【N1CTF】Oflo WriteUp

原创

古月浪子 于 2020-10-22 00:30:30 发布 255 收藏

分类专栏： CTF

本文链接：https://blog.csdn.net/tqydyqt/article/details/109144964

二进制文件里面有很多call+e8+jmp+[return address += 1]这种操作，导致IDA无法处理，直接F5出来的几乎不能看，索性手动nop一下

```
void __fastcall main(__int64 a1, char **a2, char **a3)
{
  __int64 v3; // rdi
  signed int i; // [rsp+4h] [rbp-23Ch]
  char v5[32]; // [rsp+10h] [rbp-230h]
  char v6[512]; // [rsp+30h] [rbp-210h]
  __int64 v7; // [rsp+230h] [rbp-10h]
  unsigned __int64 v8; // [rsp+238h] [rbp-8h]

  v8 = __readfsqword(0x28u);
  memset(v6, 0, sizeof(v6));
  *v5 = 0LL;
  *&v5[8] = 0LL;
  *&v5[16] = 0LL;
  *&v5[24] = 0LL;
  sub_400BBF(&v7);
  if ( sub_4008B9(v6) == -1 )
    exit(0LL);
  read(0LL, v5, 19LL);
  qword_602048 = sub_400A69;
  v3 = main & 0xFFFFC000;
  mprotect(v3, 16LL, 7LL);
  for ( i = 0; i <= 9; ++i )
  {
    v3 = *(qword_602048 + i);
    *(qword_602048 + i) = v3 ^ v5[i % 5];
  }
  sub_400CBD(v3);
  if ( sub_400A69(v6, &v5[5]) )
    write(1LL, "Cong!\n", 6LL);
  exit(0LL);
}
```

对sub_400A69前10个字节进行动态patch，异或的内容是输入的前5个字符，根据flag的格式，猜测是n1ctf，IDAPython脚本：

```
a='n1ctf'
for i in range(10):
    PatchByte(0x400a69+i,Byte(0x400a69+i)^ord(a[i%5]))
```

跑完脚本以后可以正常F5该函数了

```
signed __int64 __fastcall sub_400A69(__int64 a1, __int64 a2)
{
  signed int i; // [rsp+1Ch] [rbp-24h]
  char v4; // [rsp+20h] [rbp-20h]
  char v5; // [rsp+21h] [rbp-1Fh]
  char v6; // [rsp+22h] [rbp-1Eh]
  char v7; // [rsp+23h] [rbp-1Dh]
  char v8; // [rsp+24h] [rbp-1Ch]
  char v9; // [rsp+25h] [rbp-1Bh]
  char v10; // [rsp+26h] [rbp-1Ah]
  char v11; // [rsp+27h] [rbp-19h]
  char v12; // [rsp+28h] [rbp-18h]
  char v13; // [rsp+29h] [rbp-17h]
  char v14; // [rsp+2Ah] [rbp-16h]
  char v15; // [rsp+2Bh] [rbp-15h]
  char v16; // [rsp+2Ch] [rbp-14h]
  char v17; // [rsp+2Dh] [rbp-13h]
  unsigned __int64 v18; // [rsp+38h] [rbp-8h]

  v18 = __readfsqword(0x28u);
  v4 = 53;
  v5 = 45;
  v6 = 17;
  v7 = 26;
  v8 = 73;
  v9 = 125;
  v10 = 17;
  v11 = 20;
  v12 = 43;
  v13 = 59;
  v14 = 62;
  v15 = 61;
  v16 = 60;
  v17 = 95;
  for ( i = 0; i <= 13; ++i )
  {
    if ( *(&v4 + i) != ((*(i + a1) + 2) ^ *(i + a2)) )
    {
      sub_400B16(a1);
      return 0LL;
    }
  }
  return 1LL;
}
```

对比14次，因此后面的输出应该还有14个字符

那么现在的问题是，如何找出拿来异或的a1呢？

可以看到，它是fork了一个子进程然后trace，根据PEEKDATA可以发现是在读取子进程的输出

子进程会调用cat /proc/version

```
wesker@ubuntu:~/Desktop$ ./oflo
Linux version 5.3.0-28-generic (buildd@lcy01-amd64-009) (gcc version 7.4.0 (Ubun
tu 7.4.0-1ubuntu1~18.04.1)) #30~18.04.1-Ubuntu SMP Fri Jan 17 06:14:09 UTC 2020
```

然后我们发现，前14个输出的字符永远是“Linux version ”

求出flag：

```
a = [53, 45, 17, 26, 73, 125, 17, 20, 43, 59, 62, 61, 60, 95]
s = ''
for i in range(14):
    s += chr((ord('Linux version '[i]) + 2) ^ a[i])
print('n1ctf' + s)
```