CSACTF2025 Writeup

拿了一堆flag会先领盒饭吗
Grapesea, H2g, 1145141919810, EmotionalEDM

我们做出来的题目（除掉check in/out）是HiddenWorld, babyRev, babyLFSR, ezCrypto，排名13.

1 Misc部分

1.1 Check In-H2g

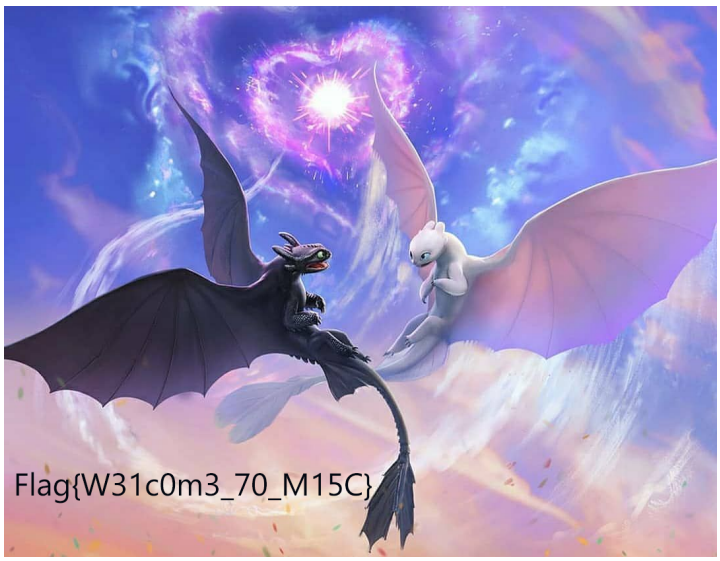
```
Q1NBQ1RGe0OxbmRzXzFudGVydHcxbmVkXyZfU2VjdXIxdHlfRDNzMWduM2R9
from Base64
CSACTF{M1nds_1ntertw1ned_&_Secur1ty_D3s1gn3d}
```

1.2 Hidden World-H2g

高度很明显不对劲，放到工具里修一下:

```
[*] Fix-PNG执行完毕，图片已经保存在文件所在的目录中或者同名目录中！

[-] Byxs20为您温馨提示：正在并行爆破图片正确的宽度和高度中...
[-] 宽度：1080，hex：0x438
[-] 高度：834，hex：0x342
[-] 运行时间为：0小时 0分钟 0秒 1毫秒
[-] CRC32：0x830E7CA6，已经为您保存到运行目录中！
```




```
Flag{W31c0m3_70_M15C}
->
CSACTF{W31c0m3_70_M15C}
```

flag格式没给对，这个不得不吐槽一下……

1.3 pyarmor-没做，但吐槽

整个队伍没人是MacOS环境，全都是Win，我觉得这题我们是没有可能动手去做的。（——Grapesea理直气壮地说）

1.4 鸿A口算-H2g做，Grapesea吐槽兼辅助

解压.hap文件，翻到了 flag.proto

```
message flag {
  string flag = 1; // 越问：FLAG的获取方式密文在末尾
}
//
51ba304139858edb60c616ad1ebe7cce750ea591f941a6f6d80780afbb19bb98ff0d5e05e5f5a235
//
3d6bd569f39e57e81ea00dbad8981096a9ddeca83de5f3b604692d9bb8236489863d452e6ad15453
//
d8d3a1ad62e3ac9caa27696718fb0c093afea7098a7ac985f1b06fae639d55d3cad902aa0e449d99
//
f4cd964cc65f7da7d35bc776c65e2fdd115dc6af6ad7dd8cf28e7ffd1e4d364cd5faaca258d77822
//
46688433a473ca1d815f249287eb3c3040c71380c8e30ab44bfda32ae1679202f8c7070ab072e179
//
ada5e70910012c8efd9dafe41cb1f614b37fa9806132987d4d13d32772aea3b559aa9e4cc16ac4a4
//
8370b4a4af15af6ad2bde38b8e83c98d7d7006ca6633d8e5806e44fb55b414eca7dde4a42dcbe109
//
3d2208e0d40764dfbdf2c488a729be33b38d38d8ee01b6beb8cb5b4e3ca6daca35a6af6ce887fb8f
//
2ba4001b39da181f0ebfb4b0ba6b4614c2512691b0fae79c119a6acfafa86f0fcb4292bc52bf6929
//
3d8a8464b9af71371ef4bfb425d5d5676b4b89e6f84dfec6d4440604e1e9213ba421d3cab0b4f730
//
eb86abc01a8a2935c25d3554ac960e3915335f713f3bd4e4b810a4057e785d331d9d6776d4607614
//
abf0571e41aeb66ec532bb04bd5e6f879ae7f2e77da60a7421133e6265665dd396bb0bd97312e18c
//
a436ba830e49eefa297ee6f45db3d1b99eb26e35ae06575b0ce09500f4144e28b90e2bf121b4e83e
//
918ccd02a9ce77905d91a9e2e3ba40a3ab38075d452eefb9c8bbe00ab2f7e0cab8978eda70ff81e2
```

目前没有任何头绪，只知道这是一个密文
在网上找到了对.hap中.abc文件的逆向工具abc-decompiler，在文件中翻到了如下内容

```
public Object func_main_0(Object functionObject, Object newTarget, a this) {
    newlexenv(3);
    _module_3_ = d1;
    _module_2_ = c1;
    _module_6_ = e5;
    _module_13_ = save;
    _module_8_ = hash;
    _module_0_ = a1;
    _module_12_ = load;
    _module_5_ = decrypt;
    _module_9_ = i10;
    _module_7_ = encrypt;
    _module_1_ = b1;
    _module_11_ = k10;
    _lexenv_0_1_ = l10;
    _module_10_ = j10;
    _module_4_ = "八十忒: 神奇的IV: 49ab3c206100bfad54d6be953955fbb4 加密方式: SM4/CBC/PKCS7 所有内容都需 bytes.fromhex";
    _lexenv_0_2_ = _module_2_(Uint8Array(createarraywithbuffer([44, 29, 68, 36, 216, 214, 220, 180, 3, 146, 131, 254, 68, 220, 38, 219, 9, 141, 109, 3, 56, 169,
    _lexenv_0_0_ = null;
    return null;
}
```

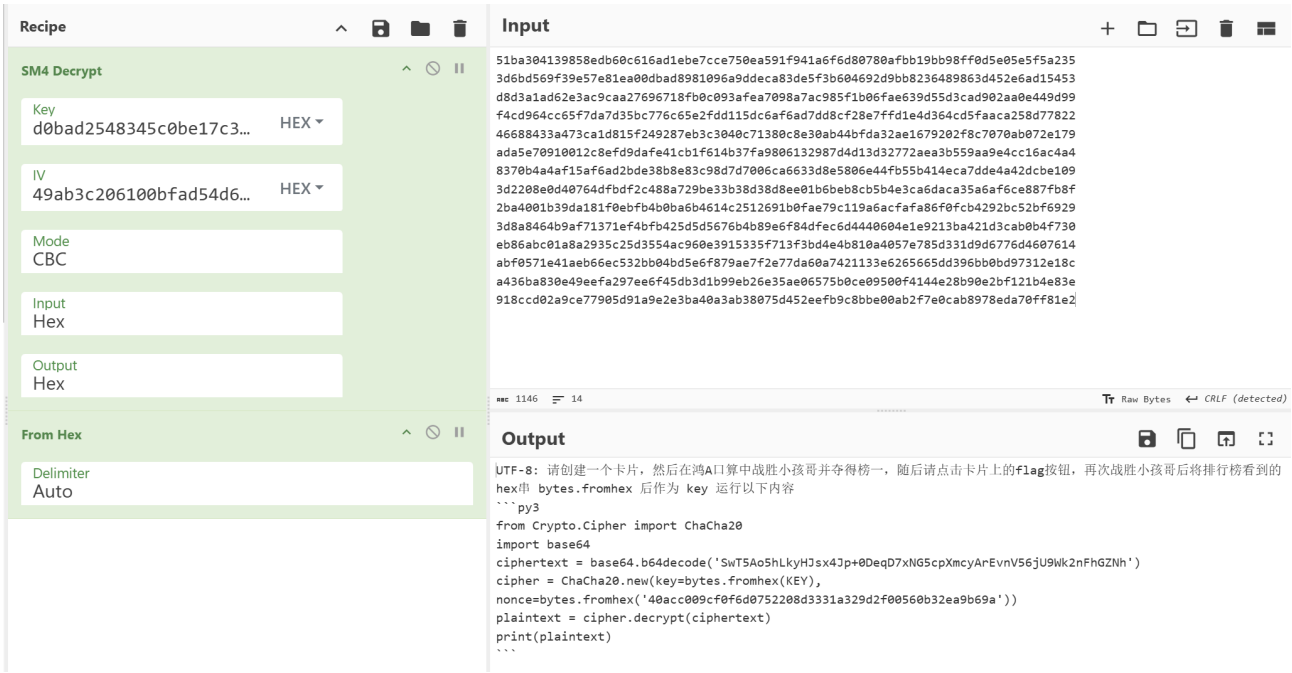
```
八十忒: 神奇的IV: 49ab3c206100bfad54d6be953955fbb4 加密方式: SM4/CBC/PKCS7 所有内容都需 bytes.fromhex
```

加密方式确定下来思路明朗多了，还给到了偏移量IV，现在只缺一个key，直接在文件夹遍历搜索一下16进制字符串

```
find hongA -type f -exec sh -c '
  for file; do
    echo "=== $file ==="
    strings "$file" | grep -E "^[0-9a-fA-F]{16,}$" | while read -r line; do
      echo "$line"
    done
  done
' sh {} +
=== hongA/module.json ===
=== hongA/pkgContextInfo.json ===
=== hongA/ets/modules.abc ===
=== hongA/ets/widget/component_collection.json ===
=== hongA/ets/widgets.abc ===
=== hongA/pack.info ===
=== hongA/resources/rawfile/flag.proto ===
=== hongA/resources/base/media/app_icon.png ===
=== hongA/resources/base/media/bottom_icon.png ===
=== hongA/resources/base/media/startIcon.png ===
=== hongA/resources/base/profile/main_pages.json ===
=== hongA/resources/base/profile/form_config.json ===
=== hongA/resources/base/profile/backup_config.json ===
=== hongA/resources.index ===
=== hongA/libs/x86_64/libentry.so ===
d0bad2548345c0be17c3f235014f69d0
=== hongA/libs/x86_64/libc++_shared.so ===
000102030405060708091011121314151617181920212223242526272829303132333435363738394041424344454647484950515253545556575859606162636465666768697071727374757677787980818283848586878889909192939495969798 99
=== hongA/libs/arm64-v8a/libentry.so ===
d0bad2548345c0be17c3f235014f69d0
=== hongA/libs/arm64-v8a/libc++_shared.so ===
```

竟然藏到依赖库里了？？

```
d0bad2548345c0be17c3f235014f69d0
```




```
UTF-8: 请创建一个卡片，然后在鸿A口算中战胜小孩哥并夺得榜一，随后请点击卡片上的flag按钮，再次战胜小孩哥后将排行榜看到的hex串 bytes.fromhex 后作为 key 运行以下内容
```py3
from Crypto.Cipher import ChaCha20
import base64
ciphertext = base64.b64decode('SwT5Ao5hLkyHJsx4Jp+0DeqD7xNG5cpXmcyArEvnV56jU9Wk2nFhGZNh')
cipher = ChaCha20.new(key=bytes.fromhex(KEY),
nonce=bytes.fromhex('40acc009cf0f6d0752208d3331a329d2f00560b32ea9b69a'))
plaintext = cipher.decrypt(ciphertext)
print(plaintext)

```
```

上面这个需要破解，目前H2g和Grapesea正在收拾剩下的内容~~但是暂时还没找到解决小孩哥的办法~~.

1.5 babySteg-EmotionalEDM, H2g, Grapesea

H2g:
文件尾有假flag?
Q1NBQ1RGezFUX1NFZU1zX0wxa0VfNF9mTDQ5fQ==
CSACTF{1T_SEeMs_L1kE_4_fL49}
看bit plane很明显是有东西藏着的，但是试了好久都没搞出来，哎呀

Grapesea & EmotionalEDM：折腾了各种短学期课上的方法但都失败了.

2 Rev部分

2.1 babyRev-1145141919810

打开IDA， 发现是TEA加密（经典数字0x61C88647）
密钥是this_isnot_a_key
加密后的内容可知是 BC8BC8E3 9069C8DB 0731D6E5 913AEACB
两个一组反解可得：5F773057 4E6B5F75 745F7730 21214133

转写可得flag: CSACTF{W0w_u_kN0w_t3A!!}

3 Crypto部分

3.1 ezCrypto-Grapesea

首先，这道题前几天我做过类似的：CryptoHack – Modular Arithmetic - Modular Binomials

化简：

$$\begin{cases} c_1 \equiv (7p+2q)^{e_1} \pmod N \\ c_2 \equiv (5p+3q)^{e_2} \pmod N \end{cases} \Longrightarrow \begin{cases} c_1 \equiv (7p)^{e_1} + (2q)^{e_1} \pmod N \\ c_2 \equiv (5p)^{e_2} + (3q)^{e_2} \pmod N \end{cases} \Longrightarrow \begin{cases} c_1^{e_2} 5^{e_1 e_2} \equiv (35p)^{e_1 e_2} + (10q)^{e_1 e_2} \pmod N \\ c_2^{e_1} 7^{e_1 e_2} \equiv (35p)^{e_1 e_2} + (21q)^{e_1 e_2} \pmod N \end{cases}$$

$$\Longrightarrow d = (21^{e_1 e_2} - 10^{e_1 e_2}) q^{e_1 e_2} \equiv 7^{e_1 e_2} c_2^{e_1} - 5^{e_1 e_2} c_1^{e_2} \pmod N$$

所以只需要计算$gcd(d, N)$即可得到q.

payload:

```
import hashlib
import itertools
import string
import re
import gmpy2
import math
from pwn import *
import numpy as np
from math import isqrt, gcd
from fractions import Fraction
from Crypto.Util.number import long_to_bytes, bytes_to_long
import sympy
from Crypto.Cipher import AES
from Crypto.Util.number import *

e1 =
99930783396499186333945514944320999874896369704264596095118574673415475865875253925541823507656422962195260208567108567460339366055010226246886903006884396
e2 =
98622315518844682053154886464742973065886401594817170354013568609719817144769714784885733470813204830330557864931801655174879435487998481554097015946952415
c  =
55901802262435214617666003592762988475817677710371853541720344698764787306707874639847782579629332096055942013026505176416985974919479004417458085456394043275360903171536103053583251790628518753952543060780848158606515610299846413376493573042907292759220155160589910184410059973310831818961560671734593013630
c1 =
40229355340269257739770896034955847330470882389629386052431564363238466271625576412466245834179497302499769847960940723672643383320214407820912062076710617921419054040241577412851511347542787094945856317937066896047259364036219750643961105485688761317530941590187661063644339339132904752238067549935799782571
c2 =
10865286970610629187138671259183246680598823537623502887154947899940871535858114805018299875584537369303954726255865785334620246767284496710355494394993665390476680801045987641377126381156219841282390565275892959334669067156557651394987854857201385209978053827837411358728650417700354947122173454209444556612 0
N  =
12383915352096861793857783312908868592880949812793514676207991312547880485569134528693988047507630374796477475220483404233690686053576790982736049422533708962763286809548326474878002962937211851676507300686647203175308551499584173509628516272825139274248486233323534228372889134284972849066226110603762507837 1
d0 = pow(7,e1*e2,N)*pow(c2,e1,N) - pow(5,e1*e2,N)*pow(c1,e2,N)
```

```
24
25  p = gcd(d0,N)
26  q = N//p
27  print(f"{p},{q}")
28  e = 65537
29  assert gmpy2.is_prime(p)
30  assert gmpy2.is_prime(q)
31  assert p * q == N
32
33  phi = (p-1)*(q-1)
34
35  d = gmpy2.invert(e,phi)
36  m = pow(c,d,N)
37  print(m)
38  flag = long_to_bytes(m)
39  print(flag)
40  # 然后丢到cyberchef里面！！！犯傻了错失三血
```

flag: CSACTF{Go0d_JoB!_We1C0mE_t0_oUR_csACtF_2o25!}

3.2 babyLFSR-Grapesea

本来只是想习惯性地打开CTF Wiki学一下这个LFSR（实则是NFSR，前几天看到这个知识点但来不及学了）结果谁知道writeup是现成的（摊手）：

非线性反馈移位寄存器 - CTF Wiki

【复现】强网杯-StreamGame3-Writeup – Rhy7hm

与2018强网杯的题目甚至是一样的，就不重复写了.