PWN dragon echo1 echo2 [pwnable.kr]CTF writeup题解系列16

原创

3riC5r 于 2020-01-06 21:23:57 发布 294 收藏
分类专栏： pwnable.kr CTF 文章标签： ctf pwn

本文链接：https://blog.csdn.net/fastergohome/article/details/103865069

由于pwnable.kr说明了不要将题解的利用脚本直接提出来，所以我就简单说下思路，本篇包括三个题目

目录

0x01 dragon

执行步骤

整数溢出漏洞：pDragon->HP 的类型是 signed byte，所以当超过127，就是负数了。这就是整数溢出漏洞

uaf漏洞：
1. malloc person
2. malloc dragon
3. free dragon
4. malloc plnput=dragon
((void (__cdecl *)(dragon *))pDragon->f_printinfo)(pDragon); = ((void (__cdecl *)(dragon *))plnput->f_printinfo)(plnput);

> ida struct

```
00000000 person          struc ; (sizeof=0x10, mappedto_5)
00000000 type            dd ?
00000004 HP              dd ?
00000008 MP              dd ?
0000000C f_printinfo     dd ?                    ; offset
00000010 person          ends
00000010
00000000 ; ---------------------------------------------------------------------------
00000000
00000000 dragon          struc ; (sizeof=0x10, mappedto_6)
00000000 f_printinfo     dd ?                    ; offset
00000004 type            dd ?
00000008 HP              db ? //signed byte
00000009 LifeRegeneration db ?
0000000A                 db ? ; undefined
0000000B                 db ? ; undefined
0000000C Damage          dd ?
00000010 dragon          ends
```

FightDragon函数

```
void __cdecl FightDragon(int nPersonType)
{
  char nCount; // al
  int nWin; // [esp+10h] [ebp-18h]
  person *pPerson; // [esp+14h] [ebp-14h]
  dragon *pDragon; // [esp+18h] [ebp-10h]
  void *pInput; // [ebp-Ch]

  pPerson = (person *)malloc(0x10u);
  pDragon = (dragon *)malloc(0x10u);
  nCount = Count++;
  if ( nCount & 1 )
  {
    pDragon->type = 1;
    pDragon->HP = 80;
    pDragon->LifeRegeneration = 4;
    pDragon->Damage = 10;
    pDragon->f_printinfo = (char *)PrintMonsterInfo;
    puts("Mama Dragon Has Appeared!");
  }
  else
  {
    pDragon->type = 0;
    pDragon->HP = 50;
    pDragon->LifeRegeneration = 5;
    pDragon->Damage = 30;
    pDragon->f_printinfo = (char *)PrintMonsterInfo;
    puts("Baby Dragon Has Appeared!");
  }
  if ( nPersonType == 1 )
  {
    pPerson->type = 1;
    pPerson->HP = 42;
    pPerson->MP = 50;
    pPerson->f_printinfo = (char *)PrintPlayerInfo;
    nWin = PriestAttack(pPerson, pDragon);
```

```
  }
  else
  {
    if ( nPersonType != 2 )
      return;
    pPerson->type = 2;
    pPerson->HP = 50;
    pPerson->MP = 0;
    pPerson->f_printinfo = (char *)PrintPlayerInfo;
    nWin = KnightAttack(pPerson, pDragon);
  }
  if ( nWin )
  {
    puts("Well Done Hero! You Killed The Dragon!");
    puts("The World Will Remember You As:");
    pInput = malloc(0x10u);
    __isoc99_scanf("%16s", pInput);
    puts("And The Dragon You Have Defeated Was Called:");
    ((void (__cdecl *)(dragon *))pDragon->f_printinfo)(pDragon);
  }
  else
  {
    puts("\nYou Have Been Defeated!");
  }
  free(pPerson);
}
```

0x02 echo1

漏洞利用函数：

```
__int64 echo1()
{
  char s; // [rsp+0h] [rbp-20h]

  ((void (__fastcall *)(object *))o->f_func1)(o);
  get_input(&s, 128);
  puts(&s);
  ((void (__fastcall *)(object *, signed __int64))o->f_func2)(o, 128LL);
  return 0LL;
}
```

get_input 可以做栈溢出
64位
root@mypwn:/ctf/work/pwnable.kr# checksec echo1
[*] '/ctf/work/pwnable.kr/echo1'
    Arch: amd64-64-little
    RELRO: Partial RELRO
    Stack: No canary found
    NX: NX disabled
    PIE: No PIE (0x400000)
    RWX: Has RWX segments

1、可以直接在栈中上传shellcode
2、或者泄漏got表函数地址，然后根据got表地址查找libc的system、binsh，再继续溢出执行system

0x03 echo2

漏洞利用函数：

```
__int64 echo2()
{
  char format; // [rsp+0h] [rbp-20h]

  (*((void (__fastcall **)(void *))o + 3))(o);
  get_input(&format, 0x20LL);
  printf(&format);
  (*((void (__fastcall **)(void *))o + 4))(o);
  return 0LL;
}
```

printf(&format);
这里有格式化字符串漏洞
可以泄漏got表，然后根据got表地址查找libc的system、binsh，再继续溢出执行system