### WEB

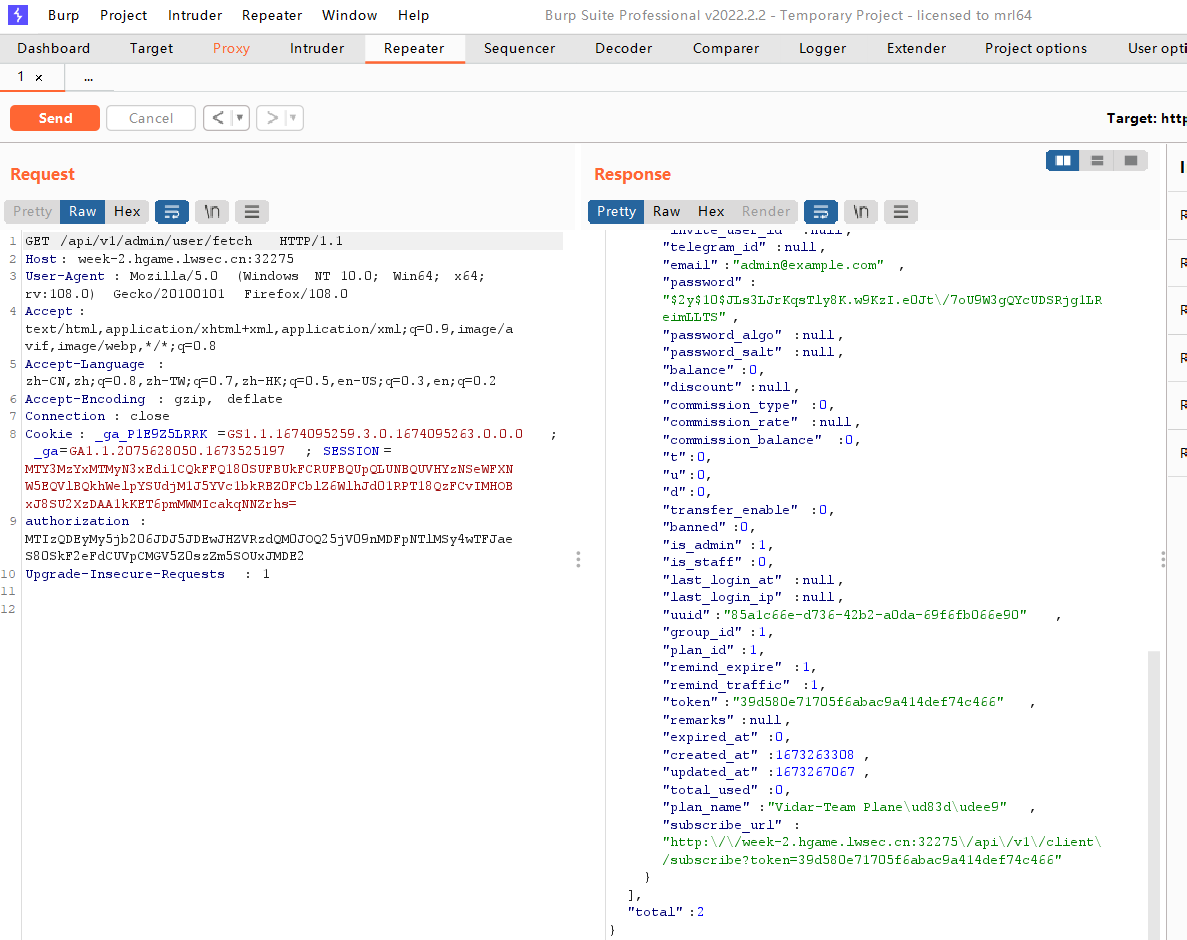
#### Git Leakage

/.git泄露，直接githack恢复flag文件即可。

Flag：{Don't^put\*Git-in\_web\_directory}

#### v2board

一个v2board的越权访问漏洞，当注册任意一个用户进行登陆后获取到 auth\_data 就可以任意调用管理员的接口：

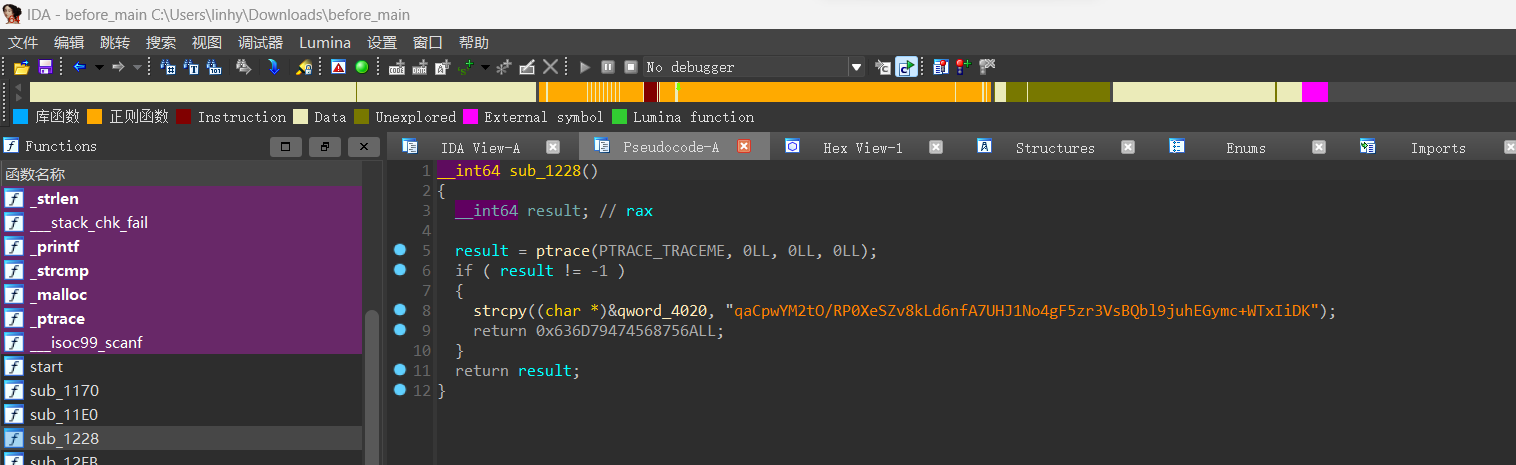


Flag：hagme{39d580e71705f6abac9a414def74c466}

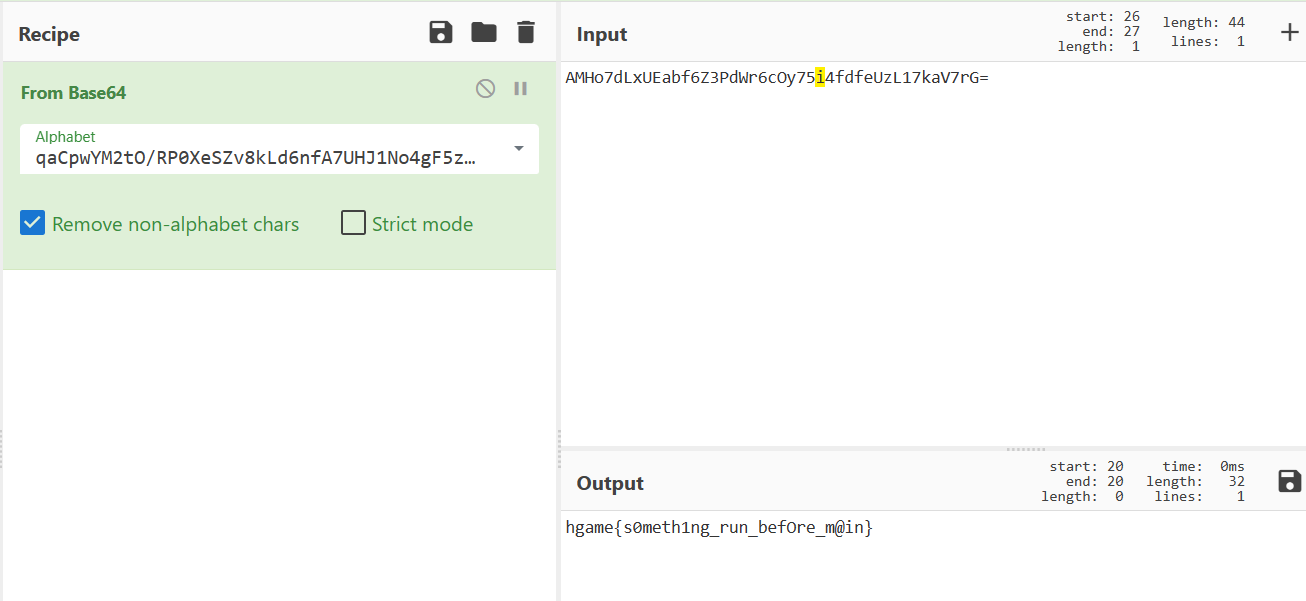
### REVERSE

#### before\_main

主函数逻辑就是进行base64加密，但是在这之前有一次换表操作：



因此用新表解base64即可：



#### Stream

Python封装的exe，使用pydumpck反编译，用struct的pyc头补充stream.pyc的头，然后反编译即可得到py源码。有了源码就写脚本逆向就行了。Exp：

|  |
| --- |
| 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 decrypt(enc, key):  enc1 = base64.b64decode(enc.encode()).decode()  flag = ''  for c, k in zip(enc1, gen(key)):  flag += chr(ord(c) ^ k)  return flag  enc = 'wr3ClVcSw7nCmMOcHcKgacOtMkvDjxZ6asKWw4nChMK8IsK7KMOOasOrdgbDlx3DqcKqwr0hw701Ly57w63CtcOl'  key = 'As\_we\_do\_as\_you\_know'  print(decrypt(enc, key)) |

Flag：hgame{python\_reverse\_is\_easy\_with\_internet}

#### VidarCamera

模拟器运行发现要输入序列号，输入错误的话会提示序列号不正确，因此我们使用jadx搜索序列号不正确，找到代码核心部分。分析代码发现是个魔改xtea，直接上exp：

|  |
| --- |
| #include<stdio.h>  void decrypt(unsigned long \*v, unsigned long \*k)  {  unsigned long v0=v[0], v1=v[1], i;  unsigned long delta=878077251;  unsigned long sum = delta\*33;  unsigned long k0=k[0], k1=k[1], k2=k[2], k3=k[3];  for(i=0; i<33; i++){  sum -= delta;  v1 -= (((v0 << 4) ^ (v0 >> 5)) + v0) ^ (sum + k[(sum>>11) & 3]);  v0 -= (((v1 << 4) ^ (v1 >> 5)) + v1) ^ (sum + k[sum & 3]) ^ sum;  }  v[0]=v0;  v[1]=v1;  }  int main()  {  unsigned long key[4] = {2233,4455,6677,8899};  unsigned long enc[11] = {0x260202FA,0x1B451064,0x867B61F1,0x228033C5,0xF15D82DC,0X9D8430B1,0x19F2B1E7,0x2BBA859C,0x2A08291D,0xDC707918,0};  //unsigned long enc[11] = {0xFA020226,0x6410451B,0xF1617B86,0xC5338022,0xDC825DF1,0XB130849D,0xE7B1F219,0x9C85BA2B,0x1D29082A,0x187970DC,0};  for(int i=9; i>0; i--){  unsigned long temp[2] = {enc[i-1], enc[i]};  decrypt(temp, key);  enc[i-1] = temp[0];  enc[i] = temp[1];  }  printf("%s\n", (char\*)enc);  return 0;  } |

Flag：hgame{d8c1d7d34573434ea8dfe5db40fbb25c0}

#### Math

核心逻辑是解五元一次方程组，exp：

|  |
| --- |
| import numpy as np  from scipy.linalg import solve  sum = [[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]]  for i in sum:  a=np.array([[126,253,62,118,59],[225,20,23,21,31],[62,124,100,184,186],[40,232,161,26,82],[216,122,36,142,79]])  b=np.array(i)  x=solve(a,b)  print(x) |

Flag：hgame{y0ur\_m@th\_1s\_gO0d}

### CRYPTO

#### 零元购年货商店

附件是go语言，环境是其web环境。整个逻辑是我们用一个用户名登录，接下来会根据用户名生成一个json，再用AES-CBC模式加密这个json，最后将生成的密文作为token显示在cookie中。

获取flag的条件是需要用户名为Vidar-Tu，但是我们不能使用这个用户名登录。经典的CBC翻转字节攻击，我们首先以Adiar-Tu作为用户名登录，这样我们要翻转的字节为第0组第9位，这样不需要iv也可以直接攻击。

Exp：

|  |
| --- |
| import base64  import urllib.parse  ciphertext = 'prvtT5Wh7UAzNgSZRbnVjpNcoRxHarhW0kI4fGupF3RDE2sxbRN6YRx2KMW2LvuGdKg6aPUgvfOqSQ%3D%3D'  cipher = base64.b64decode(urllib.parse.unquote(ciphertext))  array\_cipher = bytearray(cipher)  array\_cipher[9] = array\_cipher[9]^ ord('A') ^ ord('V')  #print(array\_cipher)  print('newCipher:',urllib.parse.quote(base64.b64encode(array\_cipher)))  #newCipher: prvtT5Wh7UAzIQSZRbnVjpNcoRxHarhW0kI4fGupF3RDE2sxbRN6YRx2KMW2LvuGdKg6aPUgvfOqSQ%3D%3D |

利用新token购买flag即可：



#### 包里有什么

经典背包加密，私钥是等比数列，根据乘数w的生成条件发现w只可能有两种情况，因此都进行尝试，发现正确的w为(m+b0)//2，有m有w有c有私钥，可以进行攻击。

Exp：

|  |
| --- |
| from Crypto.Util.number import long\_to\_bytes, inverse  c = 93602062133487361151420753057739397161734651609786598765462162  l = 406  b0 = 69356606533325456520968776034730214585110536932989313137926  m = 1528637222531038332958694965114330415773896571891017629493424  w = (b0+m) // 2  a = [2 << i for i in range(l)]  a = a[::-1]  x = inverse(w,m)\*c%m  flag = ''  for j in a:  if x >= j:  flag += '1'  x -= j  else:  flag += '0'  flag = flag[::-1]  print(long\_to\_bytes(int(flag, 2))) |

Flag：hgame{1t's\_4n\_3asy\_ba9\_isn7\_it?}

#### Rabin

如题所示，exp：

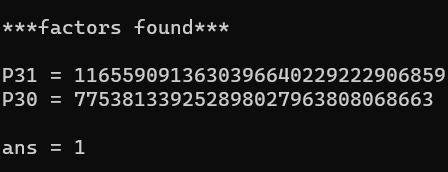
|  |
| --- |
| import gmpy2  from Crypto.Util.number import long\_to\_bytes  p=65428327184555679690730137432886407240184329534772421373193521144693375074983  q=98570810268705084987524975482323456006480531917292601799256241458681800554123  c=0x4e072f435cbffbd3520a283b3944ac988b98fb19e723d1bd02ad7e58d9f01b26d622edea5ee538b2f603d5bf785b0427de27ad5c76c656dbd9435d3a4a7cf556  e = 2  n = p\*q  c1=pow(c,(p+1)//4,p)  c2=pow(c,(q+1)//4,q)  cp1=p-c1  cp2=q-c2  t1=gmpy2.invert(p,q)#p的模q逆元  t2=gmpy2.invert(q,p)#q的模p逆元  m1=(q\*c1\*t2+p\*c2\*t1)%n  m2=(q\*c1\*t2+p\*cp2\*t1)%n # or m2=n-m1  m3=(q\*cp1\*t2+p\*c2\*t1)%n  m4=(q\*cp1\*t2+p\*cp2\*t1)%n # or m4=n-m3  for i in(m1,m2,m3,m4):  m = '%x' % i  if len(m)%2==1:  m='0'+m #padding  print(m) |

Flag：hgame{That'5\_s0\_3asy\_to\_s@lve\_r@bin}

#### RSA 大冒险1

总共有4个challenge，一个一个分析。

challenge1发现私钥的长度比较小，可以用yafu直接分解：



Exp：

|  |
| --- |
| p = 308575215160910675269385741515322500447  q = 1165590913630396640229222906859  r = 775381339252898027963808068663  e = 65537  c = 0x3c514248ea6dd5208ba15375c4503a58f42764c4733ce82d92a891f69ce719a28cb8d705b71bd8bca9  n = p\*q\*r  phi = (p-1)\*(q-1)\*(r-1)  d = inverse(e,phi)  m = pow(c,d,n)  print(long\_to\_bytes(m))  #m<n\_But\_also\_m<p |

Challenge2发现在进行加密获取密文后会再次获取一个新的q值，因此我们只需要获取两次n，取得这两个n的公因数就可以得到p的值了。Exp：

|  |
| --- |
| from Crypto.Util.number import long\_to\_bytes, inverse  from gmpy2 import gcd  n1 = 115363080541148350035098955843810429929506068421539791799116443117607025200171570002322430352811724439995076975845265556581001937873583129646088977535349785232415206002096113825882800512454245318318126768704044254622555288535336373259446030418177585994644465643757996936185469965231975158006935918851504230711  n2 = 143779732164453323314656410367424092526552595863885033629312465618648914977435952403865464612018340886266037825818397383716592907220557798718912329224966521838444233055295882127777276174644880688348211721988266576305489614060376782028834088204314619114270750478601562873936798061060399022164414111190394421639  p = gcd(n1,n2)  q = n1 // p  e = 65537  c = 0x3a3e7e594c06f48e7c611f13dbea7e30e8700580940e5c0fe0cad4fbb36273b8ee32e4f310d71f3d04a1e9221eee12f01a4d40304fb1756380e20a5fe0eea66dc289e6411dfac9babfb828d6d8a3f2b3a5379ff4e0b5954669a5b3c3957bf25ef25fbc36d37a646ed06d8289b386b88782db384d6b5f634e49ec7937dc96a0bd  phi = (p-1)\*(q-1)  d = inverse(e,phi)  m = pow(c,d,n1)  print(long\_to\_bytes(m))  #make\_all\_modulus\_independent |

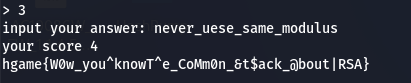
Challenge3发现e的值很小，为3，可以爆破：

|  |
| --- |
| from gmpy2 import iroot  import libnum  e = 0x3  n = 141899416715579636826739618482046641367949267357348224942546938114461299596413449379846657460912496166177317144210273416543884093721913751761601688993223059907319998983462097171187377562315371215054098779377781643034773833060838209136170225050881030702351871575978561533572553723754466974694832426033526204423  c = 0xfec61958cefda3eb5f709faa0282bffaded0a323fe1ef370e05ed3744a2e53b55bdd43e9594427c35514505f26e4691ba86c6dcff6d29d69110b15b9f84b0d8eb9ea7c03aaf24fa957314b89febf46a615f81ec031b12fe725f91af9d269873a69748  k = 0  while 1:  res = iroot(c+k\*n,e)  if(res[1] == True):  print(libnum.n2s(int(res[0])))  break  k=k+1  #encrypt\_exponent\_should\_be\_bigger |

Challenge4可以获得两组n，e，经典的共模攻击，exp：

|  |
| --- |
| import gmpy2  from Crypto.Util.number import long\_to\_bytes  n = 76284604418812528805019743643269218801063437262546923880432046778599082802406172284200822114293642550248873276927824927174183559217257008059090345088619769137133003644010021200449181932439239635449535901042255399740048928071220752234757658373046698452887627286261062848984321627727452727525588793257359121489  e = [107747, 89213]  c = [0x3060e6ff22d4e84897d0eaf9f410101fd425d13c0132e2dee2a0ef5eb0df2f9b16c9837fa2bd9c9e66ed2f77aa9b83c312cc8b8eb5a4e357a93d01b90eeb6c45e38165d444c2a4a37537384f4dabf775423cb51842873b5f78a500127d312bfb6c00a56cde0e3c14a23dcf7dd00c0c0bb0995d3f17c2999b3c13519478a2d69b,0x5be73c64b3a8c2007986e0cc4c2b424016ef800e3fb5a133543fcbef304795b3a92d072a23142a06467ba8daeed3701b5e00cec205af6185aaa2f5945acb11049912b4fc96183b93b41e122a263e2428df1f4f8331020a81c5917a418b0ced41ecca565e3e1d89b2f73ad3cc0db2bfba25c9a5c94d70e030fcb49e50e3a32a76]  c1 = c[0]  c2 = c[1]  e1 = e[0]  e2 = e[1]  s = gmpy2.gcdext(e1, e2)  s1 = s[1]  s2 = s[2]  if s1 < 0:  s1 = -s1  c1 = gmpy2.invert(c1, n)  elif s2 < 0:  s2 = -s2  c2 = gmpy2.invert(c2, n)  m = pow(c1, s1, n) \* pow(c2, s2, n) % n  print(long\_to\_bytes(m))  #never\_uese\_same\_modulus |

最后提交答案得到flag：



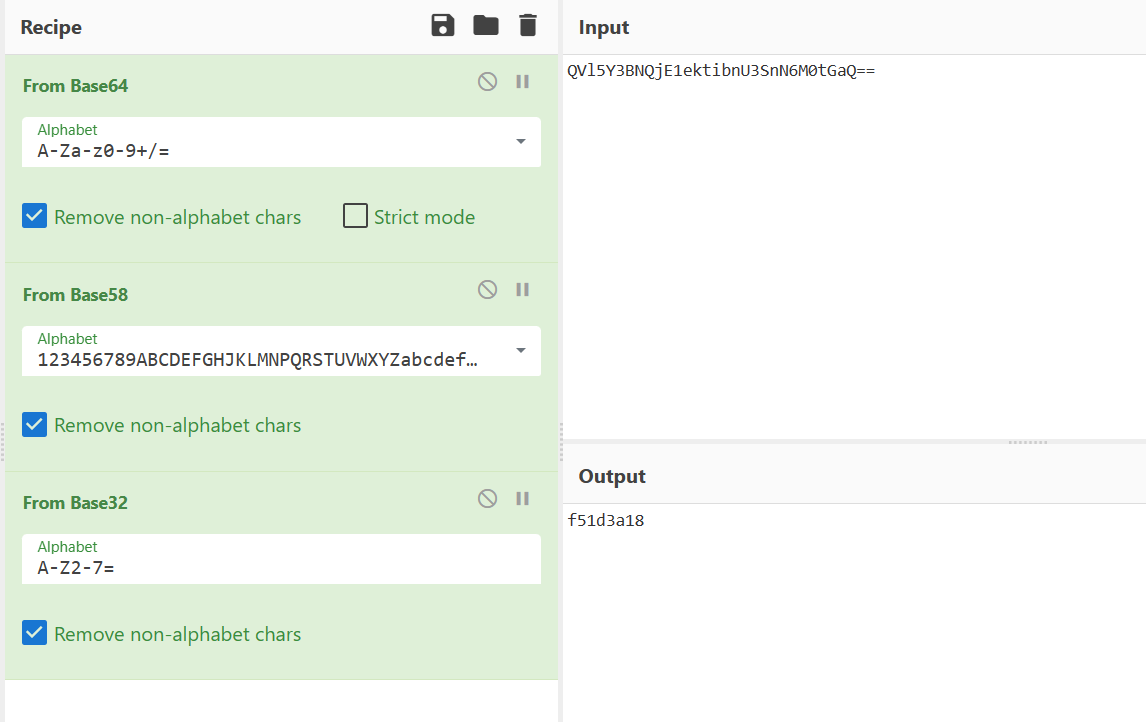
### MISC

#### Tetris Master&Tetris Master Revenge

都是非预期的，发现游戏结束后按N重新开始游戏分数是不会刷新的，因此可以刷分刷到50000分拿flag，当时flag出来没截图，懒得再搞了。

#### Sign In Pro Max

Part1：



Part2：



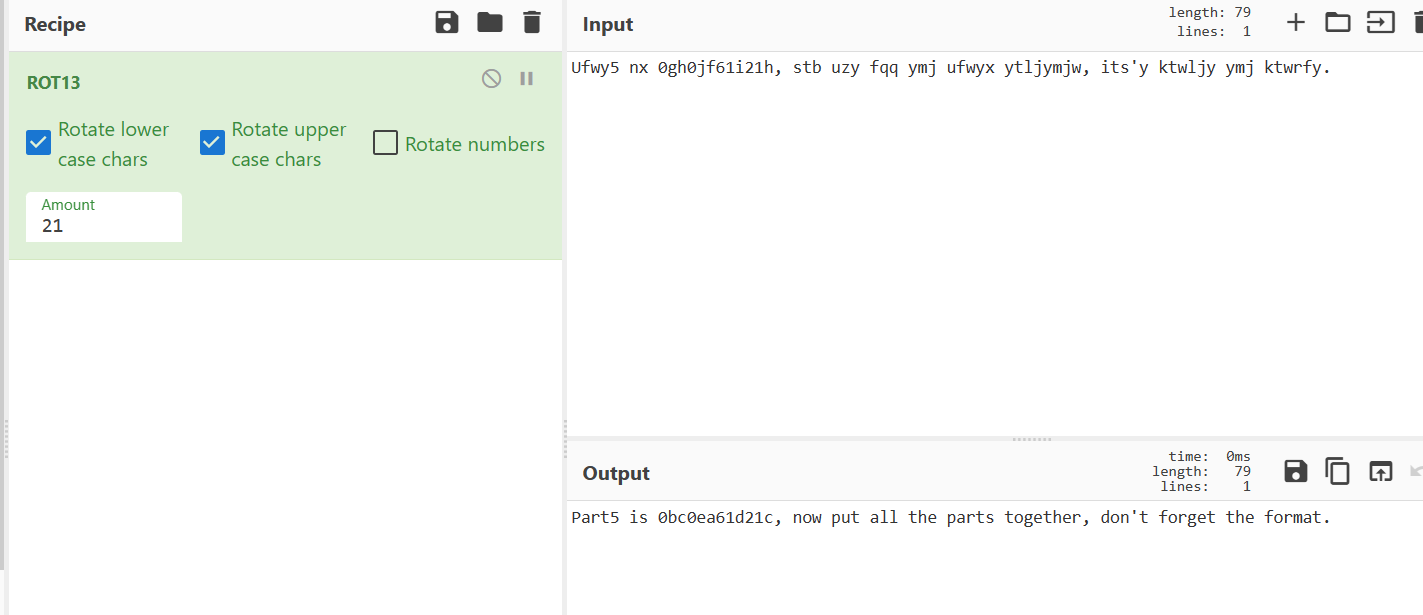
Part3：



Part4：



Part5：

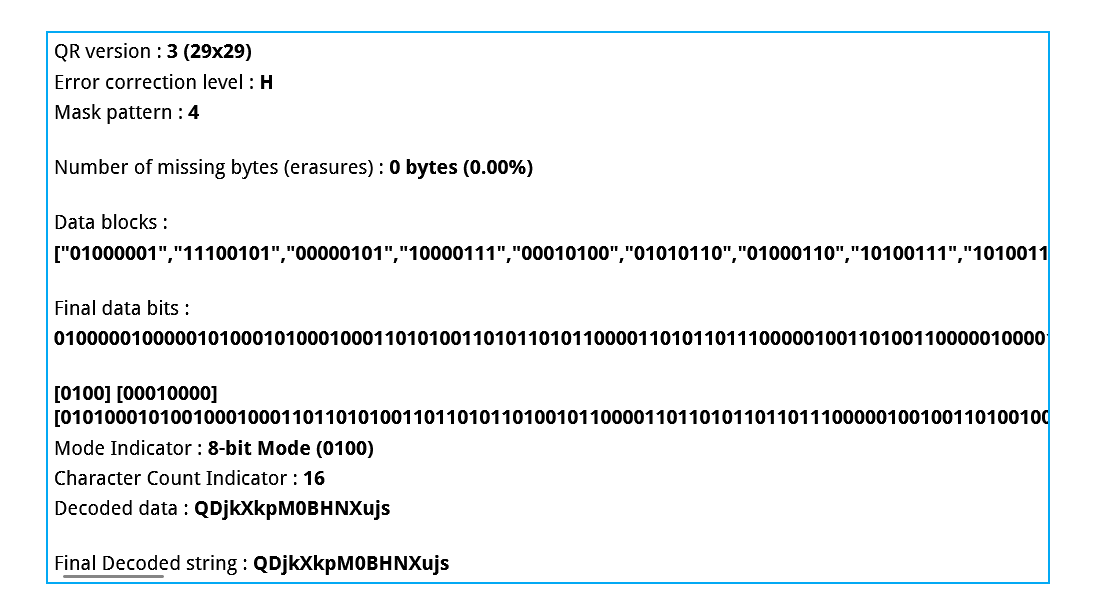


32位不是md5就是uuid，最后提示我们注意格式，那估计就是uuid了。

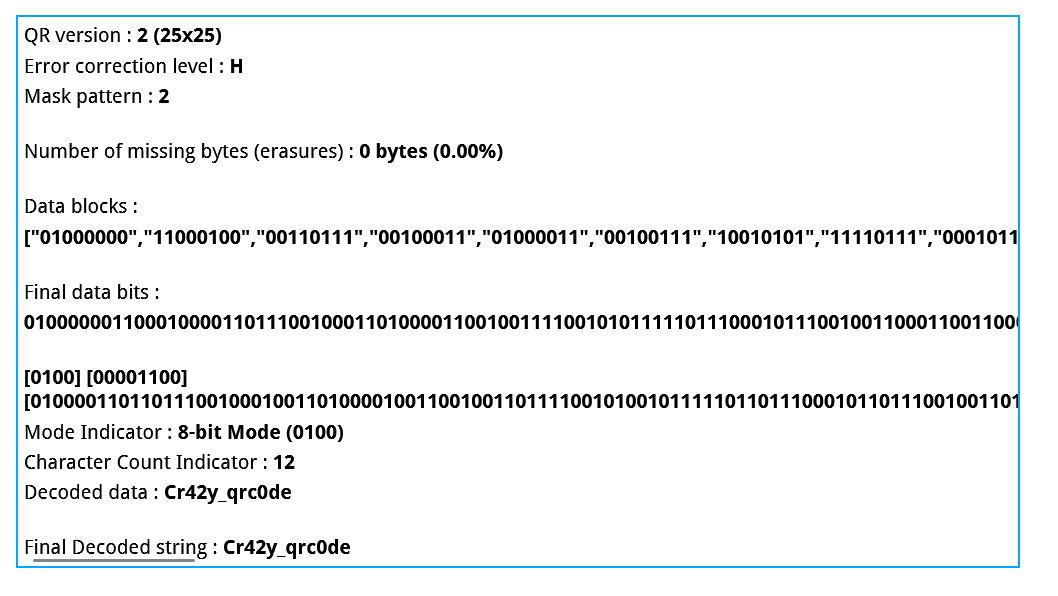
Flag：hgame{f51d3a18-f91c-4952-a3ed-0bc0ea61d21c}

#### crazy\_qrcode

附件是一个压缩包和一张图，压缩包加密，图片叫password，那就先从图片入手。发现图片是个二维码，但是直接扫描扫不出来，扔到qrazybox里分析。猜一手纠错等级的问题，调整后发现正确的纠错应该是H4：



解压后发现里面25张图片可以拼成一张二维码，图片0-25是按顺序的，但是经过了旋转，不过由于二维码的特性可以确认很多部分，剩下不能确认旋转的部分很少，可以爆破，最终成功得到flag：



### BLOCKCHAIN

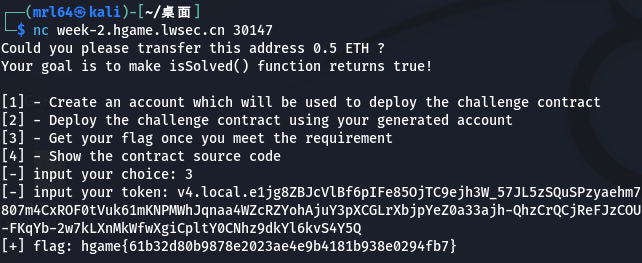
#### Transfer

查看合约代码，发现合约是空的。题目要求合约余额大于0.5ETH，但是不能直接对其转账。这用到selfdestruct函数。这是一个自毁函数，当我们调用这个函数时，它会使合约无效化并删除该地址的字节码，然后它会把合约里剩余的资金发送给参数指定的地址，比较特殊的是这笔资金的发送将无视合约的fallback函数。

攻击合约：

|  |
| --- |
| // SPDX-License-Identifier: GPL-3.0  pragma solidity ^0.8.7;  contract Attack {  uint b;  event log(uint);  constructor()public{  b=0;  }  receive()payable external{  b+=msg.value;  emit log(b);  }  function exploit(address payable \_target) public payable {  selfdestruct(\_target);  }  } |

使用remix部署后设置exploit的地址为合约地址，执行exploit即可得到flag。

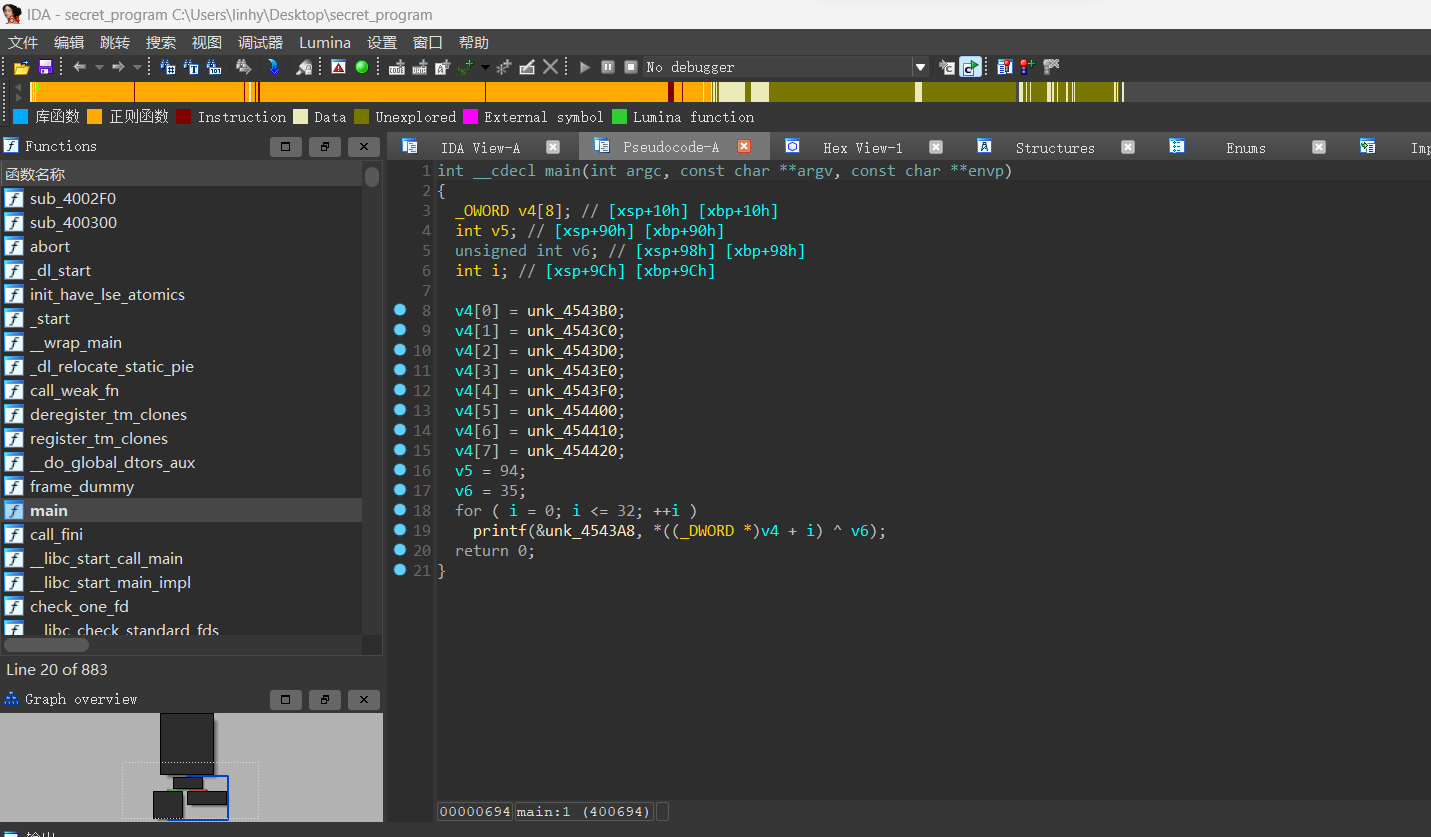


### IOT

#### Pirated router

附件是一个bin文件，提示是路由器，那么binwalk发现squashfs文件，使用firmware-mod-kit进行解压，得到squashfs-root文件。

这里在www里找可疑文件找了半天啥都没找到，后来一个一个找过去发现在/bin文件夹里面有一个叫secret\_program的文件，提取出来IDA分析：



很简单的一个逻辑，只要将v4的每个值异或35就行了：

|  |
| --- |
| enc = [0x4B,0x44, 0x42,0x4E,0x46,0x58, 0x56,0x4D,0x53,0x17,0x40,0x48,0x12,0x4D,0x44,0x7C,0x45,0x4A,0x51,0x4E,0x54,0x42,0x51,0x46,0x7C,0x12,0x50,0x7C,0x10,0x62,0x50,0x5A,0x5E]  flag = ''  for i in enc:  f = chr(i ^ 35)  flag += f  print(flag) |

Flag：hgame{unp4ck1ng\_firmware\_1s\_3Asy}