungnapfhowrpg

再进行凯撒枚举,仔细查看凯撒位移后的字符串

## AmanCTF - 凯撒(Caesar)加密/解密

在线凯撒(Caesar)加密/解密

jungnapfhowrpg

偏移量

加密

解密

枚举

ZKdwdqtvxemhtw  
yjcvcpeuwlgev  
xibubodtvckfdu  
whatancsubject  
vgzszmbtrtids  
ufyrylaqszhcar  
texqkzprygbzq  
sdwpwjyoqxfayp  
rcvovixnpwezxo  
qbunuhwmovdywn

CSDN @3tefanie \ zhou

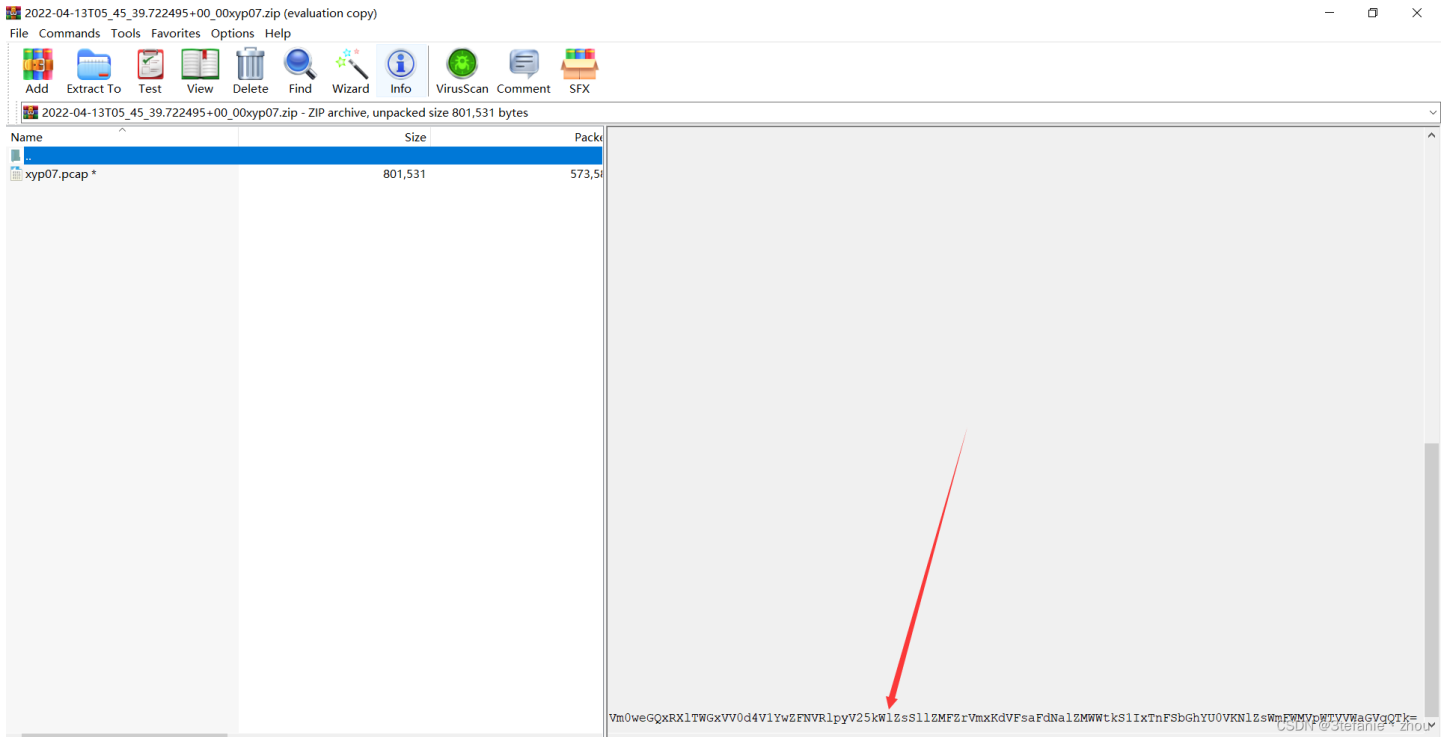
发现正常可通读字符串 `whatancsubject`

```
flag{whatancsubject}
```

### xyp07

题目描述: 电气公司的师傅九爷今日收了他的第七个徒弟, 取名做小七, 九爷生来喜欢7这个数字, 于是决定重点培养小七, 于是便给小七出了一道测试题, 初入行业的小七显得不知所措, 你能帮助他解决这个问题么?

打开压缩包，发现要密码，但是在备注发现一串base64字符串



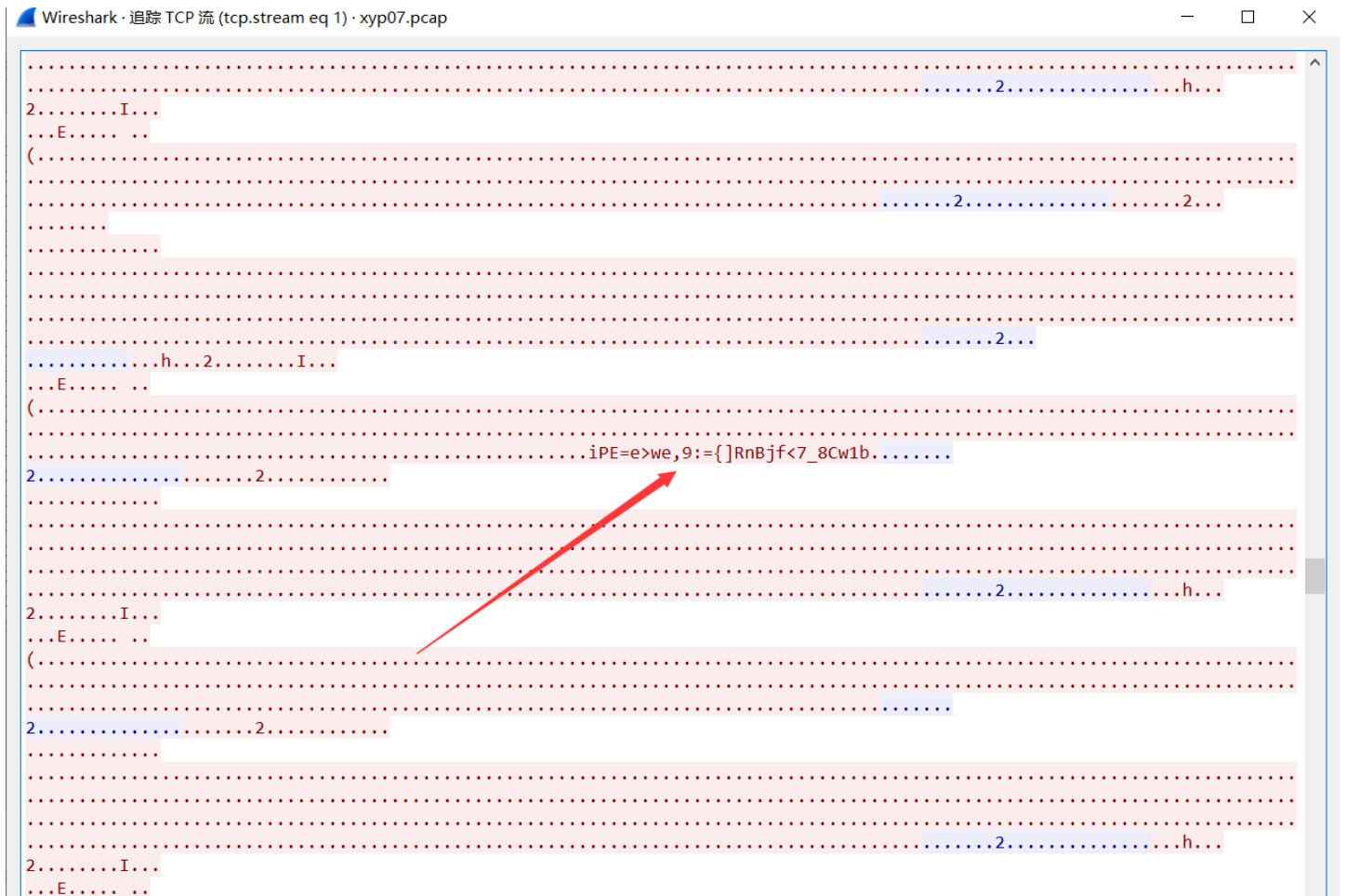
base64字

符 `Vm0weGQxRX1TWGxVV0d4V1YwZFNVRlpyV25kwlZs1lZMFZrVmxKdVFsafDNalZMwWtks1IxTnFsbGhYU0VKnlZswmFwMvpwTVVWaGVqTk=`

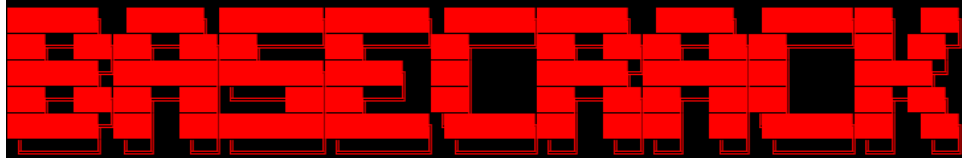
将字符串进行循环解码，得到 `Xyp77&77`

使用该字符串解压压缩包

打开数据包，追踪tcp流，在流1发现可疑字符



可疑字符: `iPE=e>we,9:={]RnBjf<7_8Cw1b`  
丢进basecrack工具中解密,发现是base91编码

```
E:\学习资料\CTF\sometools\basecrack-4.0>python basecrack.py -m  
 v4.0  
  
python basecrack.py -h [FOR HELP]  
[33m[>] Enter Encoded Base: [0miPE=e>we,9:={]RnBjf<7_8Cw1b  
[-] Iteration: 1  
[-] Heuristic Found Encoding To Be: Base91  
[-] Decoding as Base91: welcome_S7_world_xyp07  
{<<=====>>>}  
[-] Total Iterations: 1  
[-] Encoding Pattern: Base91  
[-] Magic Decode Finished With Result: welcome_S7_world_xyp07  
[-] Finished in 0.0125 seconds  
  
CSDN @3tefanie \ zhou
```

得到flag:

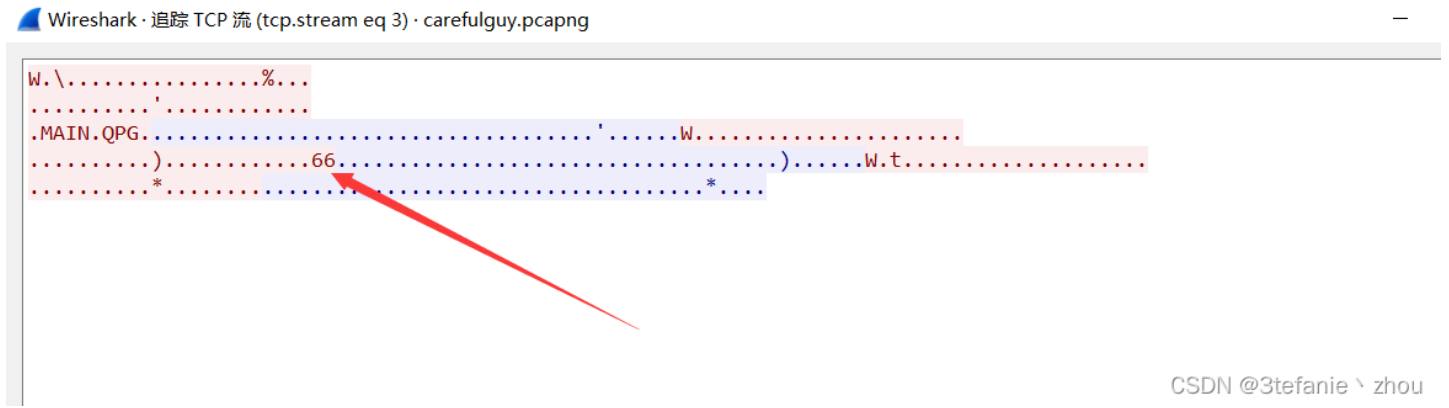
```
flag{welcome_S7_world_xyp07}
```

### carefulguy

题目描述: 电厂工程师Bob正在对将电磁阀的工程写入PLC, 传输时受到黑客攻击被迫停止, 重启后才恢复运作, 黑客的攻击导致工程师的数据丢失了一些, 实时监测设备抓到一些流量包, 你能从流量包中找出遗失的数据吗?

打开数据包，追踪tcp数据流

发现从流3开始，这个位置出现16进制数据



CSDN @3tefanie \ zhou

提取后面每一个流这个位置的16进制数据

PS: 注意剔除流24的 `__is_a_flag` 和流36的 `you_are_a_careful_guy`

得到 `666c61677b7034757333313576337279316e7433726573746963397d`

将上述16进制数据转换为字符串

### 16进制转换文本 / 文本转16进制

666c61677b7034757333313576337279316e7433726573746963397d	字符串转16进制 >>	flag{p4us315v3ry1nt3restic9}
	16进制转字符串 >>	
	结果互换	
	全部清空	

CSDN @3tefanie \ zhou

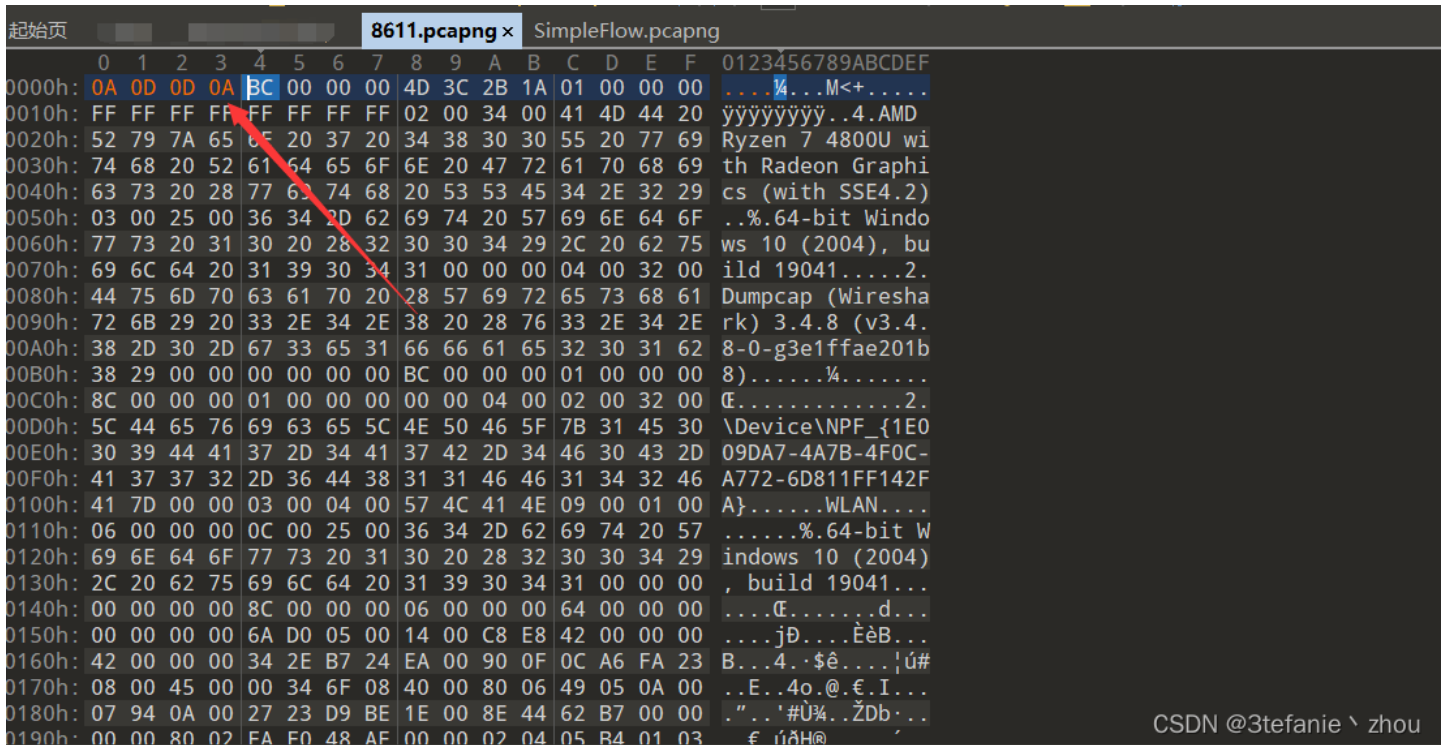
`flag{p4us315v3ry1nt3restic9}`

### 喜欢移动的黑客

题目描述: Monkey是一家汽修厂的老板, 日常喜欢改装车, 但由于发动机的转速有上限, 发动机最多能接受10000转/分钟的转速, Monkey在最新一次对发动机转速进行测试时发生了故障, 机械师阿张排查时测试期间, 有一些异常的流量, 请根据阿张捕获的流量包分析发动机的转速达到了多少转才出现的故障, flag为flag{data+包号}

打开数据包，发现报错

丢进010中一看，前四个字节不对，修改为 0A 0D 0D 0A

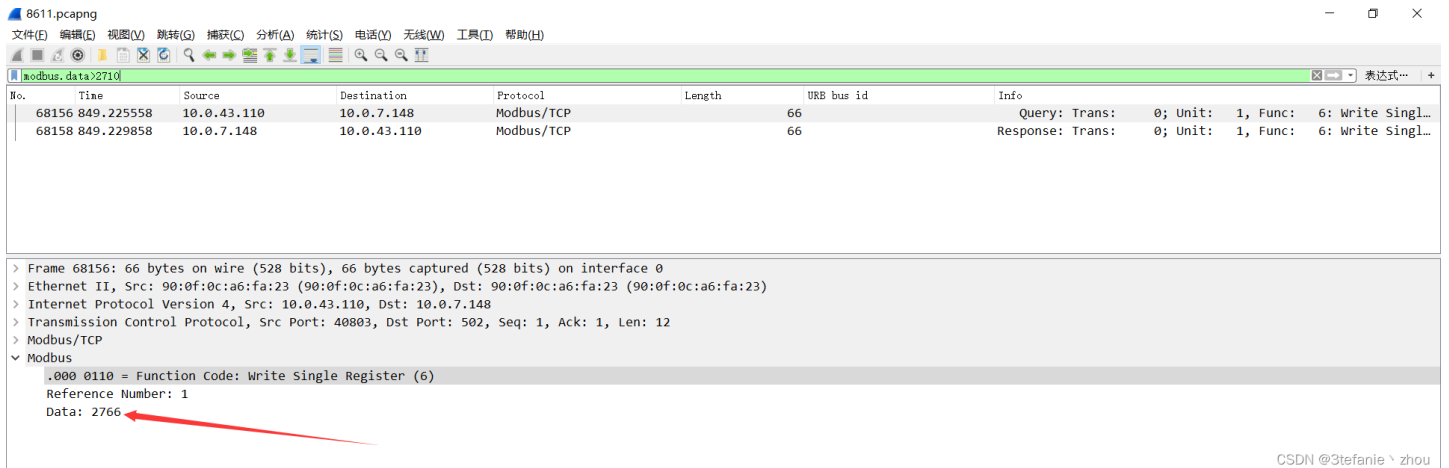


CSDN @3tefanie \ zhou

重新打开数据包

10000的16进制为2710

筛选数据包 modbus.data>2710



CSDN @3tefanie \ zhou

发现存在两个数据包的数据值为2766，10进制数为10086，包号分别为68156和68168

根据题目要求，flag为flag{data+包号}

可能情况如下：

- flag{276668156}
- flag{276668158}
- flag{2766+68156}
- flag{2766+68158}
- flag{1008668156}
- flag{1008668158}
- flag{10086+68156}
- flag{10086+68158}



经测试最终flag为:

flag{1008668156}

## LED\_BOOM

解压压缩吧，里面有一个张图片，内容为 U2FsdGVkX19c0OV8qLVgcso8U4fse+7LirQKiHFkn9HU9BuwFAivH1siJXg/Rk6z

看上去像是某种对称加密

还有一个 readme.txt:

攻击者DOM在打入核电站内网后，成功拿到一台上位机，并进行了非法操作，我们的监测组发现核电站内的LED灯间断的闪烁了三下，你跟根据监测组留下的线索，成功破案攻击者的行为吗，提交形式为flag{}

tips:听说闪烁三次的行为对应三个返回包，将他们的包号排列组合会得到你要的结果

打开数据包，过滤s7comm协议

根据tip为三个对应的返回包，在按长度将数据重新排序，仔细寻找发现仅有三个包长度一样

no.	Time	Source	Destination	Protocol	Length	URB bus	Info
755	328.282311	172.16.1.3	172.16.1.100	S7COMM	115	ROSCTR:[Userdata]	Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0424 Index=0x0000
758	328.791267	172.16.1.3	172.16.1.100	S7COMM	115	ROSCTR:[Userdata]	Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0424 Index=0x0000
762	329.291259	172.16.1.3	172.16.1.100	S7COMM	115	ROSCTR:[Userdata]	Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0424 Index=0x0000
766	329.805313	172.16.1.3	172.16.1.100	S7COMM	115	ROSCTR:[Userdata]	Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0424 Index=0x0000
773	330.315255	172.16.1.3	172.16.1.100	S7COMM	115	ROSCTR:[Userdata]	Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0424 Index=0x0000
776	330.821294	172.16.1.3	172.16.1.100	S7COMM	115	ROSCTR:[Userdata]	Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0424 Index=0x0000
798	340.089354	172.16.1.3	172.16.1.100	S7COMM	115	ROSCTR:[Userdata]	Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0424 Index=0x0000
801	340.602382	172.16.1.3	172.16.1.100	S7COMM	115	ROSCTR:[Userdata]	Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0424 Index=0x0000
805	341.118499	172.16.1.3	172.16.1.100	S7COMM	115	ROSCTR:[Userdata]	Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0424 Index=0x0000
809	341.632467	172.16.1.3	172.16.1.100	S7COMM	115	ROSCTR:[Userdata]	Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0424 Index=0x0000
813	342.137462	172.16.1.3	172.16.1.100	S7COMM	115	ROSCTR:[Userdata]	Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0424 Index=0x0000
817	342.648495	172.16.1.3	172.16.1.100	S7COMM	115	ROSCTR:[Userdata]	Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0424 Index=0x0000
820	343.147463	172.16.1.3	172.16.1.100	S7COMM	115	ROSCTR:[Userdata]	Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0424 Index=0x0000
824	343.659503	172.16.1.3	172.16.1.100	S7COMM	115	ROSCTR:[Userdata]	Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0424 Index=0x0000
827	344.165525	172.16.1.3	172.16.1.100	S7COMM	115	ROSCTR:[Userdata]	Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0424 Index=0x0000
585	294.197153	172.16.1.3	172.16.1.100	S7COMM	123	ROSCTR:[Userdata]	Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0074 Index=0x0000
692	316.876176	172.16.1.3	172.16.1.100	S7COMM	123	ROSCTR:[Userdata]	Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0074 Index=0x0000
787	333.192261	172.16.1.3	172.16.1.100	S7COMM	123	ROSCTR:[Userdata]	Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0074 Index=0x0000
473	261.169574	172.16.1.3	172.16.1.100	S7COMM	127	ROSCTR:[Userdata]	Function:[Response] -> [CPU functions] -> [Read SZL] SZL data:fragmentation \zhou

包号分别为 585、692、787

进行排列组合

585692787

585787692

692585787

692787585

787585692

787692585

使用上述数字作为Key去解密，当key为 585692787，对称加密方式为 AES 时得到flag（我才不会告诉你AES密文经常以 U2FsdGVkX19 为开头出现呢□）



在线加密解密

DES加密解密

3DES加密解密

AES加密解密

SHA1加密

MD5加密

HMAC计算

转换前:

U2FsdGVkX19cOOV8qLVgcs08U4fse+7LirQKiHFkn9HU9BuwFAivH1siJXg/Rk6z

密钥: 585692787

AES加密>

AES解密>

转换后:

flag{tietie\_tietie\_tiet13}

CSDN @3tefanie \ zhou

flag{tietie\_tietie\_tiet13}

re

freestyle

丢进ida中,发现主函数下有两个函数

fun1:

```
__int64 fun1()
{
    char s[24]; // [rsp+0h] [rbp-20h] BYREF
    unsigned __int64 v2; // [rsp+18h] [rbp-8h]

    v2 = __readfsqword(0x28u);
    puts("Welcome to Alaska!!!");
    puts("please input key: ");
    fgets(s, 20, stdin);
    if ( 4 * ( 3 * atoi(s) / 9 - 9 ) != 4400 )
        exit(0);
    puts("ok,level_1 over!\n\n");
    return 1LL;
}
```

fun2:

```

__int64 fun2()
{
    char s[24]; // [rsp+0h] [rbp-20h] BYREF
    unsigned __int64 v2; // [rsp+18h] [rbp-8h]

    v2 = __readfsqword(0x28u);
    puts("Welcome to Paradise Lost!!!");
    puts("The code value is the smallest divisible");
    puts("please input key: ");
    fgets(s, 20, stdin);
    if ( 2 * (atoi(s) % 56) != 98 )
        exit(0);
    puts("ok,level_2 over!");
    return 1LL;
}

```

两次运算分别求出值为3327和105

```

4 * (3 * atoi(s) / 9 - 9) = 4400
2 * (atoi(s) % 56) = 98

```

根据主函数中的提示，flag为md5格式

所以flag为flag{md5(3327105)},即：

```
flag{31a364d51abd0c8304106c16779d83b1}
```

【世间痴情男儿，不论地位高低，大抵都是喜欢女子便是错了，而且希望能一辈子知错不改】